Machine Learning **Spring 2021**

Lecture 1: Introduction to Machine Learning

Lecturer: Liwei Wang *Scribe: Junyi Guo, Zimai Guo, Kexing Zhou, Hongzhe Li*

1.1 Machine Learning

The notation "Machine Learning" is usually assisociated with Deep Learning and Big Data, but it is of much longer history. Machine Learning can be traced back to the 1960s, and it has a formal and clear declearation.

Definition 1.1 (Machine Learning) *Machine Learning means learning from experience or learning from data.*

1.1.1 History of Machine Learning

1. **Vapnic and Cherononkis** proposed VC theory from 1969 to 1971.
2. **Leslie Valient** developed the PAC (probabilistic approximately correct) theory, which won him the 2010 Turing Award.
3. **R.Schapire and Y.Freund** proposed boosting algorithm in 1990s, and received the Godel Price.
4. **Vapnik** brought out the SVM (support Vector Machine) algorithm.
5. **Pearl** put forward Graphic Models in 1980s and won the 2011 Turing Award.
6. **Yoshua Bengio, Geoffrey Hinton and Yann LeCun** Opened up a new field of deep learning.

1.1.2 Tasks and Brunches in Machine Learning

With decades of development, machine learning become a huge topic nowadays. Main brunches are listed below.

Supervised Learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs [1]. Many common tasks are included in supervised learning:

- *classification*: Assign a category to each item.
- *regression*: Predict a real value for each item.
- *ranking*: Order items according to some criterion.

Unsupervised Learning is a type of algorithm that learns patterns from untagged data. The philosophy of unsupervised learning is to discover patterns in the data itself. Famous unsupervised learning models include Boltzmann Machine, Autoencoder and VAE.

Reinforcement Learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward[1]. Reinforcement learning can be called "learning to control" and is an important approach in robotic systems and autonomous driving.

Online Machine Learning is a method of machine learning in which data becomes available in a sequential order and is used to update the best predictor for future data at each step. Online machine learning can form a closed-loop system that is common in commercial applications such as recommendation systems.

1.2 Recommended Books

1.2.1 Learning Theory

1. *Foundations of Machine Learning*
2. *Understanding Machine Learning from Theory to Algorithms*

1.2.2 Reinforcement Learning

1. *Reinforcement Learning Online Course*
2. **Not recommend to read** *Reinforcement Learning An Introduction*

1.2.3 Graphical Models

1. Koller's Online Course

1.2.4 Optimization

1. *Convex Optimization Algorithm and Complexity*

1.3 Formulation of Learning

The example below are based on a classification problem.

1. Collect Training data
   Symbols: $(x_1, y_1), \cdots, (x_n, y_n), x_i \in \mathcal{X}, y_i \in \mathcal{Y}$
   $\mathcal{X}$: instance space
   $\mathcal{Y}$: label space

2. Learn a Classifier
   Use algorithm $\mathcal{A}, \mathcal{A} : S \mapsto f, f : \mathcal{X} \to \mathcal{Y}$
3. Using f on test data
   $f(x_{n+1}), f(x_{n+2}), \cdots$

How can we ensure that f behaves well in test data?

1. iid: Independent homogeneous distribution, $(x_i, y_i) \sim \mathcal{D}_{\mathcal{X}\mathcal{Y}}$

How to evaluate the performance of f

1. Training Error: $\frac{1}{n}\sum_{i=1}^{n} I[y_i = f(x_i)]$
2. Test Error(Expected Error): $\mathsf{P}_{(x,y)\sim\mathcal{D}_{(\mathcal{X}\mathcal{Y})}}(y \neq f(x))$

Suppose $\mathcal{D}_{\mathcal{X}\mathcal{Y}}$ is known, the theoretical **optimal classifier** is the Bayes classifier defined as

$$f^*(x) = \underset{y\in\mathcal{Y}}{\operatorname{argmax}} \mathbb{P}(y|x)$$

1.4 Generation

Is it the best to make training error 0%?

Consider a fitting problem. For a training set $(x_1, y_1), \cdots, (x_n, y_n)$, we can choose to use either a linear model or a n-order polynomial model to fit. Obviously, the training error for the n-order polynomial model is much smaller than that for the linear model. However, small training error does not necessarily lead to small test error, which is known as the **overfitting** phenomenon. As the parameters of the model increase, the model becomes better fitted, but generation may decrease, which is the result we do not want.

How can we choose a better model?

There is no such thing as the best model for any given problem. It is possible for any model to perform well for a particular problem. Therefore, we should choose models carefully to achieve both good problem solving and good interpretability.

1.5 Basic Inequalities

1.5.1 Markov Inequality

For random value $X \geq 0$, if $EX < +\infty$, then:

$$\mathbb{P}(X \geq k) \leq \frac{EX}{k}, \quad \forall k > 0$$

1.5.2 Chebyshev Inequality

For random value X, if EX exists, and $EX^2 < +\infty$(which also means $\text{var}(X) < +\infty$), then:

$$\mathbb{P}(|X - EX| \geq t) \leq \frac{\text{var}(X)}{t^2}, \quad \forall t > 0$$

For random value X, if $EX, EX^2, ..., EX^r$ is all known, then

$$\mathbb{P}(X \geq k) \leq \min_j \frac{EX^j}{t^j}, \quad \forall j = 1, 2, \ldots, r$$

1.5.3 Moment Generating Function

Definition 1.2 (Moment Generating Function) *For random value X, if forall $n \in \mathbb{N}$, EX^n exists, we can define Moment Generating Function of X:*

$$M(t) := E(e^{tX}) = 1 + tEX + t^2\frac{EX^2}{2!} + t^3\frac{EX^3}{3!} + ...$$

We can use moment generating function to caculate a upperbound of $\mathbb{P}(X \geq k)$

Lemma 1.3 (upperbound of tail destribution) *$\forall t > 0$, we apply Markov Inequality to $\mathbb{P}(X \geq k)$:*

$$\begin{aligned}\mathbb{P}(X \geq k) &= \mathbb{P}(e^{tX} \geq e^{tk}) \\ &\leq e^{-tk}E(e^{tX}) \\ &= e^{-tk}M(t), \ (\forall t > 0)\end{aligned}$$

then we get the upperbound:

$$\mathbb{P}(X \geq k) \leq \inf_{t>0}\{e^{-tk}M(t)\}$$

References

[1] Stuart J. Russell, Peter Norvig (2010) Artificial Intelligence: A Modern Approach, Third Edition, Prentice Hall ISBN 9780136042594.

[2] https://en.wikipedia.org/wiki/Supervised_learning

[3] https://en.wikipedia.org/wiki/Unsupervised_learning

[4] https://en.wikipedia.org/wiki/Reinforcement_learning

[5] https://en.wikipedia.org/wiki/Online_machine_learning

[6] https://www.wikiwand.com/en/Markov's_inequality

[7] https://www.wikiwand.com/en/Chebyshev's_inequality

[8] https://www.wikiwand.com/en/Moment-generating_function

Machine Learning Theory **Spring 2021**

Lecture 2: Concentration Inequalities

Lecturer: Liwei Wang *Scribe: Baihe Huang, Bi'an Du, Zichen Wu, Yuanchen Qiu*

2.1 Recap

Recall Chernoff inequality and Chebyshev inequality from last lecture.

Theorem 2.1.1 (**Chernoff Inequality**). *Let X be a random variable that is non-negative with moment generating function $\mathbb{E}e^{tX}$. Then $\forall k > 0$,*

$$\mathbb{P}(X \geq k) \leq \inf_{t>0} e^{-tk}\mathbb{E}[e^{tX}].$$

Theorem 2.1.2 (**Chebyshev Inequality**). *Let random variables $X_1, X_2, \ldots, X_n \sim$ iid Bernoulli$(1, p)$. We have*

$$\mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} X_i - p \geq \epsilon) \leq \frac{\mathrm{Var}(\sum_{i=1}^{n} X_i/n)}{\epsilon^2} = \frac{p(1-p)}{n\epsilon^2}$$

Notice that Chebyshev inequality only uses second moment information of random variables, therefore its convergence rate is only inversely proportional.

From law of large number, we naturally expect

$$\mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} X_i - p \geq \epsilon) \leq e^{-O(n)}.$$

2.2 Concentration Inequalities

2.2.1 Backgrounds of information theory

Definition 2.2.1 (**Entropy**). *Let X be a random variable with probability mass function $p = (p_1, p_2, \ldots)$. The entropy of X is defined by*

$$H(X) := \begin{cases} \sum_i p_i \log_2 \frac{1}{p_i} \text{ (bits)} \\ \sum_i p_i \ln \frac{1}{p_i} \text{ (nats)} \end{cases}$$

Remark 2.2.2. *The entropy of a random variable is the average level of "information", "surprise", or "uncertainty" inherent in the variable's possible outcomes.*

Definition 2.2.3 (Relative Entropy). *For two probability mass functions $P = (p_1, p_2, \dots)$ and $Q = (q_1, q_2, \dots)$, the relative entropy from Q to P is defined to be*

$$D(P||Q) := \begin{cases} \sum_i p_i \log_2 \frac{p_i}{q_i} \text{ (bits)} \\ \sum_i p_i \ln \frac{p_i}{q_i} \text{ (nats)} \end{cases}$$

In particular, for two Bernoulli random variables $P = (p, 1-p), Q = (q, 1-q)$

$$D_B^{(e)}(p||q) := p \ln \frac{p}{q} + (1-p) \ln \frac{1-p}{1-q}$$

Remark 2.2.4. *Relative entropy measures the difference of two distributions, but this relation is asymmetric. Note that $D(P||Q) \geq 0$ for any P, Q and usually $D(P||Q) \neq D(Q||P)$.*

2.2.2 Chernoff Bound

Theorem 2.2.5. *Let $X_1, X_2, \dots, X_n$ be n iid Bernoulli random variables satisfying $\mathbb{E}[X_i] = p, \forall i \in [n]$. Then for all $\epsilon > 0$ we have*

$$\mathbb{P}(\frac{1}{n} \sum_{i=1}^{n} X_i - p \geq \epsilon) \leq e^{-n D_B^{(e)}(p+\epsilon||p)}.$$

Proof. By Chernoff inequality,

$$\mathbb{P}(\frac{1}{n} \sum_{i=1}^{n} X_i - p \geq \epsilon) \leq \inf_{t>0} e^{-t(p+\epsilon)} \mathbb{E}[e^{t \sum_{i=1}^{n} X_i}].$$

Notice that

$$\mathbb{E}[e^{t \sum_{i=1}^{n} X_i}] = \prod_{i=1}^{n} \mathbb{E}[e^{t X_i}] = (pe^t + 1 - p)^n. \tag{2.1}$$

It thus follows that

$$\begin{aligned} \mathbb{P}(\frac{1}{n} \sum_{i=1}^{n} X_i - p \geq \epsilon) &\leq \inf_{t>0} e^{-nt(p+\epsilon)} \cdot (pe^t + 1 - p)^n \\ &\leq e^{-n D_B^{(e)}(p+\epsilon||p)}. \end{aligned}$$

The last step is a simple calculation and left as homework. □

Theorem 2.2.6. *Let $X_1, \dots, X_n$ be n random variables satisfying $X_i \in [0, 1]$ and $\mathbb{E}[X_i] = p, \forall i \in [n]$. Then for all $\epsilon > 0$, we have*

$$\mathbb{P}(\frac{1}{n} \sum_{i=1}^{n} X_i - p \geq \epsilon) \leq e^{-n D_B^{(e)}(p+\epsilon||p)}.$$

Proof. Notice that exponent function is convex. By Jensen's inequality, we have

$$\mathbb{E}[e^{tX}] \leq \mathbb{E}[Xe^t] + \mathbb{E}[(1-X)e^0] = pe^t + 1 - p. \tag{2.2}$$

It thus follows that

$$\mathbb{E}[e^{t \sum_{i=1}^{n} X_i}] \leq (pe^t + 1 - p)^n.$$

Replacing Eq (2.1) by this inequality, the rest of the proof is the same as Theorem 2.2.5. □

Theorem 2.2.7. *Let $X_1, \ldots, X_n$ be n random variables satisfying $X_i \in [0,1]$ and $\mathbb{E}[X_i] = p_i, \forall i \in [n]$. Mark $p = \frac{1}{n}\sum_{i=1}^n p_i$, then for all $\epsilon > 0$ we have*

$$\mathbb{P}(\frac{1}{n}\sum_{i=1}^n X_i - p \geq \epsilon) \leq e^{-nD_B^{(e)}(p+\epsilon||p)}.$$

Proof. Notice that logarithmic function is concave. By Jensen's inequality, we have

$$\frac{\sum_{i=1}^n \ln(1-p_i+p_ie^t)}{n} \leq \ln(1-p+pe^t),$$

then combining this with Eq (2.2)

$$\begin{aligned}\mathbb{E}[e^{t\sum_{i=1}^n X_i}] &\leq \prod_{i=1}^n (1-p_i+p_ie^t) \\ &\leq (1-p+pe^t)^n.\end{aligned}$$

Replacing Eq (2.1) by this inequality, the rest of the proof is the same as Theorem 2.2.5. □

Remark 2.2.8. *The other side of tail bound can be proved similarly.*

Lemma 2.2.9. *(left as homework, find when the gap reaches infimum) $D_B^{(e)}(p+\epsilon||p) \geq 2\epsilon^2$.*

Plugging this lemma into Theorem 2.2.7, we have the following bound.

Theorem 2.2.10 (Additive Chernoff Bound). *Let $X_1, \ldots, X_n$ be n random variables satisfying $X_i \in [0,1]$ and $\mathbb{E}[X_i] = p_i, \forall i \in [n]$. Let $p = \frac{1}{n}\sum_{i=1}^n p_i$, then for all $\epsilon > 0$ we have*

$$\mathbb{P}(\frac{1}{n}\sum_{i=1}^n X_i - p \geq \epsilon) \leq e^{-2n\epsilon^2}.$$

Remark 2.2.11. *Note that Chernoff inequality requires X_i are mutually independent, while pairwise independence suffices for Chebyshev inequality.*

2.2.3 Hoeffding's inequality

Theorem 2.2.12 (Hoeffding's inequality). *Let $X_1, X_2, ...X_n$ be n independent random variables in $[a_i, b_i]$. Let $\mu = \frac{\sum_{i=1}^n E[X_i]}{n}$, then we have*

$$\mathbb{P}(\frac{1}{n}\sum_{i=1}^n X_i - \mu \geq \epsilon) \leq e^{\frac{-2n^2\epsilon^2}{\sum_{i=1}^n (b_i-a_i)^2}}.$$

2.2.4 Draw with/without replacement in a population

For N numbers $a_1, a_2, ..., a_N \in \{0,1\}$,let $p = \frac{1}{N}\sum_{i=1}^N a_i$. We consider the following cases.

Draw with replacement $x_1, x_2, ..., x_n$ are randomly drawn with replacement from $\{a_1, a_2, ..., a_N\}$. Then X_i are iid Bernoulli random variables with $\mathbb{E}[X_i] = p$. This case is essentially the same as Theorem 2.2.5.

Draw without replacement $y_1, y_2, ..., y_n$ are randomly drawn without replacement from $\{a_1, a_2, ..., a_N\}$. Now $y_1, \ldots, y_n$ are dependent. However, we can also show that:

$$\mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} y_i - p \geq \epsilon) \leq e^{-2n\epsilon^2}.$$

Proof. It suffices to proof

$$\mathbb{E}[e^{t\sum_{i=1}^{n} y_i}] \leq \mathbb{E}[e^{t\sum_{i=1}^{n} x_i}], \tag{2.3}$$

namely the moment generation function is consistently less than the case where we draw with replacement. To prove this we expand moment generation functions into polynomials

$$\mathbb{E}[e^{t\sum_{i=1}^{n} x_i}] = 1 + t\mathbb{E}[\sum_{i=1}^{n} x_i] + \frac{t^2}{2}\mathbb{E}[(\sum_{i=1}^{n} x_i)^2] + ...$$

Notice that every polynomial terms look like $f(t)\mathbb{E}[\prod_{i\in I} x_i] = f(t)\mathbb{P}(\prod_{i\in I} x_i = 1)$ where $f(t)$ is a polynomial function of t. For the case where numbers are drawn without replacement, we have $f(t)\mathbb{E}[\prod_{i\in I} y_i] = f(t)\mathbb{P}(\prod_{i\in I} y_i = 1)$. Now $\prod_{i\in I} y_i = 1$ holds only when $y_i = 1, \forall i \in T$, which happens with less probability when drawn without replacement. Then we have

$$\mathbb{E}(\prod_{i\in I} y_i) \leq \mathbb{E}(\prod_{i\in I} x_i)$$

and thus Eq (2.3) holds. □

2.2.5 McDiarmid Inequality

Chernoff bound is a special case of McDiarmid inequality.

Theorem 2.2.13 (McDiarmid's inequality). *Let $X_1, \ldots, X_n \in \mathcal{X}$ be n independent random variables and there exists constant $c_1, \ldots, c_n$ such that $f : \mathcal{X} \mapsto \mathrm{R}$ satisfies*

$$|f(x_1, ..., x_i, ..., x_n) - f(x_1, ..., x_i^{'}, ..., x_n)\ | \leq c_i$$

for all $i \in [n]$ and $\forall x_1, x_2, ..., x_n, x_i{}' \in \mathcal{X}$. Then for all $\epsilon > 0$ we have

$$\mathbb{P}(|f(x_1, ..., x_n) - \mathbb{E}[f(x_1, ..., x_n)]\ | \geq \epsilon) \leq \exp(\frac{-2\epsilon^2}{\sum_{i=1}^{n} c_i^2}).$$

2.3 VC Theory (Uniform Convergence Theory for ERM)

Binary classification We consider learning a hypothesis f from n data points $(x_1, y_1), \ldots, (x_n, y_n)$ sampled from $\mathcal{D}$, where $x_i \in \mathbb{R}^d, y_i \in \{\pm 1\}$. Training error can thus be written as $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq f(x_i)]$. We can also represent test error as $\mathbf{P}_{(x,y)\sim\mathcal{D}}(y \neq f(x))$.

Notice that $\mathbb{E}[\mathbb{1}(y_i \neq f(x_i))] = \mathbf{P}_{(x,y)\sim\mathcal{D}}(y \neq f(x))$. Generalization gap measures the gap between training loss and population loss. Fix f, $\mathbb{1}[y_i \neq f(x_i)]$ are iid Bernoulli variables. Thus we can show by Theorem 2.2.5 that for any $\epsilon > 0$

$$\mathbb{P}\left(\mathbf{P}_{(x,y)\sim\mathcal{D}}(y \neq f(x)) - \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[y_i \neq f(x_i)] \geq \epsilon\right) \leq e^{-2n\epsilon^2}.$$

This inequality seems to be conflicted with overfitting phenomenon. Actually, the function $\widehat{f}$ is learned depending on the training data so that $\mathbb{1}[y_i = \widehat{f}(x_i)]$ are not independent. We therefore cannot bound the error and may suffer from overfitting.

References

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[2] Hang Li. Statistical learning method. *Tsinghua university press*, pages 95–115, 2012.

[3] Rostamizadeh.A Mohri.M and TALWALKAR.A. *Foundations of Machine Learning*. MIT Press, 2012.

[4] Wikipedia. Concentration inequality. `https://en.wikipedia.org/wiki/Concentration_inequality`, 2021. [Online].

[5] Wikipedia. Inequalities in information theory. `https://en.wikipedia.org/wiki/Inequalities_in_information_theory`, 2021. [Online].

[6] Wikipedia. Vapnik–Chervonenkis theory. `https://en.wikipedia.org/wiki/Vapnik%E2%80%93Chervonenkis_theory`, 2021. [Online].

Machine Learning **Spring 2021**

Lecture 3: VC Theory: Generalization Theory for ERM

Lecturer: Liwei Wang *Scribe: Yixin Yang, Siqi Yang, Fan Chen, Lexing Zhang*

3.1 From finite number to infinite number

There are two measurements of the performance of a classifier:

- Training error: $P_S(y \neq f(x)) := \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{y_i \neq f(x_i)\}$
- Generalization error[1]: $P_{\mathcal{D}}(y \neq f(x)) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{1}\{y \neq f(x)\}\right]$

We are interested in the **generalization gap** $P_{\mathcal{D}}(y \neq f(x)) - P_S(y \neq f(x))$. Intuitively, it is closely related to the complexity of our model.

Even though it's unlikely that the number of classifiers can be finite in practice, it's important for us to first consider the finite case:

Theorem 3.1 *Suppose we have a finite model class $\mathcal{F}, |\mathcal{F}| < \infty$. For any classifier f in $\mathcal{F}$, we have the bound*

$$\mathbb{P}(P_{\mathcal{D}}(y \neq f(x)) - P_S(y \neq f(x)) \geq \epsilon) \leq |\mathcal{F}|e^{-2n\epsilon^2}$$

Proof:

$$\begin{aligned}
& \mathbb{P}(P_{\mathcal{D}}(y \neq f(x)) - P_S(y \neq f(x)) \geq \epsilon) \\
\leq\ & \mathbb{P}(\exists f \in \mathcal{F}, P_D(y \neq f(x)) - P_S(y \neq f(x)) \geq \epsilon) \\
\leq\ & \sum_{f\in\mathcal{F}} \mathbb{P}\left(P_{\mathcal{D}}(y \neq f(x)) - P_S(y \neq f(x)) \geq \epsilon\right) \qquad \text{(Union bound)} \\
\leq\ & |F|e^{-2n\epsilon^2} \qquad \text{(Chernoff Bound)}
\end{aligned}$$

■

Thus in conclusion, when $|\mathcal{F}|$ is finite, the generalization gap can be estimated by applying the union bound. That is, for a finite collection of events $A_1, A_2, \ldots, A_n$ we have

$$\mathbb{P}\left(\bigcup_i A_i\right) \leq \sum_i \mathbb{P}\left(A_i\right)$$

What if the model class $\mathcal{F}$ is infinite?

[1] We use $\mathcal{D}$ to denote the distribution of $z = (x, y)$.

3.2 VC-Theory

For the sake of simplicity, we first introduce some notations:

- for $f \in \mathcal{F}$ and $z = (x, y)$ sampled from the distribution $\mathcal{D}$, define $\phi_f(z) = \mathbb{1}\{f(x) \neq y\}$
- define $\Phi = \{\phi_f : f \in \mathcal{F}\}$

Our ultimate goal in this section is to bound the following quantity:

$$\mathbb{P}\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_i) - \mathbb{E}_z\left[\phi(z)\right]\right| \geq \epsilon\right)$$

3.2.1 Step1: Double Sample Trick

Lemma 3.2 *Let $X, X_1, X_2, \ldots, X_{2n}$ be i.i.d. Bernoulli random variables and $p = \mathbb{E}[X]$, $V_1 = \frac{1}{n}\sum_{i=1}^{n} X_i, V_2 = \frac{1}{n}\sum_{i=n+1}^{2n} X_i$. If $n \geq \frac{\ln 2}{\epsilon^2}$, then*

$$\frac{1}{2}\mathbb{P}\left(|V_1 - p| \geq 2\epsilon\right) \leq \mathbb{P}\left(|V_1 - V_2| \geq \epsilon\right) \leq 2\mathbb{P}\left(|V_1 - p| \geq \frac{\epsilon}{2}\right)$$

Proof: For the inequality on the right side,

$$|V_1 - V_2| \geq \epsilon \Rightarrow \left(|V_1 - p| \geq \frac{\epsilon}{2}\right) \vee \left(|V_2 - p| \geq \frac{\epsilon}{2}\right)$$

Using the union bound, we have

$$\mathbb{P}\left(|V_1 - V_2| \geq \epsilon\right) \leq \mathbb{P}\left(|V_1 - p| \geq \frac{\epsilon}{2}\right) + \mathbb{P}\left(|V_2 - p| \geq \frac{\epsilon}{2}\right) = 2\mathbb{P}\left(|V_1 - p| \geq \frac{\epsilon}{2}\right)$$

For the inequality on the left side, since $|V_1 - p| \geq 2\epsilon$ and $|V_2 - p| < \epsilon$ are independent events and

$$|V_1 - p| \geq 2\epsilon \wedge |V_2 - p| < \epsilon \Rightarrow |V_1 - V_2| \geq \epsilon$$

we have

$$\mathbb{P}\left(|V_1 - p| \geq 2\epsilon\right)\mathbb{P}\left(|V_2 - p| < \epsilon\right) \leq \mathbb{P}\left(|V_1 - V_2| \geq \epsilon\right)$$

Using Chernoff bound we can infer that $\mathbb{P}\left(|V_2 - p| < \epsilon\right) \leq \frac{1}{2}$ when $n \geq \frac{\ln 2}{\epsilon^2}$, thus

$$\frac{1}{2}\mathbb{P}\left(|V_1 - p| \geq 2\epsilon\right) \leq \mathbb{P}\left(|V_1 - V_2| \geq \epsilon\right)$$

■

Lemma 3.3 *If $n \geq \frac{\ln 2}{\epsilon^2}$, then*[2]

$$\begin{aligned}\frac{1}{2}\mathbb{P}\left(\sup_{\phi\in\Phi}\left|\mathbb{E}\left[\phi(Z)\right] - \frac{1}{n}\sum_{i=1}^{n}\phi(z_i)\right| \geq 2\epsilon\right) &\leq \mathbb{P}\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_i) - \frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_i)\right| \geq \epsilon\right) \\ &\leq 2\mathbb{P}\left(\sup_{\phi\in\Phi}\left|\mathbb{E}\left[\phi(Z)\right] - \frac{1}{n}\sum_{i=1}^{n}\phi(z_i)\right| \geq \frac{\epsilon}{2}\right)\end{aligned} \tag{3.1}$$

Proof: Left as homework. ■

[2] A slightly tricky point here is that: we are taking "sup" over a possibly uncountable family of random variables, so the event $\{\sup_{\phi\in\Phi}\left|\mathbb{E}\left[\phi(Z)\right] - \frac{1}{n}\sum_{i=1}^{n}\phi(z_i)\right| \geq \epsilon\}$ may not be $\mathbb{P}$-measurable. But this is just a purely technical issue in Probability Theory, and there is nothing to worry about.

3.2.2 Step2: Symmetrization

For sampling $z_i = (x_i, y_i)$ from the probability distribution $\mathcal{D}$, consider two equivalent sampling methods below:

1. Draw $z_1, \ldots, z_{2n}$ from $\mathcal{D}$ independently.
2. Draw set $\{z_1, \ldots, z_{2n}\}$ from $\mathcal{D}$ and randomly draw a permutation σ.

Consider the second sampling methods, we have

$$\mathbb{P}\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_i)-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_i)\right|\geq\epsilon\right)=\mathop{\mathbb{E}}_{\{z_1,\ldots,z_{2n}\}}\left[\mathbb{P}_\sigma\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_{\sigma(i)})\right|\geq\epsilon\right)\right]$$

Let $N^\Phi(z_1, \ldots, z_n)$ denotes $|\{(\phi(z_1), \ldots, \phi(z_n)) : \phi \in \Phi\}|$.

Using union bound, we have [3]

$$\mathbb{P}_\sigma\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_{\sigma(i)})\right|\geq\epsilon\right)\leq N^\Phi(z_1,\ldots,z_{2n})\mathbb{P}_\sigma\left(\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_{\sigma(i)})\right|\geq\epsilon\right)$$

Using "draw without replacement" Chernoff bound, we have

$$\begin{aligned}
&\mathbb{P}_\sigma\left(\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_{\sigma(i)})\right|\geq\epsilon\right)\\
=\;&2\mathbb{P}_\sigma\left(\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_{\sigma(i)})\geq\epsilon\right)\\
=\;&2\mathbb{P}_\sigma\left(\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-\frac{1}{2n}\sum_{i=1}^{2n}\phi(z_{\sigma(i)})\geq\frac{\epsilon}{2}\right)\\
=\;&2\mathbb{P}_\sigma\left(\frac{1}{n}\sum_{i=1}^{n}\phi(z_{\sigma(i)})-p\geq\frac{\epsilon}{2}\right)\leq 2e^{-2n(\frac{\epsilon}{2})^2}=2e^{-\frac{n\epsilon^2}{2}}
\end{aligned}$$

Therefore, we got

$$\begin{aligned}
\mathbb{P}\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_i)-\frac{1}{n}\sum_{i=n+1}^{2n}\phi(z_i)\right|\geq\epsilon\right)&\leq\mathop{\mathbb{E}}_{\{z_1,\ldots,z_{2n}\}}\left[N^\Phi(z_1,\ldots,z_{2n})\cdot 2e^{-\frac{n\epsilon^2}{2}}\right]\\
&=\mathop{\mathbb{E}}_{\{z_1,\ldots,z_{2n}\}}\left[N^\Phi(z_1,\ldots,z_{2n})\right]\cdot 2e^{-\frac{n\epsilon^2}{2}} \qquad (3.2)
\end{aligned}$$

[3] In fact, the RHS of the inequality below should be a summation over the set $\{(\phi(z_1), \ldots, \phi(z_{2n})) : \phi \in \Phi\}$ (i.e. over all dichotomies of $\{z_1, \cdots, z_{2n}\}$), which has $N^\Phi(z_1, \ldots, z_{2n})$ elements. But for the sake of simplicity and clarity, we follow the notation used in class, so from here on, when we write $\phi(z_1), \cdots, \phi(z_{2n})$, we refer to a specific element in $\{(\phi(z_1), \ldots, \phi(z_{2n})) : \phi \in \Phi\}$ (i.e. a dichotomy).

3.2.3 Step3: Estimate Growth Function

Now we have to deal with the term $\mathop{\mathbb{E}}_{\{z_1,\dots,z_{2n}\}}\left[N^\Phi(z_1,\dots,z_{2n})\right]$ in section 3.2. In the following part, we use $N^\Phi(n)$, the growth function of Φ, to represent the "worst case":

Definition 3.4 $N^\Phi(n) := \max_{z_1,\cdots,z_n\sim\mathcal{D}} N^\Phi(z_1,\cdots,z_n)$.

Concerning the behaviour of $N^\Phi(n)$ (as function of n), there are essentially two cases:

1. For all n, $N^\Phi(n) = 2^n$
2. At some n_0, $N^\Phi(n_0) < 2^{n_0}$

In case of the first one, all work we have done before will turn out to be useless, if not trivial[4]. And in the second case, by exactly the same argument, we shall expect $N^\Phi(n)$ to behaving well. So the main problem is: we know that $\forall n \geq n_0$, $N^\Phi(n)$ is strictly smaller than 2^n, but to what extent?

The methodology is to try with some easy examples. The insight is that there is a special case with enough generality:

Example 3.5 *Set $d = n_0 - 1$ and work with fixed sample $z_1,\cdots,z_n$. We know that for every $z_{i_1},\cdots,z_{i_{d+1}} (i_1 < \cdots < i_{d+1})$, there some strings [5] of length $d+1$ which is not "attainable" [6] at $z_{i_1},\cdots,z_{i_{d+1}}$. We **assume** that these "forbidden" strings can always be chosen to be $(0,\cdots,0)$.*

Then, any attainable string of length n can only contain at most d many 0's. Thus the the number of attainable strings, i.e. $N^\Phi(z_1,\cdots,z_n)$, is no more than $\sum_{k=0}^{d}\binom{n}{k}$.

In general setting, the forbidden patterns need not be the same for different sub-strings, but (intuitively) different patterns will "overlap" to create more forbidden patterns, and therefore we may expect there are less many possibility in the general case.

Lemma 3.6 *Assume that $N^\Phi(d+1) < 2^{d+1}$ and fix a sample $z_1,\cdots,z_n$ with $n \geq d+1$. Then we have $N^\Phi(z_1,\cdots,z_n) \leq \sum_{k=0}^{d}\binom{n}{k}$.* [7]

Proof: To bound the $N^\Phi(z_1,\cdots,z_n)$, we have to give a lower bound of the number of strings not attainable. The idea here is to look at the forbidden pattern of each sub-string, and then "extend" them. That is, we may naturally **extend** a forbidden pattern w (at $\{i_1 < \cdots < i_{d+1}\}$) to a set $E(w)$ of not attainable n-strings, e.g. we can extend the pattern $(0,1,1)$ at $2,3,5$ to $\{(*,0,1,*,1)\}$

Now, for each $\mathcal{I} = \{i_1 < \cdots < i_{d+1}\} \subset \{1,2,\cdots,n\}$, we choose a forbidden pattern $w_\mathcal{I}$, and extend it to a set of n-strings $E(w_\mathcal{I})$,[8] and we have to (lower) bound the $|\bigcup_\mathcal{I} E(w_\mathcal{I})|$:

[4] In fact, in such case, our hypothesis class is in some sense not learnable. For reference, see [FML12], Chapter 3, Section 5.

[5] From now on, when we talk about **(k-)string**, we refer to a vector (of length k), with each component being 0 or 1.

[6] We say a k-string is **not attainable** or it is a **forbidden pattern** at $\mathcal{I} = \{i_1 < \cdots < i_{d+1}\}$, if it is not contain in $\left\{\left(\phi(z_{i_1}),\cdots,\phi(z_{i_{d+1}})\right) : \phi \in \Phi\right\}$.

[7] When $n \leq d$, $\sum_{k=0}^{d}\binom{n}{k} = 2^n$, so the bound in fact holds for all n.

[8] In other word, we choose a pattern that cannot be produced by $z_{i_1},\cdots,z_{i_{d+1}}$, for each $d+1$ subset of $z_1,\cdots,z_n$.

First, consider the what happen at 1-st component. There are three possibility, e.g.

$$\{(*,1,0,*,\cdots)\}$$
$$\{(*,*,1,*,\cdots)\}$$
$$\{(1,0,*,0,\cdots)\}$$
$$\{(0,0,1,*,\cdots)\}$$
$$\{(0,*,*,1,\cdots)\}$$
$$\cdots$$

and we thus consider:

$$S_0 := \bigcup_{w_{\mathcal{I}} \text{ is 0 at 1-st}} E(w_{\mathcal{I}})$$
$$S_1 := \bigcup_{w_{\mathcal{I}} \text{ is 1 at 1-st}} E(w_{\mathcal{I}})$$
$$S_2 := \bigcup_{\text{no restriction at 1-st}} E(w_{\mathcal{I}})$$

Now for each $w_{\mathcal{I}}$ having 1 at 1-st component, we change its 1-st 1 to 0 to obtain $w'_{\mathcal{I}}$, and get $S'_1 = \bigcup E(w'_{\mathcal{I}})$. From definition, we have

$$|S'_1| = |S_1|, \quad S_0 \cap S_1 = \emptyset, \quad |S_2 \cap S_1| = |S_2 \cap S'_1|$$

and therefore we have $|S_0 \cup S_1 \cup S_2| \geq |S_0 \cup S'_1 \cup S_2|$.

Apply this procedure to each place, we can finally have each pattern $w_{\mathcal{I}}^{new}$ being the $d+1$-string $(0,\cdots,0)$, and thus

$$\begin{aligned} |\{\text{not attainable } n\text{-string}\}| \geq \left|\bigcup_{\mathcal{I}} E(w_{\mathcal{I}})\right| = |S_0 \cup S_1 \cup S_2| &\geq |S_0 \cup S'_1 \cup S_2| \\ &\geq \cdots \\ &\geq \left|\bigcup_{\mathcal{I}} E(w_{\mathcal{I}}^{new})\right| \\ &= \sum_{k=d+1}^{n} \binom{n}{k} \end{aligned}$$

Therefore $N^{\Phi}(z_1,\cdots,z_n) \leq 2^n - \sum_{k=d+1}^{n} \binom{n}{k} = \sum_{k=0}^{d} \binom{n}{k}$. ■

Corollary 3.7 *Combining inequalities 3.1, 3.2 and* $\sum_{k=0}^{d} \binom{n}{k} \leq \left(\frac{en}{d}\right)^d = \mathcal{O}(n^d)$*, we finally have*

$$\mathbb{P}\left(\sup_{\phi\in\Phi}\left|\frac{1}{n}\sum_{i=1}^{n}\phi(z_i) - \mathbb{E}_z\left[\phi(z)\right]\right| \geq \epsilon\right) \leq 2\left(\frac{2en}{d}\right)^d e^{-\frac{n\epsilon^2}{8}}$$

Definition 3.8 *Define the VC dimension of $\mathcal{F}$ to be the maximal d such that $N^{\Phi}(d) = 2^d$. If there is no such d, the VC dimension of $\mathcal{F}$ is defined to be $+\infty$.*

In other word, if we assume $\mathcal{F}$ has VC-dimension d, then there exist $z_1,\cdots,z_d$ satisfying $N^{\Phi}(z_1,\cdots,z_d) = 2^d$, and for all $z_1,\cdots,z_{d+1}$ drawn from $\mathcal{D}$, $N^{\Phi}(z_1,\cdots,z_{d+1}) < 2^{d+1}$.

References

[FML12] Mohri. M, Rostamizadeh. A and Talwalkar. A, Foundations of Machine Learning, *MIT Press* (2012)

[1] https://en.wikipedia.org/wiki/Boole's_inequality

[2] https://en.wikipedia.org/wiki/Vapnik-Chervonenkis_dimension

Machine Learning **Spring 2021**

Lecture 4: VC Theory, Practical Learning Algorithms, Game Theory

Lecturer: Liwei Wang *Scribe: Haoyu Li, Ting Lei, Zhenyu Du, Yang Hong, Zhen Wu*

4.1 VC Theory (Cont'd)

4.1.1 VC Theorem

Theorem 4.1 (VC Theorem) *For function space $\mathcal{F}$ with VC-dimension d, with probability at least $1-\delta$ over the random draw of training data,*

$$\mathbb{P}_{\mathcal{D}}(y \neq f(x)) \leq \mathbb{P}_{\mathcal{S}}(y \neq f(x)) + O\left(\sqrt{\frac{d\ln\frac{n}{d} + \ln\frac{1}{\delta}}{n}}\right)$$

holds uniformly for all $f \in \mathcal{F}$.

Proof: By Corollary 3.7 in Lecture 3,

$$\mathbb{P}\left(\sup_{f\in\mathcal{F}} |\mathbb{P}_{\mathcal{S}}(y \neq f(x)) - \mathbb{P}_{\mathcal{D}}(y \neq f(x))| \geq \epsilon\right) \leq \left(\frac{2en}{d}\right)^d e^{-cn\epsilon^2}$$

Let $\delta = \left(\frac{2en}{d}\right)^d e^{-cn\epsilon^2}$, then

$$\epsilon = \sqrt{\frac{d\ln\frac{n}{d} + \ln\frac{1}{\delta} + 2d}{cn}} = O\left(\sqrt{\frac{d\ln\frac{n}{d} + \ln\frac{1}{\delta}}{n}}\right)$$

Therefore, with probability at least $1-\delta$ over the random draw of training data,

$$\begin{aligned}\mathbb{P}_{\mathcal{D}}(y \neq f(x)) &\leq \mathbb{P}_{\mathcal{S}}(y \neq f(x)) + \epsilon \\ &= \mathbb{P}_{\mathcal{S}}(y \neq f(x)) + O\left(\sqrt{\frac{d\ln\frac{n}{d} + \ln\frac{1}{\delta}}{n}}\right)\end{aligned}$$

holds for all $f \in \mathcal{F}$. ■

Note that this inequality holds uniformly for all $f \in \mathcal{F}$, which implies that the bound of the deviation of the two probabilities given by this theorem is a 'worst case guarantee', or an algorithm independent bound.

Further, consider ERM(Empirical Risk Minimization) function on training data

$$\hat{f} := \arg\min_{f \in \mathcal{F}} \mathbb{P}_{\mathcal{S}}(y \neq f(x))$$

and the optimal classifier on the distribution

$$f^* := \arg\min_{f \in \mathcal{F}} \mathbb{P}_{\mathcal{D}}(y \neq f(x))$$

Applying the Theorem above, we immediately have a bound of the difference between the error of the two classifiers.

Theorem 4.2

$$\mathbb{P}_{\mathcal{D}}(y \neq \hat{f}(x)) \leq \mathbb{P}_{\mathcal{D}}(y \neq f^*(x)) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right)$$

Proof:

$$\begin{aligned}
\mathbb{P}_{\mathcal{D}}(y \neq \hat{f}(x)) &\leq \mathbb{P}_{\mathcal{S}}(y \neq \hat{f}(x)) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right) \\
&\leq \mathbb{P}_{\mathcal{S}}(y \neq f^*(x)) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right) \\
&\leq \left(\mathbb{P}_{\mathcal{D}}(y \neq f^*(x)) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right)\right) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right) \\
&= \mathbb{P}_{\mathcal{D}}(y \neq f^*(x)) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right)
\end{aligned}$$

■

We also wonder if $\mathbb{P}_{\mathcal{S}}(y \neq f(x)) + O\left(\sqrt{\frac{d \ln \frac{n}{d} + \ln \frac{1}{\delta}}{n}}\right)$ is a tight upper bound of $\mathbb{P}_{\mathcal{D}}(y \neq f(x))$. The answer to this question is positive, as the equality cases have been found.

4.1.2 VC dimension

To calculate the VC dimensions, we have to find a number d, so that there exists a set of d points which can be shattered by the classifiers, and any set of $d+1$ set can't be shattered. As a example, we introduce the VC dimension of linear classifiers below.

Theorem 4.3 *For* $\mathcal{F} = \left\{\operatorname{sgn}\left(w^T x + b\right) \mid w \in \mathbb{R}^d, b \in \mathbb{R}\right\}, d_{\mathcal{F}} = d+1$. *For* $\mathcal{F}^* = \left\{\operatorname{sgn}\left(w^T x\right) \mid w \in \mathbb{R}^d\right\}, d_{\mathcal{F}^*} = d$.

Proof: Let $x_1 = (1, 0, 0, \ldots, 0), x_2 = (0, 1, 0, \ldots, 0), \ldots, x_d = (0, 0, \ldots, 0, 1), x_{d+1} = (0, 0, \ldots, 0)$. Then we have:

$$N^{\mathcal{F}}(x_1, \ldots, x_{d+1}) = \left|\left\{(\operatorname{sgn}(w_1 + b), \ldots, \operatorname{sgn}(w_d + b), \operatorname{sgn}(b)) \mid w \in R^d, b \in R\right\}\right| = 2^{d+1}$$

Thus $d_{\mathcal{F}} \geq d+1$. Next we show $d_{\mathcal{F}} < d+2$, which indicates that $d_{\mathcal{F}} = d+1$.

$\forall x_1, \ldots, x_{d+2} \in \mathbb{R}^d, (x_1, 1), \ldots, (x_{d+2}, 1)$ are linear dependent. So there exists $c_1, \ldots, c_{d+2}, s.t$

$$c_1 (x_1, 1) + \ldots + c_{d+2} (x_{d+2}, 1) = 0$$

Let $c_{d+2} = 1$, then $\forall w \in \mathbb{R}^d, b \in \mathbb{R}$,

$$w^T x_{d+2} + b = \sum_{i=1}^{d+1} (-c_i) \left(w^T x_i + b\right)$$

Assume that $(\operatorname{sgn}(c_1), \ldots, \operatorname{sgn}(c_{d+1}), 1) \in \{(f(x_1), \ldots, f(x_{d+2}) \mid f \in \mathcal{F}\}$. Then $\exists w \in \mathbb{R}^d, b \in \mathbb{R}$, s.t. $w^T x_i + b$ and c_i have the same sign$(1 \leq i \leq d+1)$, and $\operatorname{sgn}\left(w^T x_{d+2} + b\right) = 1$, which is contradictory to $w^T x_{d+2} + b = \sum_{i=1}^{d+1} (-c_i) \left(w^T x_i + b\right) < 0$.

Thus $(\operatorname{sgn}(c_1), \ldots, \operatorname{sgn}(c_{d+1}), 1) \notin \{(f(x_1), \ldots, f(x_{d+2}) \mid f \in \mathcal{F}\}$, and $N^{\mathcal{F}}(x_1, \ldots, x_{d+2}) < 2^{d+2}$. So $d_{\mathcal{F}} = d+1$. Proof for $\mathcal{F}^*$ is similar. ■

Additionally, for binary classifiers, the VC dimension of Φ is equal to the VC dimension of $\mathcal{F}$.

Theorem 4.4 *$f \in \mathcal{F}$ are binary classifiers, let $\Phi = \{\phi_f(z) = I[y \neq f(x)] \mid f \in \mathcal{F}\}$, then $d_\Phi = d_{\mathcal{F}}$.*

Proof: Let $d_\Phi = d$, by definition we have $N^\Phi(d+1) < 2^{d+1}, N^\Phi(d) = 2^d$. We notice that $f(x) = \phi_f(x, 0)$, so we have:

$$N^{\mathcal{F}}(x_1, \ldots x_{d+1}) = N^\Phi((x_1, 0), \ldots, (x_{d+1}, 0)) \leq N^\Phi(d+1) < 2^{d+1}$$

Then $d_{\mathcal{F}} < d+1$.

Since $d_\Phi = d$, $\exists (x_1, y_1), \ldots, (x_d, y_d)$, s.t. $\forall w \in \{0,1\}^d, \exists f \in \mathcal{F}, (\phi_f(x_1, y_1), \ldots, \phi_f(x_d, y_d)) = w$, which implies that:

$$(f(x_1) + y_1, \ldots, f(x_d) + y_d) \equiv -w (\text{mod} 2) \Rightarrow (f(x_1), \ldots, f(x_d)) \equiv -w + (y_1, \ldots, y_d) (\text{mod} 2)$$

$\forall w \in \{0,1\}^d$, let $w' = -w + (y_1, \ldots, y_d), \exists f \in \mathcal{F}, s.t. (f(x_1), \ldots, f(x_d)) \equiv -w' + (y_1, \ldots, y_d) = w (\text{mod} 2)$. Then $(f(x_1), \ldots, f(x_d)) = w$. Thus $N^{\mathcal{F}}(d) = 2^d$, $d_{\mathcal{F}} = d$. ■

4.2 Practical Learning Algorithms

4.2.1 Linear Classfier

A linear classifier on $\mathbb{R}^d$ is defined as $\mathcal{F} = \{\operatorname{sgn}(w^\top x + b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$. Consider how to decide if a dataset is linear separable. That is, for $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{\pm 1\}$, given $(x_i, y_i)_{i=1}^n$, decide if $\exists f \in \mathcal{F}$, $\mathbf{s.t.} \sum_i I[f(x_i) \neq y_i] = 0$. In this specific problem, that's to say to decide if $\exists w \in \mathbb{R}^d, b \in \mathbb{R}, \mathbf{s.t.} y_i(w^\top x_i + b) >= 0 (\forall i \in [n])$. To solve this question efficiently, we can use the following linear programming(LP).

$$\begin{aligned} \max_{w,b,t} \quad & t \\ \mathbf{s.t.} \quad & y_i(w^\top x_i + b) \geq t \quad i \in [n] \end{aligned}$$

When $t > 0$, it's separable, since t is the minimum of all $y_i(w^\top x_i + b)$. For a separable training set, there might be many classifiers, to find the best one, we try to maximize the minimal distance.

$$\begin{aligned} \max_{w,b,t} \quad & t \\ \textbf{s.t.} \quad & y_i(w^\top x_i + b) \geq t \quad i \in [n] \\ & \|w\|_2 = 1 \end{aligned}$$

However this is neither a LP nor a QP(quadratic programming), we can't solve it directly with an efficient algorithm. Fortunately, the following QP is equivalent to that.

$$\begin{aligned} \min_{w,b,t} \quad & \tfrac{1}{2}\|w\|^2 \\ \textbf{s.t.} \quad & y_i(w^\top x_i + b) \geq 1 \quad i \in [n] \end{aligned}$$

The equivalence mentioned above could be obvious, i.e, if we divide the former inequation by t, and adjust the objective to $min \quad 1/t$,through some trivial substitution, the former one could have an equivalence relation with the latter one.

4.3 Appendix: Game Theory

In this section, we'll mainly talk about fundamental game theory. Game theory is the study of mathematical models of strategic interaction among rational decision-makers.Two-player Matrix Game is a simple model of game theory thus we address it as an introduction. We only consider zero-sum games, in which each participant's gains or losses are exactly balanced.

In the first step, we consider two players —— Alice and Bob —— choosing pure strategies when playing a zero-sum game. We'll find that the later mover will take advantage, which correspond to our intuition.

However, in the second step, our intuition is not that reliable. The game based on mixed strategy seems to come to an equilibrium point, making the later mover no longer take advantage at all.

Modern game theory began with the idea of mixed-strategy equilibria in two-person zero-sum games and its proof by John von Neumann. Von Neumann's original proof used the Brouwer fixed-point theorem on continuous mappings into compact convex sets, which became a standard method in game theory and mathematical economics.

In the 1950s, John Nash developed a criterion for mutual consistency of players' strategies known as the Nash equilibrium, applicable to a wider variety of games than the criterion proposed by von Neumann. Nash proved that every finite $n-$player, non-zero-sum (not just two-player zero-sum) non-cooperative game has a Nash equilibrium.

At the end of the notes, we'll talk about Sion's minimax theorem, which is a generalization of John von Neumann's minimax theorem.

So here we go. First of all, we define what a Two-player Matrix Game is.

Definition 4.5 (Two-player Matrix Game) *Let Alice and Bob are two players, $M = ((a_{ij}, b_{ij}))_{r\times c}$ is a matrix where every element is a pair of numbers $(a_{ij}, b_{ij}) \in \mathbb{R}^2$. Alice choose a row i while Bob choose a column j. Then the number a_{ij} and b_{ij} is the feedback of Alice and Bob respectively. Notice that M is known to Alice and Bob before making choices.*

As mentioned above, we only talk about Zero-sum Game. Here is its conception.

Definition 4.6 (Zero-sum Game) *The matrix M is defined as above, if $\forall i \in [1..r], j \in [1..c]$, we have $a_{ij} + b_{ij} = 0$, then we call the Two-player Matrix Game is a Zero-sum Game.*

That is to say, the benefit of one side of the opponent participating in the game process must mean the loss of the other side, so the sum of the gain and loss of both sides of the game must be zero. So M can be simplified to $M = (m_{ij})_{r\times c}$, where $m_{ij} \in \mathbb{R}$ indicates Alice should pay m_{ij} to Bob. Naturally, Alice want to minimize m_{ij} while Bob want to maximize it.

In order to maximize their own interests, both sides usually choose certain strategy.

Under each given information, if only one specific strategy can be selected, we call the strategy is a pure strategy. In the Two-player Matrix Game (if Alice go first), that is to say:

step 1: Alice choose a determined row i;

step 2: Bob observed Alice's strategy.According to information about M and Alice's strategy, Bob choose a determined column j;

step 3: Alice pays m_{ij} to Bob.

Then we say Alice and Bob use pure strategy.

If Alice go first, she always assume that if she choose row i, Bob will choose column j_i s.t. $m_{ij_i} = \max\limits_j m_{ij}$. So a natural strategy for Alice is to choose a row i_0 to minimize $\max\limits_j m_{ij}$. In this case, Alice's aim is $\min\limits_i \max\limits_j m_{ij}$.

In the same way, if Bob go first, Bob will choose a column j_0 to maximize $\min\limits_i m_{ij}$. So Alice's aim is $\max\limits_j \min\limits_i m_{ij}$.

We may ask: if the later mover will take advantage? i.e. $\forall$ given M, if we have $\min\limits_i \max\limits_j m_{ij} \geqslant \max\limits_j \min\limits_i m_{ij}$?

Intuitively, that's true. Because when Bob go first, as long as Alice always choose row i_0 (no matter what Bob choose), her loss cannot beyond $\min\limits_i \max\limits_j m_{ij}$. In mathematical language, that is to say $\min\limits_i \max\limits_j m_{ij} = \max\limits_j m_{i_0 j} \geqslant m_{i_0 j_0} \geqslant \min\limits_i m_{ij_0} = \max\limits_j \min\limits_i m_{ij}$. And we know the inequality sign can be obtained for some M, e.g $M := \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$.

So much for pure strategy. Here we come to mixed strategies where the players can choose a probability distribution on the pure strategies.

When Alice chooses the distribution vector $\vec{p} = (p_1, p_2, ..., p_r) \in [0,1]^r$ and Bob chooses $\vec{q} = (q_1, q_2, ..., q_c) \in [0,1]^c$ where $||\vec{p}|| = ||\vec{q}|| = 1$. The expectation of the value (In other words, Alice's loss or Bob's reward) is $\vec{p}^T M \vec{q}$. Obviously Alice's aim is to choose mixed strategy $\vec{p}_0$ and get $\min\limits_{\vec{p}} \max\limits_{\vec{q}} \vec{p}^T M \vec{q}$ if she go first; Bob will choose mixed strategy $\vec{q}_0$ and the expectation of the value will be $\max\limits_{\vec{q}} \min\limits_{\vec{p}} \vec{p}^T M \vec{q}$ if he go first. Similarly, we have $\min\limits_{\vec{p}} \max\limits_{\vec{q}} \vec{p}^T M \vec{q} = \max\limits_{\vec{q}} \vec{p}_0^T M \vec{q} \geqslant \vec{p}_0^T M \vec{q}_0 \geqslant \min\limits_{\vec{p}} \vec{p} M \vec{q}_0 = \max\limits_{\vec{q}} \min\limits_{\vec{p}} \vec{p}^T M \vec{q}$. It seems that the later mover will take advantage. However, the theorem given by John von Neuman conflict to our intuition, it tells us that there is no difference between go first and later.

Theorem 4.7 $\min\limits_{\vec{p}} \max\limits_{\vec{q}} \vec{p}^T M \vec{q} = \max\limits_{\vec{q}} \min\limits_{\vec{p}} \vec{p}^T M \vec{q}$.

The theorem can be proved via Brouwer's fixed point theorem or Farkas' lemma. In this course, however,

we'll present a proof with the knowledge of machine learning.

Theorem 4.8 (Sion's Minimax Theorem) *Suppose $f(x, y)$ is continuous, $\forall x \in \mathcal{X}, f(x, y)$ is concave, $\forall y \in \mathcal{Y}, f(x, y)$ is convex. Then*

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} f(x, y) = \max_{y \in \mathcal{Y}} \min_{x \in \mathcal{X}} f(x, y)$$

Furthermore, if optimal value of the left-hand side is achieved at (x_1, y_1), and right-hand side is achieved at (x_2, y_2), then consider $(x^*, y^*) := (x_1, y_2)$, we have $f(x_1, y_1) \leq f(x^*, y^*) \leq f(x_2, y_2)$, which means $f(x_1, y_1) = f(x^*, y^*) = f(x_2, y_2)$, implying the optimal value of both sides can be achieved at the same point (x^*, y^*). If the condition changes to "strictly concave" and "strictly convex", then the optimal point is unique.

From the optimality, it's clear to see that $\forall x, f(x, y^*) \geq f(x^*, y^*)$, $\forall y, f(x^*, y) \leq f(x^*, y^*)$. So (x^*, y^*) is a saddle point of this function.

Imagine two players A and B play a game on this function, A wants to minimize the left-hand side and B wants to maximize the other side, (x^*, y^*) is the strategy of players, then this strategy is a equilibrium of the game, which means both A and B can't gain a better utility if one changes its strategy while the other stays still.

John Nash proved that for a finite game with at least 2 players, there exists a mixed Nash equilibrium. However, computing the equilibrium in general case isn't easy. It's proved to be a $\mathcal{PPAD}$-complete question.

References

[1] Vapnik, Vladimir (2000). The nature of statistical learning theory. Springer.

[2] Blumer, A.; Ehrenfeucht, A.; Haussler, D.; Warmuth, M. K. (1989). "Learnability and the Vapnik–Chervonenkis dimension"

[3] https://en.wikipedia.org/wiki/Vapnik%E2%80%93Chervonenkis_dimension

Machine Learning **Spring 2021**

Lecture 5: Lagrange Duality

Lecturer: Liwei Wang *Scribe: Zhuming Shi, Yuhan Song, Le'an wang, Ying Wang, Yushuo Xiao*

5.1 Linear Separability

Last week we consider linear classifiers as

$$\mathcal{F} = \{sgn(\langle \mathbf{w}, \cdot \rangle + b) | \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{5.1}$$

then we formulate the optimization of the problem into a linear programming

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2} \| w \|^2 \\ \text{subject to} \quad & y_i(w^\mathsf{T} x_i + b) \geq 1 \end{aligned} \tag{5.2}$$

5.2 Lagrange Duality

We consider an optimization problem:

$$\begin{aligned} (P) \quad \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, i = 1, 2, ..., m \\ & h_j(x) = 0, j = 1, 2, ..., n \end{aligned} \tag{5.3}$$

where $f(x)$ and $g_i(x)$ are convex functions, and $h_i(x)$ are linear.

Let

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{n} \mu_j h_j(x). \tag{5.4}$$

Then the problem is equivalent to

$$\min_x \max_{\lambda \geq 0, \mu} L(x, \lambda, \mu). \tag{5.5}$$

For fixed x, $L(x, \lambda, \mu)$ is linear with respect to λ and μ, thus a concave function. For fixed $\lambda \geq 0$ and μ, since $f(x)$ and $g_i(x)$ are convex functions and $h_i(x)$ are linear, $L(x, \lambda, \mu)$ is a convex funtion with respect to x. Then by Sion's Minimax Theorem we know that

$$\min_x \max_{\lambda \geq 0, \mu} L(x, \lambda, \mu) = \max_{\lambda \geq 0, \mu} \min_x L(x, \lambda, \mu). \tag{5.6}$$

Then, given λ and μ, we need to find x to minimize $L(x, \lambda, \mu)$. Such x should satisfy

$$\frac{\partial L}{\partial x} = 0, \tag{5.7}$$

from which we can express x as a function of λ and μ:

$$x = \varphi(\lambda, \mu). \tag{5.8}$$

Then the primal problem P becomes another optimization problem D:

$$\begin{aligned} (D) \quad & \max_{\lambda,\mu} && L(\varphi(\lambda,\mu),\lambda,\mu) \\ & \text{subject to} && \lambda_i \geq 0, \qquad i = 1, 2, ..., m \end{aligned} \tag{5.9}$$

We call D the dual of P.

5.3 Apply to Linear Classifier

Let's return to problem 5.2. Now we have the optimization problem D, using 5.7, we have

$$L(w, b, \lambda) := \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \lambda_i \left(1 - y_i \left(w^\top x_i + b\right)\right) \tag{5.10}$$

Applying the Lagrange optimization to 5.10 we get

$$\frac{\partial L}{\partial \mathbf{w}} = \frac{1}{2} \times 2\mathbf{w} - \sum_{i=1}^{n} \lambda_i y_i x_i = 0 \tag{5.11}$$

and

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \lambda_i y_i = 0 \tag{5.12}$$

so we get

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i y_i x_i \tag{5.13}$$

and

$$\sum_{i=1}^{n} \lambda_i y_i = 0 \tag{5.14}$$

Now we use these result to 5.10, finally we have

$$\min_{\lambda} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j x_i^\mathsf{T} x_j - \sum_{i=1}^{n} \lambda_i \tag{5.15}$$

$$\text{subject to } \sum_{i=1}^{n} \lambda_i y_i = 0, 0 \leq \lambda_i \leq C \tag{5.16}$$

5.4 KKT Condition

Given general problem:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad i = 1, \dots m \\ & h_j(x) = 0, \quad j = 1, \dots n \end{aligned} \tag{5.3}$$

We have the Lagrangian function:

$$L(x, \lambda, \mu) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{n} \mu_j h_j(x) \tag{5.4}$$

and the Lagrange dual function:

$$g(\lambda, \mu) := \min_x L(x, \lambda, \mu) \tag{5.17}$$

The subsequent dual problem is:

$$\begin{aligned} \max_{\lambda,\mu} \quad & g(\lambda, \mu) \\ \text{subject to} \quad & \lambda \geq 0 \end{aligned} \tag{5.18}$$

The Karush-Kuhn-Tucker conditions or KKT conditions for (x^*, λ^*, μ^*) are:

(1) Stationarity: $\bigtriangledown_x L|_{x^*,\lambda^*,\mu^*} = 0$.

(2) Primal feasible: $\forall i, g_i(x^*) \leq 0;\ \forall j, h_j(x^*) = 0$.

(3) Dual feasible: $\forall i, \lambda_i \geq 0$.

(4) Complementary slackness: $\forall i, \lambda_i^* g_i(x^*) = 0$.

Then we have the following theorem:

Theorem 5.1 *x^*, λ^* and μ^* are primal and dual solutions if and only if they satisfy the KKT conditions.*

Proof:
The proof for necessity is trivial, we just prove its sufficiency.
Notice that

$$\begin{aligned} g(\lambda^*, \mu^*) &= f(x^*) + \sum_{i=1}^{m} \lambda_i^* g_i(x^*) + \sum_{j=1}^{n} \mu_j^* h_j(x^*) \\ &= f(x^*) \end{aligned}$$

where the first equality holds from stationarity, and the second holds from complementary slackness.
Then, due to the duality (here $\forall \lambda, \mu, x,\ g(\lambda, \mu) \leq f(x)$) between the primary and dual problem, x^*, λ^* and μ^* are primal and dual solutions. ■

5.5 Support Vector Machine

Assume training data is linear separable, we hope SVM can maximize the margin. It means:

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2} \| w \|^2 \\ \text{subject to} \quad & y_i(w^\mathsf{T} x_i + b) \geq 1 \end{aligned} \tag{5.2}$$

Its Lagrange multiplier is:

$$L(w, b, \lambda) := \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \lambda_i \left(1 - y_i \left(w^\top x_i + b\right)\right) \tag{5.10}$$

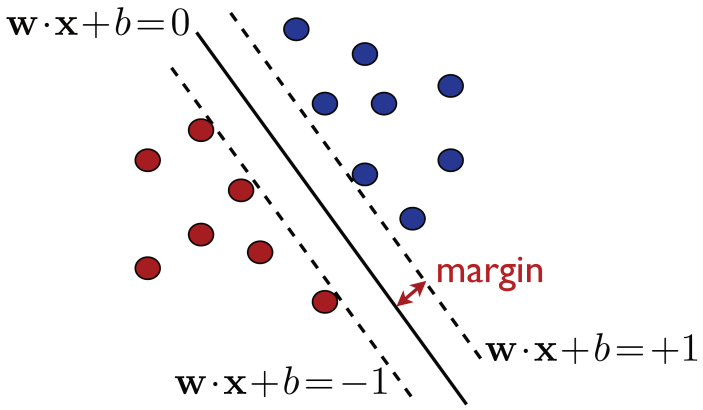
Consider its KKT condition. By stationary and complementary slackness, we have:

$$w^* = \sum_{i=1}^{n} \lambda_i^* y_i x_i \tag{5.19}$$

and

$$\lambda_i^* (y_i \left(w^{*\top} x_i + b^* \right) - 1) = 0 \tag{5.20}$$

So w^* is a linear combination of $y_i x_i$, with weights λ_is. And, it can be observed that only the points closest to the margin can satisfy the condition $y_i \left(w^{*\top} x_i + b^* \right) - 1 = 0$. So only the λ_is corresponding to these points can be positive, while other λ_is are all zero. That is to say, these few points that are closest to the margin decide w^*, and the whole SVM model, and we call these points support vectors (note that SVM is just the abbr. of support vector machine!).
So the dual problem matters. It is not because it helps to calculate the solutions, but that it provides insight into this model.



5.6 Soft Margin SVM

In many scenarios, the training data is linearly inseparable. We can solve this problem by allowing the SVM to fail at certain points. Specifically, we introduce Soft Margin SVM.

$$\min_{w,b,\epsilon} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \epsilon_i (\text{C is a fixed constant}) \tag{5.21}$$

$$\text{subject to } y_i (w^{\mathsf{T}} x_i + b) \geq 1 - \epsilon_i \tag{5.22}$$

The Lagrangian duality is as follows:

$$\min_{\lambda} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j x_i^{\mathsf{T}} x_j - \sum_{i=1}^{n} \lambda_i \tag{5.23}$$

$$\text{subject to } \sum_{i=1}^{n} \lambda_i y_i = 0, 0 \leq \lambda_i \leq C \tag{5.24}$$

The detailed derivation process is given below:

Let $g_i = 1 - \epsilon_i - y_i(w^{\mathsf{T}} x_i + b), \tilde{g}_i = -\epsilon_i$.The constraint condition is $g_i \leq 0, \tilde{g}_i \leq 0$.

$$\min_{w,b,\epsilon} \max_{\lambda,\mu} L(w, b, \epsilon, \lambda, \mu) \tag{5.25}$$

$$\text{s.t.} \lambda_i \geq 0, \mu_i \geq 0 \tag{5.26}$$

$$\text{where } L(w, b, \epsilon; \lambda, \mu) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} \epsilon_i + \sum_{i=1}^{n} \lambda_i g_i + \sum_{i=1}^{n} \mu_i \tilde{g}_i \tag{5.27}$$

For fixed λ, μ, find w^*, b^*, ϵ^* to minimize L.

Here we have,

$$\frac{\partial L}{\partial w_i} = w_i - \sum_{j=1}^{n} \lambda_j y_j x_{ji} = 0 \tag{5.28}$$

$$\frac{\partial L}{\partial b} = - \sum_{i=1}^{n} \lambda_i y_i = 0 \tag{5.29}$$

$$\frac{\partial L}{\partial \epsilon_i} = C - \lambda_i - \mu_i = 0 \tag{5.30}$$

then

$$L(w^*(\lambda, \mu), b^*(\lambda, \mu), \epsilon^*(\lambda.\mu); \lambda, \mu) = \frac{1}{2} \| w^* \|^2 + C \sum_{i=1}^{n} \epsilon_i + \sum_{i=1}^{n} \lambda_i g_i + \sum_{i=1}^{n} \mu_i \tilde{g}_i \tag{5.31}$$

$$= \frac{1}{2} \| w^* \|^2 + C \sum_{i=1}^{n} \epsilon_i + \sum_{i=1}^{n} \lambda_i [1 - \epsilon_i - y_i({w^*}^{\mathsf{T}} x_i + b)] - \sum_{i=1}^{n} \mu_i \epsilon_i \tag{5.32}$$

$$= \frac{1}{2} \| \sum_{i=1}^{n} \lambda_i y_i x_i \|^2 + C \sum_{i=1}^{n} \epsilon_i + \sum_{i=1}^{n} \lambda_i [1 - \epsilon_i - y_i (\sum_{j=1}^{n} \lambda_j y_j x_j^{\mathsf{T}} x_i + b)] - \sum_{i=1}^{n} \mu_i \epsilon_i \tag{5.33}$$

$$= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^{\mathsf{T}} x_j + C \sum_{i=1}^{n} \epsilon_i + \sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \lambda_i \epsilon_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^{\mathsf{T}} x_j - \sum_{i=1}^{n} \lambda_i y_i b - \sum_{i=1}^{n} \mu_i \epsilon_i \tag{5.34}$$

$$= - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^{\mathsf{T}} x_j + \sum_{i=1}^{n} (C - \lambda_i - \mu_i) \epsilon_i + \sum_{i=1}^{n} \lambda_i - b \sum_{i=1}^{n} \lambda_i y_i \tag{5.35}$$

$$= - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j x_i^{\mathsf{T}} x_j + \sum_{i=1}^{n} \lambda_i (\text{where } \sum_{i=1}^{n} \lambda_i y_i = 0,\ 0 \leq \lambda_i = C - \mu_i \leq C) \tag{5.36}$$

So we get the dual form as 5.23.

5.6.1 Kernel Methods Basics

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using kernel methods, implicitly mapping their inputs into high-dimensional feature spaces. The idea is to define a function $\Phi : \mathcal{X} \to \mathbb{H}$, which maps input x into a Hilbert space[1] $\mathbb{H}$ (called a *feature space*), expecting that $\Phi(x)$ becomes linearly separable in $\mathbb{H}$. However, noticing that x_i, x_j only appear in pairs as an inner product in the dual of the optimization problem of SVM (see equation 5.23), we only need to define a *kernel function*

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle. \tag{5.37}$$

This eliminates the need of explicitly calculating the function $\Phi(x)$, which can be inefficient and sometimes even impossible (if $\dim \mathbb{H} = \infty$).

Kernel methods are often illustrated by the XOR example. The data in Figure 5.1 (a) is clearly not linearly separable. But we can map the data into another space with the function

$$\Phi(x) = (\sqrt{2}x_1x_2, \sqrt{2}x_1). \tag{5.38}$$

The mapped points are shown in Figure 5.1 (b) and become linearly separable.

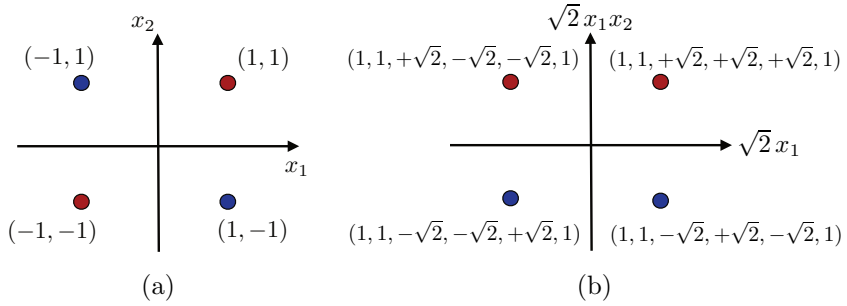


Figure 5.1: Illustration of the XOR classification problem and the use of kernel.

5.7 Boosting

Boosting is a family of learning methods that combines several predictors to create a more accurate one. We specifically focus on AdaBoost (short for *Adaptive Boosting*), which has been shown to be very effective in practice in some scenarios and is based on a rich theoretical analysis.

We first give the algorithm of AdaBoost in Algorithm 1.

The algorithm runs for T rounds. At each round, a new base classifier h_t is selected to minimize training error on S weighted by the distribution D_t. Then the distribution for the next round, D_{t+1}, is calculated. Intuitively, D_{t+1} is defined from D_t by increasing the weight on i if x_i is incorrectly classified, and decreasing it if x_i is correctly classified. After T rounds of boosting, the classifier returned is based on a non-negative linear combination of the base classifiers h_t. More accurate classifiers are assigned a larger weight and less accurate classifiers are assigned a smaller weight.

(*To be continued...*)

[1] A Hilbert space is a vector space equipped with an inner product, and that is complete (all Cauchy sequences are convergent).

Algorithm 1: AdaBoost algorithm

Input: Sample data $S = \{(x_1, y_1), \cdots, (x_n, y_n)\}$, base learning algorithm $\mathcal{A}$.
Result: A classifier $F(x)$.
Initialize: $D_1(i) = 1/n,\ i \in [n]$.
for $t = 1, 2, \cdots, T$ **do**
  Use $\mathcal{A}$ to learn a base classifier $h_t(x)$ on D_t.
  $\varepsilon_t \leftarrow \sum_{i=1}^{n} D_t(i) I[y_i \neq h_t(x_i)]$
  $\gamma_t \leftarrow 1 - 2\varepsilon_t$
  $\alpha_t \leftarrow \frac{1}{2} \ln \frac{1+\gamma_t}{1-\gamma_t}$
  $Z_t \leftarrow 2[\varepsilon_t(1-\varepsilon_t)]^{\frac{1}{2}}$
  $D_{t+1}(i) = \frac{D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i))}{Z_t}$
end
Output: $F(x) = \operatorname{sgn}\left[\sum_{t=1}^{T} \alpha_t h_t(x)\right]$.

References

[1] Rostamizadeh.A Mohri.M and TALWALKAR.A. Foundations of Machine Learning. MIT Press, 2012.

[2] https://www.cs.cmu.edu/~ggordon/10725-F12/slides/16-kkt.pdf

[3] https://en.wikipedia.org/wiki/Support-vector_machine

Machine Learning **Spring 2021**

Lecture 6: Boosting

Lecturer: Liwei Wang *Scribe: Qin Han, Zixuan Huo, Zijian Lan, Zezhou Wang, Zhengbo Xu*

6.1 Soft Margin SVM

Under many circumstances, training data is not linear separable. We can deal with this problem by allowing the SVM makes mistakes on some of the data. Specifically, we introduce *Soft-Margin SVM*:

$$\begin{aligned} (P) \quad \min_{w,b,\xi} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \\ s.t. \quad & y_i(w^T x_i + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \forall 1 \leq i \leq n \end{aligned} \tag{6.1}$$

Now we try to gain a deeper understanding of soft margin SVM. Denote $w^T x_1 + b$ as $f(x_i)$.

Obviously, (P) is equivalent to the following problem:

$$\min_{w,b} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}[1 - y_i(w^T x_i + b)]_+$$

where $[u]_+ := \max(u, 0)$

Before further discussion, we introduce some important loss functions. We know that

$$y_i f(x_i) > 0 \Leftrightarrow \text{the classification is correct}$$

So we can define:

Definition 1 *0-1 loss*

$$L_{0-1} = \begin{cases} 1, & if \quad y_i f(x_i) > 0 \\ 0, & if \quad y_i f(x_i) < 0 \end{cases}$$

Definition 2 *Hinge loss*

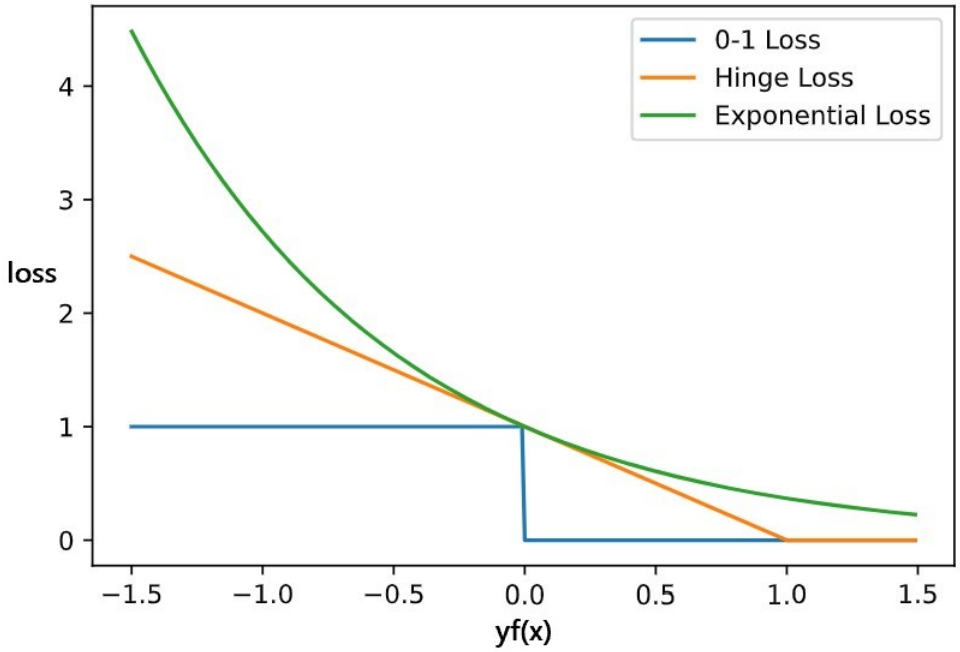
$$L_{hinge} = \begin{cases} 1 - y_i f(x_i), & if \quad y_i f(x_i) \leq 1 \\ 0, & if \quad y_i f(x_i) > 1 \end{cases}$$

Definition 3 *Exponential loss*

$$L_{exp} = e^{-y_i f(x_i)}$$

There are several important properties:

1. Hinge loss and exponential loss are both upper bounds for the 0-1 loss, which means when we minimize these two losses, 0-1 loss can be bounded.
2. Hinge loss and exponential loss are convex and continuous.
3. Once a data is classified correctly, its 0-1 loss reaches zero and no longer changes, causing the learning process to stop. This means $yf(x)$ is near zero. Meanwhile, the hinge loss will push $yf(x)$ to be above 1 and the exponential loss will push $yf(x)$ to infinity, which greatly improve generalization and give the model much robustness.



Back to our problem: $\min_{w,b} \quad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}[1 - y_i(w^T x_i + b)]_+$.

We can see $C\sum_{i=1}^{n}[1 - y_i(w^T x_i + b)]_+$ is the hinge loss of the classifier. The other term $\frac{1}{2}||w||^2$ is the $L2$ norm of the parameter w, so this is exactly $L2$ regularization. And now we have another perspective towards the *Soft-Margin SVM*: **Minimize Surrogate Loss function + Regularization**.

6.2 Boosting

Last week we have given the algorithm of AdaBoost.

Some propositions about the algorithm can help to understand why it performs well in practice.

Proposition 6.1 *Let* $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$, *we have:*

$$\prod_{t=1}^{T} Z_t = \frac{1}{n}\sum_{i=1}^{n} exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right) = \frac{1}{n}\sum_{i=1}^{n} exp\left(-y_i f(x_i)\right)$$

Where Z_t is normalization factor.

Proof:

$$\begin{aligned} Z_T &= \sum_{i=1}^{n} D_T(i) exp(-y_i \alpha_T h_T(x_i)) \\ &= \sum_{i=1}^{n} \frac{D_{T-1}(i) exp(-y_i \alpha_{T-1} h_{T-1}(x_i))}{Z_{T-1}} exp(-y_i \alpha_T h_T(x_i)) \\ \implies Z_{T-1} Z_T &= \sum_{i=1}^{n} D_{T-1}(i) exp\left(-y_i \alpha_{T-1} h_{T-1}(x_i) - y_i \alpha_T h_T(x_i)\right) \end{aligned}$$

Likewise, we can finally get

$$\begin{aligned} \prod_{t=1}^{T} Z_T &= \sum_{i=1}^{n} D_1(i) exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)\right) \end{aligned}$$

■

Proposition 6.2 *For α_t and Z_t in the algorithm, $t \in [T]$, we have:*

$$\alpha_t = arg\min_{\alpha} Z_t$$

Proof:

$$\min_{\alpha} Z_t = \min_{\alpha} \sum_{i=1}^{n} D_t(i)(e^{-y_i h_t(x_i)})^{\alpha}$$

According to the definition of classifier, $y_i h_t(x_i) = \begin{cases} 1 & y_i = h_t(x_i) \\ -1 & y_i \neq h_t(x_i) \end{cases}$

So we have

$$\sum_{i=1}^{n} D_t(i)(e^{-y_i h_t(x_i)})^{\alpha} = \epsilon_t e^{\alpha} + (1-\epsilon_t)e^{-\alpha} \geq 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

The equality holds if and only if

$$\epsilon_t e^{\alpha} = (1-\epsilon_t)e^{-\alpha} \iff \alpha = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t} = \frac{1}{2} ln \frac{1+\gamma_t}{1-\gamma_t}$$

So the proposition is proved, and we greedily assign values to α_t and Z_t as follows in our AdaBoost algorithm:

$$\alpha_t = \frac{1}{2} ln \frac{1+\gamma_t}{1-\gamma_t}$$

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

■

In Proposition 6.2, we notice that the selection of α_t is to minimize Z_t. Furthermore, Proposition 6.1 shows that the product of Z_t reflects the exponential loss $e^{-yf(x)}$ of classifier.

Given that the exponential loss is upper bound of the 0-1 loss, if we minimize the exponential loss, then the 0-1 loss decreases accordingly, which means the classifier may work better on the training set.

Then we consider the connections between the updated classifier and the original classifier.

Proposition 6.3

$$\sum_{i=1}^{n} D_{t+1}(i) \cdot \mathbb{I}[y_i \neq h_t(x_i)] = \frac{1}{2}$$

Proof:

$$\begin{aligned}
\sum_{i=1}^{n} D_{t+1}(i) \cdot \mathbb{I}[y_i \neq h_t(x_i)] &= \sum_{i=1}^{n} \frac{D_t(i) \cdot exp(-\alpha_t \cdot y_i h_t(x_i))}{Z_t} \cdot \mathbb{I}[y_i \neq h_t(x_i)] \\
&= \frac{\sum_{i=1}^{n} D_t(i) \cdot exp(\alpha_t) \cdot \mathbb{I}[y_i \neq h_t(x_i)]}{\sum_{i=1}^{n} D_t(i) \cdot exp(\alpha_t) \cdot \mathbb{I}[y_i \neq h_t(x_i)] + \sum_{i=1}^{n} D_t(i) \cdot exp(-\alpha_t) \cdot \mathbb{I}[y_i = h_t(x_i)]} \\
&= \frac{\epsilon_t \ exp(\alpha_t)}{\epsilon_t \ exp(\alpha_t) + (1-\epsilon_t) \ exp(-\alpha_t)}
\end{aligned}$$

The selection of α_t is to minimize z_t and so we have $\epsilon_t \ exp(\alpha_t) = (1-\epsilon_t) \ exp(-\alpha_t)$, thus

$$\sum_{i=1}^{n} D_{t+1}(i) \cdot \mathbb{I}[y_i \neq h_t(x_i)] = \frac{1}{2}$$

■

In Proposition 6.3, we notice that the performance of the updated classifier is as same as a random classifier on the original distribution, so we infer that the optimization directions of different classifiers are orthogonal.

Beyond the intuition, the error rate of the classifier and the speed of algorithm can be estimated.

Proposition 6.4 *Assume $\gamma_t \geq \gamma > 0$, for $t \in [T]$, then the following inequality holds:*

$$P_s(yf(x) \leq 0) \leq (1-\gamma^2)^{\frac{T}{2}}$$

Proof: We have known exponential loss $e^{-yf(x)}$ upper bounds 0-1 loss, so

$$\begin{aligned} P_s(yf(x) \leq 0) &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[y_i f(x_i) \leq 0] \\ &\leq \frac{1}{n}\sum_{i=1}^{n} exp(-y_i f(x_i)) \\ &= \prod_{t=1}^{T} z_t = \prod_{i=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \end{aligned}$$

From the constraint $\gamma_t \geq \gamma > 0$, we can infer that $0 \leq \epsilon_t \leq \frac{1-\gamma}{2} < \frac{1}{2}$, so

$$\begin{aligned} 2\sqrt{\epsilon_t(1-\epsilon_t)} &\leq 2\sqrt{\frac{1-\gamma}{2}(1-\frac{1-\gamma}{2})} \\ &= \sqrt{1-\gamma^2} \end{aligned}$$

Then we have

$$\begin{aligned} P_s(yf(x) \leq 0) &\leq \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \\ &\leq \prod_{t=1}^{T} \sqrt{1-\gamma^2} \\ &= (1-\gamma^2)^{\frac{T}{2}} \end{aligned}$$

■

Notice that the minimum positive value of $P_s(yf(x) \leq 0)$ is $\frac{1}{n}$. So if $(1-\gamma^2)^{\frac{T}{2}} < \frac{1}{n}$, we have $P_s(yf(x) \leq 0) = 0$, which means the classifier works totally correct on training set $\mathcal{S}$.

To reach that, we only need $O(log\ n)$ rounds, namely $T = O(log\ n)$, in the case of $\gamma > 0$.

We have discussed in the Proposition 6.3 that the choice of α indicates that each time h_t learns some orthogonal features. By normalizing α, learned f can be seen as the distance of these features to a hyperplane, which gives a new perspective of AdaBoost that when minimizing exponential loss it maximizes margin like SVM does.

Proposition 6.5 *Suppose (x,y) is a sample from sample space, denote α = (α_1,..., α_T) (with $\sum_{i=1}^{T} \alpha_i = 1$ and $\alpha_i > 0$) as normal vector of a hyperplane, and h(x) = ($h_1(x)$,...,$h_T(x)$) $\in R^T$ as a data point, then y f(x) gives a kind of distance from h(x) to the hyperplane $\{\xi : \alpha\xi = 0\}$.*

Hint: L_∞ distance.

6.3 Term Project

6.3.1 Project I: Teaching Dimension of Neural Network

1. **Without-Teaching Model**: Given a data space X, a hypothesis space $\mathcal{F}$. We can find the VC-Dimension of $\mathcal{F}$.

2. **Teaching Model**: Given a data space X, a hypothesis space $\mathcal{F}$. Teacher chooses classifier $f \in \mathcal{F}$ and test on $\Omega \subseteq X$. Learner can specify f by Ω.

There are some formal definitions for a hypothesis space C and it's data space X.

Definition 4 *Teaching Dimension of a classifier c in C*

$$TD(c, C) = \min_{b \in B_c} |b|$$

where

$$B_c = \{b \subseteq X | \forall c' \in C, c' = c \text{ or } c'|_b \neq c|_b\}$$

Definition 5 *Worst-case Teaching Dimension*

$$TD(C) = \max_{c \in C} TD(c, C)$$

Definition 6 *Best-case Teaching Dimension*

$$TD_{min}(C) = \min_{c \in C} TD(c, C)$$

Definition 7 *Recursive Teaching Dimension*

$$RTD(C) = \max_{0 \leq t \leq T} TD_{min}(C_t)$$

where

$$C_0 = C, \quad C_{t+1} = C_t - \{c \in C_t | TD(c, C_t) = TD_{min}(C_t)\}$$

and

$$T = \arg\min_t C_t = \emptyset$$

Assignment: Try to calculate teaching dimension of a deep neural network.

References

[1] Quadratic Upper Bound for Recursive Teaching Dimension of Finite VC Classes

6.3.2 Project II: Certified Robustness

Adversarial examples are inputs to machine learning models designed to intentionally fool them or to cause mispredictions. One of the threats of adversaries can be modeled as Gradient access.

So far, some countermeasures have been proposed. One of the popular methods is adversarial training, which repeatedly generates adversarial samples for model training. However, its shortcomings are obvious. It can only protect against specific attacks, and it also lacks theoretical guarantees.

Assignment: Design an algorithm to reach the state of the art on some data set, for example, MINIST, FASHION-MINIST, and CIFAR.

References

[1] Towards Certifying l_∞ Robustness using Neural Networks with l_∞-dist Neurons

[2] https://www.microsoft.com/en-us/research/blog/adversarial-robustness-as-a-prior-for-better-trans

6.4 Bagging (Booststrap Aggregating)

6.4.1 Algorithm

- **Step 1:** Given a standard training set $\mathcal{S} = \{x_1, x_2, ..., x_n\}$.
- **Step 2:** Generate a new training set $\mathcal{S}_i = \{x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)}\}$ by drawing with replacement from $\mathcal{S}$; train the model with $\mathcal{S}_i$ to obtain a new classifier h_i.
- **Step 3:** Repeat **Step 2** N times. Combine the classifiers by mean aggregator: $\frac{1}{N}\sum_{i=1}^{N} h_i(x)$.

We can easily notice that Booststrap Aggregating is more effective in the case that the base classifier is unstable, which means performance varies greatly with data.

References

[1] MOHRI.M, ROSTAMIZADEH.A and TALWALKAR.A, Foundations of Machine Learning, MIT Press (2012)

[2] ZHIHUA ZHOU, Machine Learning, Tsinghua University Press (2016)

[3] https://en.wikipedia.org/wiki/Hinge_loss

[4] https://en.wikipedia.org/wiki/AdaBoost

[5] https://en.wikipedia.org/wiki/Bootstrap_aggregating

[6] https://en.wikipedia.org/wiki/Random_forest

Machine Learning Theory **Spring 2021**

Lecture 7: PAC-Bayesian Theory

Lecturer: Liwei Wang *Scribe: Binghui Li, Shiji Xin, Fang Sun, Qizhe Zhang, Mi Yan*

7.1 Review

VC Theory focuses on the uniform convergence for all classifiers in hypothesis space $\mathcal{F}$. Specifically, with probability $1-\delta$,

$$P_D \leqslant P_S + O\left(\sqrt{\frac{d\ln n + \ln 1/\delta}{n}}\right) \tag{6.1}$$

Where P_D is the test error w.r.t. $f \in \mathcal{F}$, P_S is the training error w.r.t. $f \in \mathcal{F}$, d is the VC-dim of $\mathcal{F}$.
Notice that the result is independent of the distribution D and the learning algorithm A, and only depends on the hypothesis space $\mathcal{F}$ and sample size n.
However, VC Theory fails to account for the great generalization ability of neural networks. For a neural network, the number of parameters is usually far greater than that of training data. In such a case, the generalization bound yielded by VC Theory is scarcely meaningful. Perhaps randomness plays a vital role behind the huge success of neural networks. After all, stochastic gradient descent (SGD), rather than gradient descent (GD), is the common method for optimizing neural networks. [1–3]

7.2 Comparison of Frequentist and Bayesian

There are two points of view of statistical inference: Frequentist and Bayesian. Here is a brief comparison:

	Frequentist	Bayesian
Views of Probability	Law of Large Numbers	Degree of Belief
Parameters Estimation	Maximum Likelihood Estimate	Maximum a posteriori estimation $P(\theta\|x) = \frac{P(x\|\theta)P(\theta)}{P(x)}$
Output of Learning	a classifier f	a distribution of classifiers $\mathcal{Q}$
Prior	Hypothesis space $\mathcal{F}$	distribution of classifiers $\mathcal{P}$
Performance	$\mathrm{Err}_D(f)$	$\mathbb{E}_{f\sim Q}[\mathrm{Err}_D(f)]$
Generalization	VC Theory uniform convergence for all classifiers in a hypothesis space	PAC-Bayesian Theory for all distributions of classifiers
Gap between $\mathrm{Err}_D(f)$ and $\mathrm{Err}_S(f)$	$O\left(\sqrt{\frac{d\ln n + \ln(1/\delta)}{n}}\right)$	$O\left(\sqrt{\frac{D(\mathcal{Q}\|\mathcal{P}) + \ln(3/\delta)}{n}}\right)$

In the Frequentist view, the gap between test error and training error only depends on hypothesis space $\mathcal{F}$ and sample size, i.e. w/ prob $1-\delta$, $\forall h \in \mathcal{F}$,

$$\mathrm{Err}_D(h) \leqslant \mathrm{Err}_S(h) + O\left(\sqrt{\frac{d \ln n + \ln(1/\delta)}{n}}\right)$$

Where $d = \mathrm{VCD}(\mathcal{F})$.

7.3 The main theorem of PAC-Bayesian Theory

In this section, we are going to present the main theorem of PAC-Bayesian Theory.

Theorem 6.1 (PAC-Bayesian) *For any fixed prior distribution of classifiers $\mathcal{P}$, with probability $1-\delta$ over the random draw of training dataset S of size n, the following inequality holds uniformly for all distributions (i.e. stochastic classifier) $\mathcal{Q}$:*

$$\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)] \leqslant \mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_S(h)] + \sqrt{\frac{D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) + \ln(3/\delta)}{n}} \tag{6.2}$$

Here we denote $\mathbb{P}_{(x,y)\sim D}[y \neq h(x)]$ by $\mathrm{Err}_D(h)$ and $\mathbb{P}_{(x,y)\sim S}[y \neq h(x)]$ by $\mathrm{Err}_S(h)$, $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) \coloneqq \mathbb{E}_{h\sim Q}\left[\ln \frac{q(x)}{p(x)}\right]$ is the KL-divergence between $\mathcal{Q}$ and $\mathcal{P}$.

Intuitively, since $\mathcal{P}$ is independent of the dataset $S \sim D^n$, we can bound the gap between $\mathbb{E}_{h'\sim\mathcal{P}}[\mathrm{Err}_D(h')]$ and $\mathbb{E}_{h'\sim\mathcal{P}}[\mathrm{Err}_S(h')]$ by Chernoff bound, so we only need to bound the gap between $\mathbb{E}_{h\sim Q}[\mathrm{Err}_D(h)]$ and $\mathbb{E}_{h'\sim P}[\mathrm{Err}_D(h')]$. Before proving the theorem, we list some useful lemmas here.

Lemma 6.2 (Change of Measure) *For all distribution $\mathcal{P}, \mathcal{Q}$ over hypothesis space $\mathcal{F}$ and all function $f : \mathcal{F} \to \mathbb{R}$, we have*

$$\mathbb{E}_{h\sim\mathcal{Q}}[f(h)] \leqslant \ln \mathbb{E}_{h'\sim\mathcal{P}}\left[e^{f(h')}\right] + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) \tag{6.3}$$

Proof: In fact,

$$\begin{aligned}
\mathrm{RHS} - \mathrm{LHS} &= \ln \mathbb{E}_{h'\sim\mathcal{P}}\left[e^{f(h')}\right] + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) - \mathbb{E}_{h\sim\mathcal{Q}}[f(h)] \\
&= \mathbb{E}_{h\sim\mathcal{Q}}\left[\ln \frac{q(h)}{p(h)}\right] - \mathbb{E}_{h\sim\mathcal{Q}}[f(h)] + \ln \mathbb{E}_{h'\sim\mathcal{P}}\left[e^{f(h')}\right] \\
&= \mathbb{E}_{h\sim\mathcal{Q}}\left[\ln \frac{q(h)}{\frac{p(h)e^{f(h)}}{\mathbb{E}_{h'\sim\mathcal{P}}e^{f(h')}}}\right] \\
&= D_{\mathrm{KL}}\left(\mathcal{Q}\|\frac{p(h)e^{f(h)}}{\mathbb{E}_{h'\sim\mathcal{P}}e^{f(h')}}\right) \geqslant 0
\end{aligned}$$

■

Lemma 6.3 *For any $\delta > 0$,*

$$\mathbb{P}_{S\sim D^n}\left[\mathbb{E}_{h\sim\mathcal{P}}[e^{n(\mathrm{Err}_D(h)-\mathrm{Err}_S(h))^2}] \geqslant 3/\delta\right] \leqslant \delta \tag{6.4}$$

Proof: We begin our proof by considering a bound for a fixed $h \sim \mathcal{P}$.

$$\mathbb{E}_{S\sim D^n}\left[e^{n(\mathrm{Err}_D(h)-\mathrm{Err}_S(h))^2}\right] \leqslant 3$$

For simplicity, let $\Delta \coloneqq |\mathrm{Err}_D(h) - \mathrm{Err}_S(h)|$. Using Chernoff bound, we have

$$\mathbb{P}_{S\sim D^n}[\Delta \geqslant \varepsilon] \leqslant 2\exp(-2n\varepsilon^2)$$

Then,

$$\begin{aligned}
\mathbb{E}_{S\sim D^n}[e^{n\Delta^2}] &= \int_0^\infty \mathbb{P}_{S\sim D^n}\left[e^{n\Delta^2} \geqslant t\right] \mathrm{d}t \\
&= \int_1^\infty \mathbb{P}_{S\sim D^n}\left[e^{n\Delta^2} \geqslant t\right] \mathrm{d}t + \int_0^1 \mathbb{P}_{S\sim D^n}\left[e^{n\Delta^2} \geqslant t\right] \mathrm{d}t \\
&= \int_1^\infty \mathbb{P}_{S\sim D^n}\left[\Delta \geqslant \sqrt{\frac{\ln t}{n}}\right] \mathrm{d}t + 1 \\
&\leqslant \int_1^\infty 2e^{-2\ln t}\mathrm{d}t + 1 \\
&= 3
\end{aligned}$$

By applying Markov's Inequality, we get

$$\begin{aligned}
\mathbb{P}_{S\sim D^n}\left[\mathbb{E}_{h\sim\mathcal{P}}\left[e^{n(\mathrm{Err}_D(h)-\mathrm{Err}_S(h))^2}\right] \geqslant 3/\delta\right] &= \mathbb{P}_{S\sim D^n}\left[\mathbb{E}_{h\sim\mathcal{P}}\left[e^{n\Delta^2}\right] \geqslant 3/\delta\right] \\
&\leqslant \frac{\mathbb{E}_{S\sim D^n}\left[\mathbb{E}_{h\sim P}\left[e^{n\Delta^2}\right]\right]}{3/\delta} \\
&= \frac{\mathbb{E}_{h\sim P}\left[\mathbb{E}_{S\sim D^n}\left[e^{n\Delta^2}\right]\right]}{3/\delta} \quad \text{(Fubini Theorem)} \\
&\leqslant \frac{\mathbb{E}_{h\sim P}[3]}{3/\delta} = \delta
\end{aligned}$$

■

Now we can prove PAC-Bayesian Theorem (6.1) by applying Jensen's Inequality.
With probability $1-\delta$,

$$\begin{aligned}
(\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h) - \mathrm{Err}_S(h)])^2 &\leqslant \frac{1}{n}\mathbb{E}_{h\sim\mathcal{Q}}\, n[\mathrm{Err}_D(h) - \mathrm{Err}_S(h)]^2 \\
&\leqslant \frac{1}{n}\left(\ln \mathbb{E}_{h'\sim\mathcal{P}}\left[e^{n\Delta^2}\right] + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})\right) \\
&\leqslant \frac{1}{n}\left(\ln(3/\delta) + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})\right)
\end{aligned}$$

Thus,

$$\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)] \leqslant \mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_S(h)] + \sqrt{\frac{D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) + \ln(3/\delta)}{n}}$$

Lemma 6.4 *For any fixed $h \sim \mathcal{P}$, we have*

$$\mathbb{E}_{S\sim D^n}\left[e^{nD_{\mathrm{B}}(\mathrm{Err}_S(h)\|\,\mathrm{Err}_D(h))}\right] \leqslant n+1 \tag{6.5}$$

Proof: For simplicity, we denote $\mathrm{Err}_D(h)$ by p, $\mathrm{Err}_S(h)$ by $\hat{p}_S$. Since $\mathcal{P}$ is the prior, h is independent of S. Therefore,

$$\begin{aligned}\mathbb{E}_{S\sim D^n}\left[e^{nD_{\mathrm{B}}(\mathrm{Err}_S(h)\|\,\mathrm{Err}_D(h))}\right] &= \mathbb{E}_{S\sim D^n}\left[e^{nD_{\mathrm{B}}(\hat{p}_S\|p)}\right]\\ &= \mathbb{E}_{S\sim D^n}\left[\left(\frac{\hat{p}_S}{p}\right)^{n\hat{p}_S}\left(\frac{1-\hat{p}_S}{1-p}\right)^{n(1-\hat{p}_S)}\right]\end{aligned}$$

Note that $S\sim D^n$ implies $\sum_{i=1}^n I[y_i\neq h(x_i)]\sim B(n,p)$, so the equation above can be rewritten as:

$$\begin{aligned}\mathbb{E}_{S\sim D^n}\left[\left(\frac{\hat{p}_S}{p}\right)^{n\hat{p}_S}\left(\frac{1-\hat{p}_S}{1-p}\right)^{n(1-\hat{p}_S)}\right] &= \sum_{k=0}^n\binom{n}{k}p^k(1-p)^{n-k}\left(\frac{k/n}{p}\right)^{n(k/n)}\left(\frac{1-k/n}{1-p}\right)^{n(1-k/n)}\\ &= \sum_{k=0}^n\binom{n}{k}\left(\frac{k}{n}\right)^k\left(1-\frac{k}{n}\right)^{n-k}\\ &\leqslant \sum_{k=0}^n\left(\binom{n}{0}\left(\frac{k}{n}\right)^0\left(1-\frac{k}{n}\right)^{n-0}+\binom{n}{1}\left(\frac{k}{n}\right)^1\left(1-\frac{k}{n}\right)^{n-1}+\cdots\right.\\ &\quad\left.+\binom{n}{k}\left(\frac{k}{n}\right)^k\left(1-\frac{k}{n}\right)^{n-k}+\cdots+\binom{n}{n}\left(\frac{k}{n}\right)^n\left(1-\frac{k}{n}\right)^{n-n}\right)\\ &= \sum_{k=0}^n\sum_{r=0}^n\binom{n}{r}\left(\frac{k}{n}\right)^r\left(1-\frac{k}{n}\right)^{n-r}\\ &= \sum_{k=0}^n\left(\frac{k}{n}+\left(1-\frac{k}{n}\right)\right)^n\\ &= n+1\end{aligned}$$

■

Using Lemma 6.4, we can get a better result:

Theorem 6.5 *With probability* $1-\delta$,

$$D_{\mathrm{B}}\left(\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_S(h)]\|\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)]\right)\leqslant\frac{1}{n}\left(D_{\mathrm{B}}(\mathcal{Q}\|\mathcal{P})+\ln\left(\frac{n+1}{\delta}\right)\right)\tag{6.6}$$

Proof: For fixed $h'\sim\mathcal{P}$, by applying Markov Inequality and Lemma 6.4, we have

$$\begin{aligned}\mathbb{P}_{S\sim D^n}\left[\mathbb{E}_{h'\sim\mathcal{P}}\left[e^{nD_{\mathrm{B}}(\mathrm{Err}_S(h')\|\,\mathrm{Err}_D(h'))}\right]\geqslant\frac{n+1}{\delta}\right] &\leqslant \frac{\mathbb{E}_{h'\sim\mathcal{P}}\left[\mathbb{E}_{S\sim D^n}\left[e^{nD_{\mathrm{B}}(\mathrm{Err}_S(h')\|\,\mathrm{Err}_D(h'))}\right]\right]}{\frac{n+1}{\delta}}\\ &\leqslant \frac{n+1}{\frac{n+1}{\delta}}=\delta\end{aligned}$$

Since $D_{\mathrm{B}}(\cdot\|\cdot)$ is convex, by Lemma 6.2 and Jensen's Inequality, with probability $1-\delta$,

$$\begin{aligned} D_{\mathrm{B}}(\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_S(h)]\|\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)]) &\leqslant \mathbb{E}_{h\sim\mathcal{Q}}[D_{\mathrm{B}}(\mathrm{Err}_S(h)\|\,\mathrm{Err}_D(h))] \\ &= \frac{1}{n}\mathbb{E}_{h\sim\mathcal{Q}}\left[nD_{\mathrm{B}}(\mathrm{Err}_S(h)\|\,\mathrm{Err}_D(h))\right] \\ &\leqslant \frac{1}{n}\left(\ln\left(\mathbb{E}_{h'\sim\mathcal{P}}\left[e^{nD_{\mathrm{B}}(\mathrm{Err}_S(h')\|\,\mathrm{Err}_D(h'))}\right]\right) + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})\right) \\ &\leqslant \frac{1}{n}\left(\ln\left(\frac{n+1}{\delta}\right) + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})\right) \end{aligned}$$

Thus proving the claim. ■

7.4 PAC-Bayesian Bound for SVM

Now we will apply PAC-Bayesian Theory to linear classifiers. Suppose $\mathcal{Q}$ is a distribution over classifiers, let us define a deterministic voting classifier $g_{\mathcal{Q}}(\boldsymbol{x})$

$$g_{\mathcal{Q}}(\boldsymbol{x}) = \mathrm{sgn}(\mathbb{E}_{h\sim\mathcal{Q}}h(\boldsymbol{x})) \tag{6.7}$$

We have the following proposition:

Proposition 6.6

$$\mathrm{Err}_D(g_{\mathcal{Q}}) \leqslant 2\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)] \tag{6.8}$$

Proof: For each point $(\boldsymbol{x}, y)\sim D$, $g_{\mathcal{Q}}(\boldsymbol{x}) \neq y$ implies that at least half of hs in $\mathcal{Q}$ satisfy $h(\boldsymbol{x}) \neq y$, which leads to the conclusion that $\mathrm{Err}_D(g_{\mathcal{Q}}) \leqslant 2\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)]$.

■

Now we consider linear classifiers $h(\boldsymbol{x}) = \mathrm{sgn}(\boldsymbol{w}^\top\boldsymbol{x} + b)$, $\boldsymbol{x}\in\mathbb{R}^d, y\in\{\pm 1\}, \boldsymbol{w}\in\mathbb{R}^d, b\in\mathbb{R}$.
If we assume $\mathcal{P} = U(S^{d-1})$, the spherical integral of $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})$ would be complicated to compute. Alternatively, we assume $\mathcal{P} = \mathcal{N}(\boldsymbol{0}, I_d), \mathcal{Q} = \mathcal{N}(\mu\boldsymbol{w}, I_d)$, here $\|\boldsymbol{w}\|_2 = 1$ and μ is a scale factor. According to PAC-Bayesian Theorem (6.1), we have

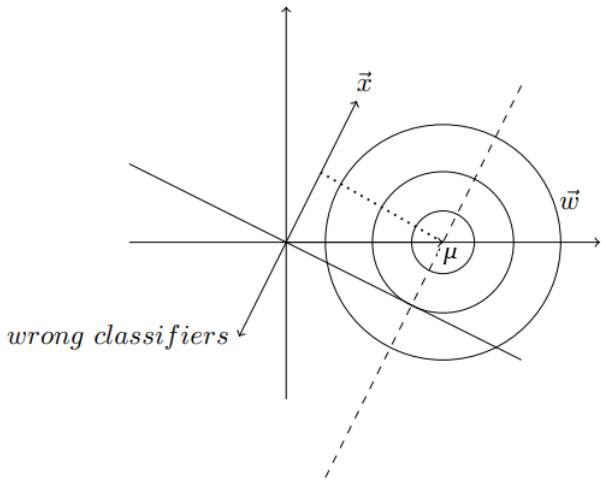
$$\mathrm{Err}_D(g_{\mathcal{Q}}) \leqslant 2\left[\mathrm{Err}_S(\mathcal{N}(\mu\boldsymbol{w}, I_d)) + \sqrt{\frac{D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) + \ln(3/\delta)}{n}}\right] \tag{6.9}$$

It suffices to determine $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})$ and $\mathrm{Err}_S(\mathcal{N}(\mu\boldsymbol{w}, I_d))$ respectively.

6.4.1 Determine $D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P})$

$$\begin{aligned}
D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) &= \int_{\mathbb{R}^d} \frac{1}{\left(\sqrt{2\pi}\right)^d} \exp\left[-\frac{1}{2}\|\boldsymbol{x}-\mu\boldsymbol{w}\|^2\right] \frac{1}{2}\left(\|\boldsymbol{x}\|^2 - \|\boldsymbol{x}-\mu\boldsymbol{w}\|^2\right) \mathrm{d}\boldsymbol{x} \\
&= \int_{\mathbb{R}^d} \frac{1}{\left(\sqrt{2\pi}\right)^d} \exp\left[-\frac{1}{2}\|\boldsymbol{x}-\mu\boldsymbol{w}\|^2\right] \left(\mu\boldsymbol{w}^\top\boldsymbol{x} - \frac{1}{2}\mu^2\right) \mathrm{d}\boldsymbol{x} \\
&= -\frac{1}{2}\mu^2 + \int_{\mathbb{R}^d} \frac{1}{\left(\sqrt{2\pi}\right)^d} \exp\left[-\frac{1}{2}\|\boldsymbol{x}-\mu\boldsymbol{w}\|^2\right] \left(\mu\boldsymbol{w}^\top(\boldsymbol{x}-\mu\boldsymbol{w}) + \mu^2\right) \mathrm{d}\boldsymbol{x} \\
&= \frac{1}{2}\mu^2 + \mu \int_{\mathbb{R}^d} (\boldsymbol{w}^\top\boldsymbol{x}) \frac{1}{\left(\sqrt{2\pi}\right)^d} \exp\left[-\frac{1}{2}\|\boldsymbol{x}\|^2\right] \mathrm{d}\boldsymbol{x} \\
&= \frac{1}{2}\mu^2 + \mu\boldsymbol{w}^\top \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}[\boldsymbol{x}] \\
&= \frac{1}{2}\mu^2
\end{aligned}$$

6.4.2 Determine $\mathrm{Err}_S(\mathcal{N}(\mu\boldsymbol{w}, I_d))$



For a fixed point $(\boldsymbol{x}, y)$ and a classifier $\boldsymbol{h} \sim \mathcal{N}(\mu\boldsymbol{w}, I_d)$, with the intuition from the figure above, we have

$$\frac{y\boldsymbol{x}^\top\boldsymbol{h}}{\|\boldsymbol{x}\|} \sim \mathcal{N}\left(\frac{y\boldsymbol{x}^\top(\mu\boldsymbol{w})}{\|\boldsymbol{x}\|}, \frac{y^2\boldsymbol{x}^\top\boldsymbol{x}}{\|\boldsymbol{x}\|^2}\right) = \mathcal{N}\left(\frac{y\mu(\boldsymbol{w}^\top\boldsymbol{x})}{\|\boldsymbol{x}\|}, 1\right)$$

So

$$\begin{aligned}
\mathrm{Err}_S(\mathcal{N}(\mu\boldsymbol{w}, I_d)) &= \mathbb{P}_{\boldsymbol{h}\sim\mathcal{N}(\mu\boldsymbol{w}, I_d)}\left[\frac{y\left(\boldsymbol{x}^\top\boldsymbol{h}\right)}{\|\boldsymbol{x}\|} < 0\right] \\
&= \bar{\Phi}\left(\frac{y\mu(\boldsymbol{w}^\top\boldsymbol{x})}{\|\boldsymbol{x}\|}\right)
\end{aligned}$$

Here, $\Phi(t) = \int_{-\infty}^{t} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \mathrm{d}x, \bar{\Phi}(t) = 1 - \Phi(t)$.

6.4.3 Putting them together

By PAC-Bayesian Theorem (6.1), for SVM model

$$\begin{aligned}\mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_D(h)] &\leqslant \mathbb{E}_{h\sim\mathcal{Q}}[\mathrm{Err}_S(h)] + \sqrt{\frac{D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) + \ln\frac{3}{\delta}}{n}} \\ &= \mathbb{E}_{S\sim D^n}\left[\bar{\Phi}\left(\frac{y\mu\left(\boldsymbol{w}^\top\boldsymbol{x}\right)}{\|\boldsymbol{x}\|}\right)\right] + \sqrt{\frac{\frac{\mu^2}{2} + \ln\frac{3}{\delta}}{n}}\end{aligned}$$

Therefore, with probability at least $1-\delta$ over the random draw of n training data, for all μ and $\boldsymbol{w}$, we have

$$\mathrm{Err}_D(g_{\mathcal{Q}}) \leqslant 2\left[\mathbb{E}_{S\sim D^n}\left[\bar{\Phi}\left(\frac{y\mu\left(\boldsymbol{w}^\top\boldsymbol{x}\right)}{\|\boldsymbol{x}\|}\right)\right] + \sqrt{\frac{\frac{\mu^2}{2} + \ln\frac{3}{\delta}}{n}}\right]$$

By optimizing the value of μ, we have

$$\mathrm{Err}_D(g_{\mathcal{Q}}) \leqslant 2\inf_{\mu>0}\left\{\left[\mathbb{E}_{S\sim D^n}\left[\bar{\Phi}\left(\frac{y\mu\left(\boldsymbol{w}^\top\boldsymbol{x}\right)}{\|\boldsymbol{x}\|}\right)\right] + \sqrt{\frac{\frac{\mu^2}{2} + \ln\frac{3}{\delta}}{n}}\right]\right\}$$

Since the Gaussian tail probability is monotonically decreasing w.r.t. μ and $\mu^2/2n$ is monotonically increasing w.r.t. μ, we cannot determine the monotonicity of the bound. Intuitively, the Gaussian tail probability will be small if the margin is large, thus the larger the margin is, the tighter the bound will be, which means the generalization ability of SVM will be good when it has a large margin.

References

[1] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48, pages 1225–1234, 2016.

[2] Samuel Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 9058–9067, 2020.

[3] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.

Machine Learning **Spring 2021**

Lecture 8: Algorithm Stability and Generalization, Clustering

Lecturer: Liwei Wang *Scribe: Jianing Lou, Xuanyu Peng, Yuedi Chen, Zhao Zhang*

8.1 Term Project 3

Deep learning has a large number of parameters, which often exceed the number of data. Therefore, the generalization of deep learning is a problem worth studying. Read the paper *Uniform Convergence May be unable to Explain the Generalization in deep learning*. If agree with the article, trying to construct general case; Give reasons for disagreeing.

8.2 Algorithmic Stability and Generalization

Definition 8.1 (Uniform Stability) *Let $\mathcal{A}$ be a learning algorithm. $S = (z_1, \cdots, z_n)$ be a training dataset. Let $S^i = (z_1, \cdots, z_{i-1}, z_i', z_{i+1}, \cdots, z_n)$ denote a neighboring dataset. Let $\mathcal{A}(S)$ denote a classifier learned by $\mathcal{A}$ from S. Let $\ell(\cdot, \cdot)$ be a loss function.*

A learning algorithm $\mathcal{A}$ is said to have uniform stability β with respect to loss $\ell(\cdot, \cdot)$, if $\forall S, \forall i, \forall S^i, \forall z$,

$$|\ell(\mathcal{A}(S), z) - \ell(\mathcal{A}(S^i), z)| \leq \beta$$

Theorem 8.2 (Uniform stability implies generalization) *Define the risk (similar to test error) as follows,*

$$R(\mathcal{A}(S)) = \mathbb{E}_{z \sim D}[\ell(\mathcal{A}(S), z)]$$

And define the empirical risk (similar to training error) as follows,

$$R_{emp}(\mathcal{A}(S)) = \frac{1}{n} \sum_{i=1}^{n} \ell(\mathcal{A}(S), z_i)$$

Then assume $|\ell(\cdot, \cdot)| \leq M$, we have

$$\mathbb{P}(R(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S)) \geq \beta + \epsilon) \leq \exp\left(\frac{-n\epsilon^2}{2(n\beta + M)^2}\right)$$

The proof of 8.2 is based on the following lemmas.

Lemma 8.3 *Suppose $\mathcal{A}$ is symmetric with respect to $(z_1, \cdots, z_n)$,* i.e. *for any permutation σ, $\mathcal{A}(z_1, \cdots, z_n) = \mathcal{A}(\sigma(z_1, \cdots, z_n))$, then*

$$\mathbb{E}_S[R(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S))] \leq \beta \tag{8.1}$$

Proof: *On the one hand,*

$$\begin{aligned}\mathbb{E}_S[R_{emp}(\mathcal{A}(S))] &= \mathbb{E}_S[\frac{1}{n}\sum_{i=1}^{n} l(\mathcal{A}(S), z_i)] \\ &= \mathbb{E}_S[l(\mathcal{A}(S), z_1)]\end{aligned}$$

That is because $l(\mathcal{A}(z_1, \cdots, z_i, \cdots, z_n), z_i) = l(\mathcal{A}(z_i, \cdots, z_1, \cdots, z_n), z_1) = l(\mathcal{A}(S), z_1)$*, according to the symmetry of* $\mathcal{A}$*.*
On the other hand,

$$\begin{aligned}\mathbb{E}_S[R(\mathcal{A}(S))] &= \mathbb{E}_S\mathbb{E}_z[l(\mathcal{A}(S), z)] \\ &= \mathbb{E}_{z_1,\cdots,z_n,z}[l(\mathcal{A}(S), z)]\end{aligned}$$

That means the expected loss on the random data $z_1, \cdots, z_n, z$*. Switch* z *and* z_1*, we have*

$$\mathbb{E}_S[R(\mathcal{A}(S))] = \mathbb{E}_S[l(\mathcal{A}(S'), z_1)]$$

where S' *denotes* $(z, z_2, \cdots, z_n)$*.*
According to the definition of β*,*

$$\begin{aligned}\mathbb{E}_S[R(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S))] &= \mathbb{E}_S[l(\mathcal{A}(S), z_1) - l(\mathcal{A}(S'), z_1)] \\ &\leq \beta\end{aligned}$$

■

Lemma 8.4 (McDiarmid's Inequality) *Suppose* $|f(x_1, \cdots, x_n) - f(x_1, \cdots, x_i', \cdots, x_n)| \leq c_i, \forall i \in [n], \forall x_1, \cdots x_i, x_i', \cdots, x_n$*. Then*

$$\mathbb{P}(f(X_1, \cdots, X_n) - \mathbb{E}f(X_1, \cdots, X_n) \geq \epsilon) \leq \exp\{-\frac{2\epsilon^2}{\sum c_i^2}\} \tag{8.2}$$

Lemma 8.5 *Assume* $|l(\cdot, \cdot)| \leq M$*,*

$$\left|[R(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S))] - [R(\mathcal{A}(S^i)) - R_{emp}(\mathcal{A}(S^i))]\right| \leq 2(\beta + \frac{M}{n}) \tag{8.3}$$

Proof:

$$\begin{aligned}& \left|[R(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S))] - [R(\mathcal{A}(S^i)) - R_{emp}(\mathcal{A}(S^i))]\right| \\ \leq\ & \left|R_{emp}(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S^1))\right| + \left|R(\mathcal{A}(S)) - R(\mathcal{A}(S^1))\right| \\ \leq\ & \frac{1}{n}\left|l(\mathcal{A}(S), z_1) - l(\mathcal{A}(S^1), z_1')\right| + \\ & \frac{1}{n}\sum_{i=2}^{n}\left|l(\mathcal{A}(S), z_i) - l(\mathcal{A}(S^1), z_i)\right| + \\ & \mathbb{E}_z[l(\mathcal{A}(S), z) - l(\mathcal{A}(S^1), z)] \\ \leq\ & \frac{1}{n}(\left|l(\mathcal{A}(S), z_1) - l(\mathcal{A}(S^1), z_1)\right| + \left|l(\mathcal{A}(S^1), z_1)\right| + \left|l(\mathcal{A}(S^1), z_1')\right|) + \frac{n-1}{n}\beta + \beta \\ \leq\ & 2(\beta + \frac{M}{n})\end{aligned}$$

■

Though loss functions are usually unbounded, 8.3 still holds. That's because the proof only uses $\left|l(\mathcal{A}(S^1), z_1)\right| \leq M$ and $\left|l(\mathcal{A}(S^1), z_1')\right| \leq M$, which are ensured by the bounded data.

Finally, we conclude the proof of Theorem 8.2:

Proof: Denote $\Phi(S) = R(\mathcal{A}(S)) - R_{emp}(\mathcal{A}(S))$. According to Lemma 8.3, we have

$$\mathbb{P}(\Phi(S) \geq \beta + \epsilon) \leq \mathbb{P}(\Phi(S) - \mathbb{E}_S[\Phi(S))] \geq \epsilon)$$

Lemma 8.5 means that $\Phi(S)$ is a stable function, where $c_i = 2(\beta + \frac{M}{n})$, then we can used McDiarmid's Inequality to get the result,

$$\mathbb{P}(\Phi(S) - \mathbb{E}_S[\Phi(S))] \geq \epsilon) \leq \exp\left(\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right) = \exp\left(\frac{-n\epsilon^2}{2(n\beta + M)^2}\right)$$

■

8.3 Clustering

Clustering is an unsupervised learning task, and is described as follows:

Given a set of datas $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ and a non-zero integer $k \leq n$, where each data is a d-dimensional real vector, clustering(k-means clustering) aims to partition these n datas into k sets $S_1, S_2, \ldots, S_k$ so as to minimize the following loss function

$$\phi = \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} ||\boldsymbol{x} - \boldsymbol{\mu}_i||^2$$

where $\boldsymbol{\mu}_i$ is the cluster center of S_i.

The most common algorithm is "k-means algorithm".

Algorithm 8.3.1: k-means algorithm

```
Initialize: choose k points randomly as the cluster centers m_1, ..., m_k;
do
    Assign each data to the cluster center with the nearest mean;
    S_i ← {x_j : x_j is assigned to m_i}, ∀i;
    m_i ← the mean of points in S_i, ∀i;
while k cluster centers changes;
return m_1, ..., m_k;
```

However, this naive algorithm is only guaranteed to find a local optimum.

Improvement: k**-means++**

We can optimize the "initialize" step in line 1 as follows:

Algorithm 8.3.2: Improved initialization

1 Choose one center uniformly at random among the data points;
2 **for** $i : 2 \to k$ **do**
3 | Choose one new data point at random as a new center, a point $\boldsymbol{x}$ is chosen with probability proportional to $||\boldsymbol{x} - m^*||^2$, where $m^* \in \{m_1, \ldots, m_{i-1}\}$ and is nearest to $\boldsymbol{x}$.
4 **end**

Letting ϕ_{OPT} denote the global optimal loss, it has been proved by Arthur and Vassilvitskii[1] that after choosing centers in this way, we have

$$\mathbb{E}[\phi] \leq 8(\ln k + 2)\phi_{OPT}$$

References

[1] Arthur, D.; Vassilvitskii, S. (2007). "k-means++: the advantages of careful seeding". *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms.* Society for Industrial and Applied Mathematics Philadelphia, PA, USA. pp. 1027–1035.

Machine Learning **Spring 2021**

Lecture 9: Dimensionality Reduction, Online Learning

Lecturer: Liwei Wang *Scribe: Shipeng Cen, Chenqi Zhao, Xuheng Li, Kaiyun Tan*

9.1 Dimensionality Reduction

Assume we have N points $x_1, x_2, ..., x_N$ in R^n, and we want to map these points to a space with lower dimension, for example R^d, where $d \ll n$. An intuition is to map these points to the "dim" in which the data share high variance and the loss is related to the projection distance. And to minimize the distance loss, we will use Principle Component Analysis method, as an traditional use of SVD. If we put all the points into a matrix A and use SVD on A, we will get $A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$, then we save the d terms with the largest singular value in the above sum equation, surely the we can also get the form by solving the eigenvalues and eigenvectors of AA^T or A^TA. It's not difficult, and details will not be repeated.

Here we introduce another method to realize dimensionality reduction.

9.1.1 Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss lemma is a fundamental result in dimensionality reduction that states that any m points in high-dimensional space can be mapped to a much lower dimension, $k \geq O(\frac{\log m}{\epsilon^2})$, without distorting pairwise distance between any two points by more than a factor of $(1 \pm \epsilon)$.

To begin with, the proof of this theorem is an existential proof, not a constructive one.

Lemma 9.1 Let Q be a random variable following a χ^2-squared distribution with k degrees of freedom. Then we have following inequality:

$$P((1-\epsilon)k \leq Q \leq (1+\epsilon)k) \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}$$

Proof:

$$P(Q \geq (1+\epsilon)k) = P(\exp(tQ) \geq \exp((1+\epsilon)tk)), \quad t > 0 \tag{9.1}$$

According to Markov's inequality, we have:

$$P(\exp(tQ) \geq \exp((1+\epsilon)tk)) \leq \frac{E(\exp(tQ))}{\exp((1+\epsilon)tk)))} = \frac{(1-2t)^{-\frac{k}{2}}}{\exp((1+\epsilon)tk)))} \tag{9.2}$$

The equation above we use is the moment-generating function of a χ^2-squared distribution with k degrees of freedom. According to $\frac{\partial f(t)}{\partial t} = 0$,we choose $t = \frac{\epsilon}{2(1+\epsilon)}$ to minimize the RHS.And we get that:

$$P(Q \geq (1+\epsilon)k) \leq \left(\frac{1+\epsilon}{e^\epsilon}\right)^{\frac{k}{2}} \tag{9.3}$$

Inspired by Taylor's formula, we have:

$$1 + \epsilon \leq e^{\epsilon - \frac{\epsilon^2}{2} + \frac{\epsilon^3}{2}} \tag{9.4}$$

So the above inequality comes into the following form:

$$P(Q \geq (1+\epsilon)k) \leq e^{-\frac{k\epsilon^2(1-\epsilon)}{4}} \tag{9.5}$$

The statement of the lemma follows by using similar techniques to bound another situation and by applying the union bound. ■

Lemma 9.2 *Let $x \in \mathbb{R}^N$, define $k \leq N$ and assume that entries in $A \in \mathbb{R}^{k \times N}$ are sampled independently from $N(0,1)$. Then for any $0 \leq \epsilon \leq \frac{1}{2}$, we have following inequality:*

$$P\left[(1-\epsilon)||x||^2 \leq ||\frac{1}{\sqrt{k}}Ax||^2 \leq (1+\epsilon))||x||^2\right] \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}$$

Proof: we notice that, $\frac{||Ax||^2}{||x||^2} = \sum_{i=1}^k T_i^2$, where $T_i \sim N(0,1)$, so we know that $\sum_{i=1}^k T_i^2$ obeys χ^2-squared distribution with k degrees of freedom.

$$P\left[(1-\epsilon)||x||^2 \leq ||\frac{1}{\sqrt{k}}Ax||^2 \leq (1+\epsilon))||x||^2\right] = P\left[(1-\epsilon)k \leq \sum_{i=1}^k T_i^2 \leq (1+\epsilon)k\right] \tag{9.6}$$

Then we use Lemma 9.1 and the proof is completed. ■

Now we can prove JL lemma.

Lemma 9.3 For any tolerance $\epsilon \in$ (0,1) and any $m > 4$, $k > \frac{8logm}{\epsilon^2(1-\epsilon)}$, there exists a map f:$R^N \to R^k$,such that for any u,v $\in$ set V of m points in R^N, we have:

$$(1-\epsilon)||u-v||^2 \leq ||f(u)-f(v)||^2 \leq (1+\epsilon)||u-v||^2 \tag{9.7}$$

Proof: We choose $f = \frac{1}{\sqrt{k}}$. According to Lemma 9.2, we have $P[(1-\epsilon)||x||^2 \leq ||\frac{1}{\sqrt{k}}Ax||^2 \leq (1+\epsilon))||x||^2] \geq 1 - 2e^{-(\epsilon^2-\epsilon^3)k/4}$.
According to union bound, we have:

$$P[\text{fail}] \leq \binom{m}{2} * 2e^{-(\epsilon^2-\epsilon^3)k/4} \tag{9.8}$$

Notice $\binom{m}{2} \leq m^2$, Finally we get following inequality to guarantee the existence of the mapping:

$$P[\text{fail}] \leq m^2 e^{-(\epsilon^2-\epsilon^3)k/4} < 1 \tag{9.9}$$

which is equivalent to following inequality:

$$k > \frac{8\log m}{\epsilon^2(1-\epsilon)} \tag{9.10}$$

■

9.2 Online Learning

Online machine learning is a method of machine learning in which data becomes available in a sequential order and is used to update the best predictor for future data at each step.[1] It differs from our previous methods in that: first, online learning mixes training and test phases; second, with online learning, *no distributional assumption* is made and thus there is no notion of generalization. Instead, the performance of online learning algorithms is measured using a *mistake model* and the notion of *regret*. To derive guarantees in this model, theoretical analyses are based on a worst-case or adversarial assumption.

Let's begin with a traditional setting of online learning:

9.2.1 Prediction with expert advice

Assume that there are T rounds and N experts. At the tth round, every expert $i \in [N]$ makes a prediction, denoted as $\widetilde{y_{t,i}} \in \{0,1\}$, then the learner makes prediction $\widetilde{y}_t$. After that, the adversary reveals $y_t \in \{0,1\}$. The objective is to minimize the accumulate loss (here it is 0-1loss) $\sum_{t=1}^{T} I[\widetilde{y}_t \neq y_t]$.

Our simple intuition might lead us to the "follow the leader" strategy, which means at the tth round, we make the same prediction as best expert, that is, the expert who makes the smallest number of mistakes in the former $t-1$ rounds. However, this strategy gives poor performance both theoretically and practically.

9.2.1.1 Weighted Majority Algorithm

In this section we will introduce a boosting-like algorithm, the Weighted Majority (WM) algorithm, that weights the importance of experts and applies Multiplicative Weight Updating to reduce the weight of incorrect experts at each round.

Algorithm 9.2.1: Weighted Majority Algorithm

Initialize: $w_{1,i} = 1, \forall i \in [N]$;
Parameter: $\beta \in (0,1)$;
for *t=1,2,...,T* **do**
  Learner makes weighted majority vote: $\widetilde{y}_t = \begin{cases} 0 & if \sum_{\widetilde{y_{t,i}}=0} w_{t,i} > \sum_{\widetilde{y_{t,i}}=1} w_{t,i} \\ 1 & otherwise. \end{cases}$;
  if $\widetilde{y}_t = y_t$ **then**
    $w_{t+1,i} \leftarrow w_{t,i} \quad \forall i \in [N]$;
  else
    $w_{t+1,i} \leftarrow \begin{cases} \beta w_{t,i} & \widetilde{y_{t,i}} \neq y_t \\ w_{t,i} & \widetilde{y_{t,i}} = y_t \end{cases} \quad \forall i \in [N]$;
  end
end

The following theorem presents a mistake bound of WM algorithm after $T \geq 1$ rounds as a function of the number of mistakes made by the best expert.

Theorem 9.4 *For $\beta \in (0,1)$, after all T rounds, define the loss of learner as $L_T = \sum_{t=1}^{T} I[\widetilde{y}_t \neq y_t]$, define the*

loss of ith expert as $m_{T,i} = \sum_{t=1}^{T} I[\widetilde{y_{t,i}} \neq y_t], \forall i \in [N]$, *define the loss of best expert as* $m_T^* = \min_{i\in[N]} m_{T,i}$. *We have*

$$L_T \leq \frac{m_T^* \log 1/\beta + \log N}{\log(2/(1+\beta))}$$

Proof: To prove this theorem, we first introduce a *potential function*. For any $t \geq 1$, define the potential function as $W_t = \sum_{i=1}^{N} w_{t,i}$. On the one hand, since predictions are generated using weighted majority vote, if the algorithm makes an error at round t, this implies that

$$W_{t+1} \leq \left(\frac{1}{2} + \frac{1}{2}\beta\right) W_t = \frac{1+\beta}{2} W_t \tag{9.11}$$

Since $W_1 = N$ and L_T mistakes are made after T rounds, we thus have the following upper bound:

$$W_{T+1} \leq \left(\frac{1+\beta}{2}\right)^{L_T} N \tag{9.12}$$

On the other hand, since all weights are non-negative, we have $\forall i \in [N]$

$$W_{T+1} \geq w_{T+1,i} = \beta^{m_{T,i}} \tag{9.13}$$

Applying this lower bound to the best expert and combining it with the upper bound in (9.12) gives us:

$$\begin{aligned}
&\beta^{m_T^*} \leq (\frac{1+\beta}{2})^{L_T} N \\
\Rightarrow &m_T^* \log\beta \leq \log N + L_T \log(\frac{1+\beta}{2}) \\
\Rightarrow &L_T \log(\frac{2}{1+\beta}) \leq m_T^* \log\frac{1}{\beta} + \log N
\end{aligned}$$

■

Thus, the theorem guarantees a bound of the following form for WM algorithm:

$$L_T \leq \text{constant} \times [\text{mistakes of best expert}] + O(\log N)$$

Note that as $\beta \to 1$, the constant $\to 2$ (according to L'Hospital's rule).

9.2.1.2 Randomized Weight Updating

We first introduce another algorithm, which is different from the former one mainly in that the voting process is replace with randomized selection of expert.

Algorithm 9.2.2: Randomized weight updating

```
Initialize: w_{1,i} = 1, i ∈ [N].
Parameter: β ∈ [1/2, 1).
for t = 1, ..., T do
    Learner chooses i_t ∈ [N] with probability w_{t,i_t} / Σ_j w_{t,j} and ỹ_t ← ỹ_{t,i_t}.
    Update w_{t+1,i} ← β w_{t,i} for all i such that ỹ_{t,i} ≠ y_t.
end
```

The expected loss of the learner at round t is

$$l_t = \frac{\sum_i w_{t,i}|\tilde{y}_{t,i} - y_t|}{\sum_j w_{t,j}}, \tag{9.14}$$

and the total expected loss is

$$L_T = \sum_{t=1}^{T} l_t. \tag{9.15}$$

For Algorithm 2, we have the following theorem:

Theorem 9.5 *For $\beta \in [1/2, 1)$, the total expected loss is bounded by*

$$L_T \leq (2-\beta)m_T^* + \frac{\log N}{1-\beta}. \tag{9.16}$$

Proof: Consider again the potential function

$$W_t = \sum_i w_{i,t}. \tag{9.17}$$

Note that

$$\begin{aligned} W_{t+1} &= \sum_{\tilde{y}_{t,i}=y_t} w_{t,i} + \beta \sum_{\tilde{y}_{t,i}\neq y_t} w_{t,i} = \sum_i w_{t,i} + (\beta-1)\sum_{\tilde{y}_{t,i}\neq y_t} w_{t,i} \\ &= W_t + (\beta-1)\sum_i w_{t,i}|\tilde{y}_{t,i} - y_t| = W_t + (\beta-1)W_t l_t = W_t[1 + (\beta-1)l_t]. \end{aligned}$$

So we have

$$W_{T+1} = W_1 \prod_{t=1}^{T}[1 + (\beta-1)l_t] = N\prod_{t=1}^{T}[1 + (\beta-1)l_t].$$

On the other hand,

$$W_{T+1} \geq \max_i w_{T,i} = \beta^{m_T^*},$$

so

$$N\prod_{t=1}^{T}[1 + (\beta-1)l_t] \geq \beta^{m_T^*},$$

which is

$$\sum_{t=1}^{T} \log(1+(\beta-1)l_t) \geq m_T^* \log\beta - \log N.$$

Note that

$$\log(1+(\beta-1)l_t) \leq (\beta-1)l_t,$$

so

$$m_T^* \log\beta - \log N \leq \sum_{t=1}^{T}(\beta-1)l_t = (\beta-1)L_T.$$

Thus we have a bound for L_T:

$$L_T \leq \frac{\log N}{1-\beta} - \frac{\log\beta}{1-\beta} m_T^*.$$

It suffices to show that

$$-\log\beta \leq (1-\beta)(2-\beta) \tag{9.18}$$

for $\beta \in [1/2, 1)$. Let

$$f(\beta) = \log\beta + (1-\beta)(2-\beta),$$

then

$$f'(\beta) = \frac{1}{\beta} - 3 + 2\beta = \frac{(1-\beta)(1-2\beta)}{\beta} \leq 0.$$

So $f(\beta) \geq f(1) = 0$, and (9.18) is proven. ■

For Algorithm 2, the coefficient before m_T^* is improved to $2-\beta$, strictly smaller than 2.

If

$$\beta = 1 - \sqrt{\frac{\log N}{T}} \geq 1/2,$$

combined with $m_T^* \leq T$, we have

$$L_T \leq m_T^* + \sqrt{\frac{\log N}{T}} m_T^* + \sqrt{T\log N} \leq m_T^* + \sqrt{T\log N} + \sqrt{T\log N} = m_T^* + 2\sqrt{T\log N},$$

and the average loss per round is bounded by

$$\frac{L_T}{T} \leq \frac{m_T^*}{T} + 2\sqrt{\frac{\log N}{T}}.$$

9.2.2 Proof of Von Neumann's Minimax Theorem

9.2.2.1 Preliminary

We first introduce Hedge algorithm, a general case of randomized weight updating. Instead of 0-1 loss (i.e. whether an expert gives the right prediction), this time we meet the loss function $g_t(i) \in [0,1]$, which indicates the loss we undertake if we follow expert i at round t.

Algorithm 9.2.3: Hedge

```
Initialize: w_{1,i} = 1, i ∈ [N].
Parameter: β ∈ (0, 1).
for t = 1, ..., T do
    Learner chooses i_t ∈ [N] with probability w_{t,i_t} / Σ_j w_{t,j} and incurs loss g_t(i).
    Update w_{t+1,i} ← w_{t,i} · β^{g_t(i)} for all i.
end
```

Similarly, we define the expected loss of the learner at round t:

$$l_t = \frac{\sum_i w_{t,i} g_t(i)}{\sum_i w_{t,i}}$$

and the total expected loss is

$$L_T = \sum_{t=1}^{T} l_t$$

Theorem 9.6 *Applying Alg 3, we have*

$$\text{Regret}_T = L_T - \min_i \sum_{t=1}^{T} g_t(i) = O(\sqrt{T \log N})$$

Proof: Set $\epsilon = -\ln \beta > 0$ and $W_t = \sum_{t=1}^{T} w_{t,i}$ for all $t \le T$.
It is easy to verify the following inequalities

$$\forall x \ge 0,\ e^{-x} \le 1 - x + x^2 \tag{9.19}$$

$$1 + x \le e^x \tag{9.20}$$

Inspecting the sum of weights

$$\begin{aligned} W_{t+1} &= \sum_i w_{t,i} e^{-\epsilon g_t(i)} \\ &\le \sum_i w_{t,i}(1 - \epsilon g_t(i) + \epsilon^2 g_t(i)^2) \quad \text{(apply (9.19))} \\ &\le W_t(1 - \epsilon l_t + \epsilon^2) \quad \text{(notice that } g_t(i) \le 1) \\ &\le W_t e^{-\epsilon l_t + \epsilon^2} \quad \text{(apply (9.20))} \end{aligned}$$

Let $i^* = \arg\min_i \sum_{t=1}^{T} g_t(i)$, we have

$$e^{-\epsilon \sum_t g_t(i^*)} = w_{T+1,i^*} \le W_{T+1} \le W_1 e^{-\epsilon \sum_t l_t + \epsilon^2 T} = N e^{-\epsilon L_T + \epsilon^2 T}$$

Taking logarithm of both sides we get:

$$-\epsilon \sum_t g_t(i^*) \le \ln N - \epsilon L_T + \epsilon^2 T$$

It immediately follows that

$$\text{Regret}_T \le \epsilon T + \frac{\ln N}{\epsilon} \le 2\sqrt{T \ln N} = O(\sqrt{T \log N})$$

■

9.2.2.2 Minimax Theorem and its proof

Theorem 9.7 (Von Neumann's minimax theorem) *For any two-person zero-sum game defined by matrix $M \in \mathcal{R}^{m \times n}$*

$$\min_{p \in \Delta_m} \max_{q \in \Delta_n} p^T M q = \max_{q \in \Delta_n} \min_{p \in \Delta_m} p^T M q \quad (\Delta_k = \{a \in \mathcal{R}^k : \|a\|_1 = 1 \wedge a \succeq 0\})$$

We have learned that if two players follow pure strategy (i.e. restrict p and q to having only one non-zero entry), the result cannot be strictly better for the one who plays second if the playing order is reversed. (i.e. $\min_i \max_j M_{ij} \geq \max_j \min_i M_{ij}$)
In case of two players adopting mixed strategy, it is also intuitive that playing second is better, because playing second means having more information without any cost. So we need only focus on proving the reverse inequality. We will demonstrate that by adopting online learning algorithm, the first player can reduce the disadvantage of playing first to an infinitesimal as learning time goes to infinity.
Proof: The inequality

$$\min_{p\in\Delta_m} \max_{q\in\Delta_n} p^T M q \geq \max_{q\in\Delta_n} \min_{p\in\Delta_m} p^T M q \tag{9.21}$$

is straightforward. (Let $q^* \in \arg\max_{q\in\Delta_n} \min_{p\in\Delta_m} p^T M q$, $p^* \in \arg\min_{p\in\Delta_m} \max_{q\in\Delta_n} p^T M q$, $LHS = \max_{q\in\Delta_n} p^{*T} M q \geq p^{*T} M q^* \geq \min_{p\in\Delta_m} p^T M q^* = RHS$)

To show the reverse inequality, consider an online learning setting where Alg 3 is applied. In this case, row player (playing fist) is the learner, who chooses p_t such that $(p_t)_i = \frac{w_{t,i}}{W_t}$. Column player is the adversary who always select the optimal adversarial q_t (i.e. $q_t \in \arg\max_{q\in\Delta_n} p_t^T M q$). There are m experts who keep suggesting choosing one single row. Thus the loss function $g_t(i) = (Mq_t)_i$ (we can let $M_{ij} \in [0,1]$ without loss of generality).
Then from Theorem 9.6 we have

$$L_T - \min_i \sum_{t=1}^T g_t(i) = \sum_{t=1}^T p_t^T M q_t - \min_i (M \sum_{t=1}^T q_t)_i = O(\sqrt{T \log m})$$

Then

$$\begin{aligned}
\frac{1}{T}\sum_{t=1}^T p_t^T M q_t &\leq \frac{1}{T} \min_i \left(M \sum_{t=1}^T q_t \right)_i + O\left(\sqrt{\frac{\log m}{T}}\right) \\
&= \min_{p\in\Delta_m} \left(p^T M (\frac{1}{T}\sum_{t=1}^T q_t) \right) + o(1) \\
&\leq \max_{q\in\Delta_n} \min_{p\in\Delta_m} p^T M q + o(1)
\end{aligned}$$

And we have

$$\min_{p\in\Delta_m} \max_{q\in\Delta_n} p^T M q \leq \max_{q\in\Delta_n} \left(\frac{1}{T}\sum_{t=1}^T p_t^T \right) M q \leq \frac{1}{T}\sum_{t=1}^T \max_{q\in\Delta_n} p_t^T M q = \frac{1}{T}\sum_{t=1}^T p_t^T M q_t \leq \max_{q\in\Delta_n} \min_{p\in\Delta_m} p^T M q + o(1)$$

So we have

$$\min_{p\in\Delta_m} \max_{q\in\Delta_n} p^T M q \leq \max_{q\in\Delta_n} \min_{p\in\Delta_m} p^T M q$$

Combined with (9.21), the proof is completed. ■

References

[1] https://en.wikipedia.org/wiki/Online_machine_learning

[2] https://www.cs.princeton.edu/~rlivni/cos511/lectures/lect18.pdf

Machine Learning **Spring 2021**

Lecture 10: Online Learning, Multi-arm Bandits Problem

Lecturer: Liwei Wang *Scribe: Zijian Ding, Yifan Chen, Suchen Liu, Xin Xu*

10.1 More on Randomized Weight Updating

In this problem, we have an adversary and a learner.
The learner is asked to learn a distribution $\vec{x}$ over [**N**].
The following interaction repeats:

For $t = 1, 2, \cdots$:
1. Adversary provides $\vec{f_t} \in \{0,1\}^N$
2. Learner predictes $\langle \vec{f_t}, \vec{x} \rangle$ and gives its answer to adversary
3. Adversary reveals the answer for $\langle \vec{f_t}, \vec{x} \rangle$

We would like to design a mechanism so that the leaner makes $\delta - error$ at most finite times. More precisely, $\exists T$, $\forall t > T$, $|R_t - \langle f_t, x \rangle| \leq \delta$, where R_t is the prediction made by learner at time t.

The following algorithm bounded T to $O(\frac{\log N}{\delta^2})$.

Algorithm 10.1.1: Randomized Weight Updating

Initialize $x_1 = (\frac{1}{N} \cdots \frac{1}{N})$, set parameter $\epsilon \in (0, \delta]$
for $t = 1, 2, \cdots$ **do**
  if $\langle f_t, x_t \rangle$-$\langle f_t, x \rangle \geq \delta$ **then**
    $x_{t+1}[i] \leftarrow (1+\epsilon) x_t[i], \quad \forall f_{t,i} = 0$
    $x_{t+1}[i] \leftarrow x_t[i], \quad \forall f_{t,i} = 1$
    Normalize x_{t+1}
  end
  else if $\langle f_t, x_t \rangle$-$\langle f_t, x \rangle \leq -\delta$ **then**
    $x_{t+1}[i] \leftarrow (1+\epsilon) x_t[i], \quad \forall f_{t,i} = 1$
    $x_{t+1}[i] \leftarrow x_t[i], \quad \forall f_{t,i} = 0$
    Normalize x_{t+1}
  end
  else
    $x_{t+1} \leftarrow x_t$
  end
end

We now prove the efficiency of the algorithm.

Lemma 10.1 *The learner update at most* $\frac{2 \ln N}{\epsilon \delta}$ *times.*

Proof: Write $\{i \in [N] | f_t[i] = 0\}$ as F_0^t and $[N]/F_0^t$ as F_1^t. When $\langle f_t, x_t\rangle - \langle f_t, x\rangle \geq \delta$, we have

$$\langle f_t, x_t\rangle - \langle f_t, x\rangle = \sum_{i\in[N]} f_t[i](x_t[i] - x[i]) = \sum_{i\in F_1^t} x_t[i] - x[i] \geq \delta.$$

For $\sum_{i=1}^N x[i] = 1 = \sum_{i=1}^N x_t[i]$, we get

$$\sum_{i\in F_0^t} x_t[i] - x[i] \leq -\delta$$

Consider potential function $D(x\|x_t)$.

$$\begin{aligned} D(x\|x_{t+1}) - D(x\|x_t) =& \sum_{i=1}^N x[i] \ln \frac{x_t[i]}{x_{t+1}[i]} \\ =& \sum_{F_0^t} x[i] \ln \frac{1+\epsilon\sum_{F_0^t} x_t[i]}{1+\epsilon} + \sum_{F_1^t} x[i] \ln(1+\epsilon\sum_{F_0^t} x_t[i]) \\ =& \ln(1+\epsilon\sum_{F_0^t} x_t[i]) - \sum_{F_0^t} x[i]\ln(1+\epsilon) \end{aligned}$$

For given δ and $\epsilon \leq \delta$, $\ln(1+\epsilon) > \epsilon(1-\delta/2)$. And $\ln(1+\epsilon\sum_{F_0^t} x_t[i]) \leq \epsilon\sum_{F_0^t} x_t[i] \leq \epsilon\sum_{F_0^t} x[i] - \epsilon\delta$. Thus,

$$D(x\|x_{t+1}) - D(x\|x_t) \leq \epsilon\sum_{F_0^t} x[i] - \epsilon\delta - \sum_{F_0^t} x[i]\epsilon(1-\delta/2) \leq -\epsilon\delta/2$$

It's similar when $\langle f_t, x_t\rangle - \langle f_t, x\rangle \leq -\delta$, we have

$$\sum_{i\in F_1^t} x_t[i] - x[i] \leq -\delta$$

and

$$\begin{aligned} D(x\|x_{t+1}) - D(x\|x_t) =& \sum_{F_1^t} x[i] \ln \frac{1+\epsilon\sum_{F_1^t} x_t[i]}{1+\epsilon} + \sum_{F_0^t} x[i] \ln(1+\epsilon\sum_{F_1^t} x_t[i]) \\ =& \ln(1+\epsilon\sum_{F_1^t} x_t[i]) - \sum_{F_1^t} x[i]\ln(1+\epsilon) \\ \leq& \epsilon\sum_{F_1^t} x[i] - \epsilon\delta - \sum_{F_1^t} x[i]\epsilon(1-\delta/2) \\ \leq& -\epsilon\delta/2 \end{aligned}$$

This means that each update of the potential function reduces at least $\epsilon\delta/2$. Due to $D(x\|x_t) \geq 0$, and $D(x\|x_1) = \sum_{i\in[N]} x[i] \ln \frac{x[i]}{1/N} = \sum_{i\in[N]} x[i]\ln x[i] + \ln N \leq \ln N$, the learner update at most $\frac{2\ln N}{\epsilon\delta}$ times. ■

10.2 Multi-arm Bandits Problem

10.2.1 Setting

In a k-slot machine, each arm has a probability loss $\mu_i (i \in [k])$. And the game has T rounds in total, at each step t, the player chooses an arm a_t, and observes a loss $l_t(a_t)$. Be ignorant of μ_i, the player wants

to minimize the total loss in expectation. To be specific, for each arm $i \in [k]$, $l_1(a_i), \cdots, l_T(a_i)$ are i.i.d. random variables under some distribution $\mathcal{D}_i$ with mean μ_i. Define regret R_T as

$$R_T \coloneqq \mathbb{E}_{\mathcal{A}} \left[\sum_{t=1}^{T} \mu(a_t) - \mu^* \right]$$

where we replace $l_t(a_t)$ with mean $\mu(a_t)$ and define $\mu^* = \min_{i \in [k]} \mu_i$. Our goal is then to minimize R_T.

Multi-arm Bandits Problem is a simplified reinforcement learning problem. Because we don't know actual reward, the trade-off between exploration and exploitation is necessary.

10.2.2 UCB Algorithm

To minimize regret R_T, UCB algorithm gives an effective strategy, which is based on a simple principle: *Optimism in the face of uncertainty*. To be exact, after t rounds, each arm has an estimated interval of loss (with high probability), and we just assume the lowest bound of the interval is the loss probability of the arm. The algorithm is called upper confidence bound because people used to consider the probability of win in the past. The implement of UCB algorithm is as follow.

Algorithm 10.2.1: UCB Algorithm

```
Initialization: n_0(a) = 0(∀a ∈ [k]);
n_t(a) represents #times arm a is pulled at time t;
μ̂_t(a) represents the empirical loss of arm a at time t;
for t = 1, 2, ··· , T do
    For each arm a, compute UCB_t(a) = μ̂_{t−1}(a) − sqrt(ln T / n_{t−1}(a)) ;
    Pull the arm a_t = arg min_{a∈[k]} UCB_t(a) ;
    Update n_t(a_t), μ̂_t(a_t) ;
end
```

Note that at the very beginning, if there exists $n_{t-1}(a) = 0$, then $\mathrm{UCB}_t(a) = -\infty$. Therefore the algorithm tends to explore arm a. Now consider a substitute. If at each step of UCB algorithm, we choose the minimum among upper bounds instead of among lower bounds, then the algorithm prefers exploitation rather than exploration. Hence it may stick in some local optima.

10.2.3 Upper Confidence Bound

W.L.O.G. Suppose $\mu_1 = \min_{i \in [k]} \mu_i$, which means that the first arm has the lowest average loss.

Theorem 10.2 *Assume $\mu_1 \leq \mu_2, \cdots, \mu_k$, the regret of UCB algorithm can be bounded as:*

$$R_T = \mathbb{E}_{\mathcal{A}} \left[\sum_{t=1}^{T} \mu(a_t) - \sum_{t=1}^{T} \mu_1 \right] \leq \sum_{a:\Delta_a > 0} \left(\frac{16 \ln T}{\Delta_a} + 2\Delta_a \right) \tag{10.1}$$

where $\Delta_a = \mu_a - \mu_1$ denotes the gap of average loss between arm a and the optimal arm.

Proof: First of all, note that R_T can be written as

$$R_T = \mathbb{E}_{\mathcal{A}} \left[\sum_{t=1}^{T} \mu(a_t) \right] - \sum_{t=1}^{T} \mu_1 = \sum_{a=1}^{k} \Delta_a \cdot \mathbb{E}_{\mathcal{A}}[n_T(a)]$$

Since the regret R_T equals to

$$\sum_a (\#\text{times arm } a \text{ is pulled}) \times (\text{gap of loss between arm } a \text{ and the optimal arm})$$

which simply uses the technique of double counting.

Therefore we only need to prove that, for each sub-optimal arm $a(\Delta_a > 0)$, we have

$$\mathbb{E}_{\mathcal{A}}[n_T(a)] = O\left(\frac{\ln T}{\Delta_a^2}\right) + 2$$

■

Lemma 10.3 *For each sub-optimal arm a with $\Delta_a > 0$, we have*

$$\mathbb{E}_{\mathcal{A}}[n_T(a)] = O\left(\frac{\ln T}{\Delta_a^2}\right) + 2 \tag{10.2}$$

where $n_T(a)$ denotes #times arm a is pulled in T rounds.

Proof: By linearity of expectation, $\mathbb{E}_{\mathcal{A}}[n_T(a)]$ could be written as

$$\mathbb{E}_{\mathcal{A}}[n_T(a)] = \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{1}[a \text{ is pulled at round } t]\right] = \sum_{t=1}^{T} \Pr[a \text{ is pulled at round } t]$$

Therefore for any n, we have

$$\begin{aligned}
\mathbb{E}_{\mathcal{A}}[n_T(a)] &= \sum_{t=1}^{T}\sum_{k=1}^{T} \Pr[a_t = a \wedge n_t(a) = k] \\
&\le n + \sum_{t=1}^{T}\sum_{k=n+1}^{T} \Pr[a_t = a \wedge n_t(a) = k] \\
&\le n + \sum_{t=n+1}^{T} \Pr[a_t = a \wedge n_{t-1}(a) \ge n]
\end{aligned}$$

Then our goal is to better estimate $\mathbb{E}_{\mathcal{A}}[n_T(a)]$ by choosing the parameter n according to Δ_a. Before fine-turning n, we present a proposition here.

Proposition. If sub-optimal arm $a(\Delta_a > 0)$ is pulled at time t, then we can claim that at least one of the following events occur:

1. $\hat{\mu}_{t-1}(1) > \mu_1 + \sqrt{\frac{\ln T}{n_{t-1}(1)}}$
2. $\hat{\mu}_{t-1}(a) < \mu_1 + \sqrt{\frac{\ln T}{n_{t-1}(a)}} = \mu_a - \Delta_a + \sqrt{\frac{\ln T}{n_{t-1}(a)}}$

Suppose neither of them occur, then $\hat{\mu}_{t-1}(1) - \sqrt{\frac{\ln T}{n_{t-1}(1)}} \le \mu_1 \le \hat{\mu}_{t-1}(a) - \sqrt{\frac{\ln T}{n_{t-1}(1)}}$. But in UCB algorithm, arm a is choosen to be $a_t = \arg\min_{a\in[k]} \left(\hat{\mu}_{t-1}(a) - \sqrt{\frac{\ln T}{n_{t-1}(a)}}\right)$, raising a contradiction.

Union Bound. With the proposition above, we have

$$\begin{aligned}\Pr\left[a_t = a \wedge n_{t-1}(a) \geq n\right] \leq & \Pr\left[\hat{\mu}_{t-1}(1) > \mu_1 + \sqrt{\frac{\ln T}{n_{t-1}(1)}}\right] \\ & + \Pr\left[\hat{\mu}_{t-1}(a) < \mu_a - \Delta_a + \sqrt{\frac{\ln T}{n_{t-1}(a)}} \wedge n_{t-1}(a) \geq n\right]\end{aligned}$$

And sum over t,

$$\begin{aligned}\sum_{t=n+1}^{T} \Pr\left[a_t = a \wedge n_{t-1}(a) \geq n\right] \leq & \sum_{t=n+1}^{T} \Pr\left[\hat{\mu}_{t-1}(1) > \mu_1 + \sqrt{\frac{\ln T}{n_{t-1}(1)}}\right] \\ & + \sum_{t=n+1}^{T} \Pr\left[\hat{\mu}_{t-1}(a) < \mu_a - \Delta_a + \sqrt{\frac{\ln T}{n_{t-1}(a)}} \wedge n_{t-1}(a) \geq n\right]\end{aligned}$$

Deal with the two summations separately. For the first summation,

$$\begin{aligned}\sum_{t=n+1}^{T} \Pr\left[\hat{\mu}_{t-1}(1) > \mu_1 + \sqrt{\frac{\ln T}{n_{t-1}(1)}}\right] &\leq \sum_{t=n+1}^{T}\sum_{k=1}^{T} \Pr\left[\hat{\mu}_{t-1}(1) > \mu_1 + \sqrt{\frac{\ln T}{k}} \wedge n_{t-1}(1) = k\right] \\ &\leq \sum_{t=n+1}^{T}\sum_{k=1}^{T} \Pr\left[\hat{\mu}_{t-1}(1) > \mu_1 + \sqrt{\frac{\ln T}{k}} \,\middle|\, n_{t-1}(1) = k\right] \\ &\leq \sum_{t=n+1}^{T}\sum_{k=1}^{T} \exp\left(-2k\left(\sqrt{\frac{\ln T}{k}}\right)^2\right) \\ &= \sum_{t=n+1}^{T}\sum_{k=1}^{T} \frac{1}{T^2} \leq 1\end{aligned}$$

where $n_t(1)$ denotes the counter of pulling arm 1, and the third inequality is a Chernoff bound.

Similarly, for the second summation, we choose n such that $\Delta_a = 2\sqrt{\frac{\ln T}{n}}$, and therefore

$$\begin{aligned}&\sum_{t=n+1}^{T} \Pr\left[\hat{\mu}_{t-1}(a) < \mu_a - \Delta_a + \sqrt{\frac{\ln T}{n_{t-1}(a)}} \wedge n_{t-1}(a) \geq n\right] \\ \leq &\sum_{t=n+1}^{T} \Pr\left[\hat{\mu}_{t-1}(a) < \mu_a - \sqrt{\frac{\ln T}{n_{t-1}(a)}} \wedge n_{t-1}(a) \geq n\right] \leq 1\end{aligned}$$

(the first inequality holds because $n_{t-1}(a) \geq n$). Combining these three inequalities, we have

$$\mathbb{E}_{\mathcal{A}}[n_T(a)] \leq n + 2 = O\left(\frac{\ln T}{\Delta_a^2}\right) + 2$$

which completes the proof.

■

Note: The bound given above is instance-dependent regret bound since it contains Δ_a. If we treat Δ_a as a constant, then $R_T = O(k \ln T)$, which provides a better bound than $O(\sqrt{T})$ in Expert Advice Problem. But

Δ_a can be small for some arms (Δ_a can not be seen as a constant in this situation), and as a consequence, the regret bound given above will be loose. In fact, if Δ_a is treated carefully, we can have a better bound for R_T than this theorem does.

Next, we will give a instance-independent regret bound which is worst-case regret bound of UCB.

Theorem 10.4 *Assume Δ_a is bounded, then the worst-case regret bound of UCB is:*

$$R_T = O\left(\sqrt{T \cdot k \ln T}\right) \tag{10.3}$$

Proof: Divide the arms into two groups at the gap of $\delta = \sqrt{\frac{k \cdot \ln T}{T}}$. For $\Delta_a < \delta$, we have

$$R_T^{(1)} = \sum_{t=1}^{T} \sum_{a:\Delta_a < \delta} \Delta_a \cdot \Pr[a_t = a] \leq T \cdot \delta \leq \sqrt{T \cdot k \ln T}$$

And for $\Delta_a \geq \delta$, by applying the theorem above, we have

$$R_T^{(2)} \leq \sum_{a:\Delta_a \geq \delta} \left(\frac{16 \ln T}{\Delta_a} + 2\Delta_a\right) = O\left(k \cdot \frac{\ln T}{\delta}\right) = O\left(\sqrt{T \cdot k \ln T}\right)$$

where Δ_a is bounded. Therefore $R_T = R_T^{(1)} + R_T^{(2)} = O\left(\sqrt{T \cdot k \ln T}\right)$.

■

References

[1] https://en.wikipedia.org/wiki/Multi-armed_bandit

Machine Learning **Spring 2021**

Lecture 11: Thompson Sampling, Differential Privacy

Lecturer: Liwei Wang *Scribe: Xiyan Xu, Haoyu Wang, Zhewen Hao, Ningyuan Li, Tianran Zhu*

11.1 Thompson Sampling

Thompson sampling was first proposed by William R. Thompson in 1933. The principle behind Thompson sampling is Beta distribution. Beta distribution forms a family of continuous probability distributions on the interval $(0, 1)$. And for $Beta(\alpha, \beta)(\alpha > 0, \beta > 0)$, the probability density function is given by:

$$f(x, \alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

For a Bernoulli bandit problem, we can use Beta distribution to update the Bernoulli loss, because if the prior is a $Beta(\alpha, \beta)$, we can sure that the posterior distribution is $Beta(\alpha + 1, \beta)$ or $Beta(\alpha, \beta + 1)$.
The Thompson Sampling algorithm assumes arm a to have prior $Beta(1, 1)$ on μ_a, and initialize $S_a = 0, F_a = 0$. At time t, the algorithm updates the distribution on $\mu_a$as $Beta(S_a + 1, F_a + 1)$. The algorithm then samples from $\mu_a{}'s$ posterior distributions. According to the probability of its mean being the largest, then plays an arm. And if $r_t(a_t) \notin \{0, 1\}$, the algorithm tosses a $r_t(a_t)$-coin to get a $\{0, 1\}$-result.

Algorithm 1 Thompson Sampling

Initialization $a = 1, 2 \cdots K, S_a = 0, F_a = 0$
for $t = 1, 2, ..., T$ **do**
    For each arm a, select $\Theta_a(t) \sim Beta(S_a + 1, F_a + 1)$
    Play arm $a_t = \arg\max_a \Theta_a(t)$ and get reward $r_t(a_t) \in [0, 1]$
    Toss a $r_t(a_t)$-coin, get $\widetilde{r}_t(a_t) \in \{0, 1\}$
    If $\widetilde{r}_t(a_t) = 1$, $S_a \leftarrow S_a + 1$; Else, $F_a \leftarrow F_a + 1$

Theorem 11.1 *For $\forall \epsilon > 0$, Thompson sampling has regret:*

$$R_T \leq (1 + \epsilon) \sum_{a \neq a^*} \frac{\log T\Delta_a}{d(\mu_a, \mu_{a^*})} + O(\frac{k}{\epsilon^2})$$

where $d(u, v) = u \ln \frac{u}{v} + (1 - u) \ln \frac{1-u}{1-v}$

Proof: We notice that we have the equation

$$R_T = E_{\mathcal{A}} \sum_{t=1}^{T} l_t(a_t) - \sum_{t=1}^{T} \mu_1 = \sum_{a=1}^{k} \Delta_a E_{\mathcal{A}}[n_T(a)]$$

Now we need to bound each arm's $E_{\mathcal{A}}[n_T(a)]$to prove our theorem. It's quite complex so you can refer to [1] to see the detailed calculation:

$$E_{\mathcal{A}}[n_T(a)] \leq (1+\epsilon_0)^2 \frac{\log T}{d(\mu_a, \mu_{a^*})} + O(\frac{1}{\epsilon_0^2})$$

And then we take $\epsilon = 3\epsilon_0$,

$$R_T \leq \sum_{a \neq a^*} (1+\epsilon_0)^2 \frac{\log T\Delta_a}{d(\mu_a, \mu_{a^*})} + O(\frac{1}{\epsilon_0^2}) \leq (1+\epsilon) \sum_{a \neq a^*} \frac{\log T\Delta_a}{d(\mu_a, \mu_{a^*})} + O(\frac{k}{\epsilon^2})$$

■

11.2 Differential Privacy

Data leakage refers to a mistake made by the creator of a machine learning model in which they accidentally share information with the user. For example, in the traditional SVM model, the output classifier $f(x) = \sum_i \alpha_i y_i k(x : x_i)$, gives the information of all of the margin. Before solving this problem, we have to define it first. It's hard to define what's privacy, which is quite abstract and philosophical. Instead, we can define what is a privacy leak.

The primary object of a privacy-preserving algorithm is to release some statistical information about the dataset but do not leak much information about specific data. We now model the privacy leakage in query answering.

Definition 11.2 (Neighboring Datasets) *Consider dataset $D = \{x_1, x_2, \cdots, x_n\}$, $x_i \in X, D \in X^n$. We say that two datasets $D, D' \in X$ are neighboring if they differ in only a single data element.*

Definition 11.3 (Counting Query) *A counting query Q_h, defined in terms of a predicate $h : X \to \{0, 1\}$ is defined to be*

$$Q_h(D) := \frac{1}{|D|} \sum_i h(x_i), h(x_i) \in \{0, 1\}$$

It evaluates the fraction of elements in the dataset that satisfy the prediction h.

Definition 11.4 (Differential Privacy) *Let A be a randomized algorithm, $A : X^n \to R$, with input D, we say A satisfies ϵ-differential privacy if for all neighboring datasets $D, D^{'} \in X^n$, and all set $S \subseteq Output(A)$, we have*

$$\Pr[A(D) \in S] \leq e^{\varepsilon} \Pr[A(D') \in S]$$

This definition captures the core nature of privacy. It describes the stability of the answer under the data disturbance. If the change of a certain data will cause the result to fluctuate greatly, then this algorithm does not protect the data privacy well. In particular, if ϵ is equal to 0, the answer for adjacent data will be the same.

Definition 11.5 ((α, β)-accuracy) *Say randomized algorithm A has $(\alpha, \beta) -$ accuracy with respect to counting query Q if:*

$$\forall D \in X^n, \quad \mathbb{P}(|A(D) - Q(D)| \geq \alpha) \leq \beta$$

Definition 11.6 (The Laplace Distribution) *The Laplace Distribution(centered at 0) with scale σ is the distribution with probability density function:*

$$f(x|\sigma) = \frac{1}{2\sigma} exp(\frac{-|x|}{\sigma})$$

The Laplace distribution is a symmetric version of the exponential distribution, and we will write $Lap(\sigma)$ to denote the Laplace distribution with scale σ.

Lemma 11.7 *let $f(x|\sigma)$ denote the probability density function of the Laplace Distribution, then*

$$\forall x_1, x_2 \in \mathbb{R}, \frac{f(x_1|\sigma)}{f(x_2|\sigma)} \leq exp(\frac{|x_1 - x_2|}{\sigma})$$

.

Proof:

$$\frac{f(x_1|\sigma)}{f(x_2|\sigma)} = exp(\frac{-|x_1| + |x_2|}{\sigma}) \leq exp(\frac{|x_1 - x_2|}{\sigma})$$

. ■

We will now define the **Laplace Mechanism**. As its name suggests, the Laplace mechanism will simply compute f, and perturb each coordinate with noise drawn from the Laplace distribution.

Definition 11.8 (The Laplace Mechanism) *Given any function $f : X^n \to \mathbb{R}^k$, the Laplace Mechanism responds to f by returning $f(x) + Z$, where $Z = (Y_1, \cdots, Y_k)$ is a k-dimension random variable and $\forall i \in [k], i.i.d. Y_i \sim Lap(\sigma)$.*

In the case where f is just a single query($k = 1$), the Laplace Mechanism return $f(D) + Z, Z \sim Lap(\sigma)$.

Theorem 11.9 *The Laplace Mechanism preserves ϵ-differential privacy and has $(\alpha, \beta) - accuracy$ with respect to the single counting query $Q : X^n \to [0, 1]$, where $\epsilon = \frac{1}{n\sigma}, \alpha = \sigma ln \frac{1}{\beta}, \beta := neg(n)$ i.e.$\forall k \in \mathbb{N}, \beta < n^k$.*

Proof: Let A denotes the Laplace Mechanism(consistent with the notation of the randomized algorithm we defined earlier). According to Lemma 11.7, for every $a \in Output(A)$, the ratio

$$\frac{\mathbb{P}(A(D) = a)}{\mathbb{P}(A(D') = a)} = \frac{f(a - Q(D))}{f(a - Q(D'))} \leq exp(\frac{|Q(D) - Q(D')|}{\sigma}) \leq exp(\frac{1}{n\sigma})$$

As for α, consider the equation

$$\mathbb{P}(|A(D) - Q(D)| \geq \alpha) = 2 \cdot \frac{1}{2\sigma} \int_{\alpha}^{\infty} e^{-\frac{t}{\sigma}} dt = \beta$$

We get $\alpha = \sigma ln \frac{1}{\beta}$. ■

Definition 11.10 (Accuracy(multiple queries)) *Let $Q = (Q_1, \cdots, Q_k) : X^n \to \mathbb{R}^k$ be a counting query sequence. Let A be the algorithm with k output:$A(D) = (A_1(D), \cdots, A_k(D))$.*
We say A has $(\alpha, \beta) - accuracy$ with respect to queries in Q, if for every $D \in X^n$:

$$\mathbb{P}(||A(D) - Q(D)||_\infty \geq \alpha) \leq \beta$$

i.e. with probability at least $1 - \beta$, $\max_{i \in [k]} |Q_i(D) - A_i(D)| \leq \alpha$.

Lemma 11.11 *The Laplace Mechanism satisfies $k\epsilon$-differential privacy and $(\alpha, k\beta)$-accuracy with respect to a sequence of k queries $Q = (Q_1, \cdots, Q_k)$, if for every Q_i $(i = 1, \ldots, k)$, the Laplace Mechanism satisfies ϵ-differential privacy and (α, β)-accuracy respectively.*

Proof: Let $A_i(D)$ denote the i-th output of the Laplace Mechanism. By definition, $[A_i(D) - Q_i(D)] \sim Lap(\sigma)$ independently, and the joint probability density function of $A(D) = (A_1(D), \cdots, A_k(D))$ is

$$f_{A(D)}(\vec{x} = (x_1, \ldots, x_k)) = \prod_{i=1}^{k} f_{A_i(D)}(x_i) = \prod_{i=1}^{k} f_\sigma(x_i - Q_i(D)) \ .$$

From ϵ-differential privacy of A_i, we have $f_{A_i(D)}(x_i) \leq e^\epsilon f_{A_i(D')}(x_i)$, and thus

$$f_{A(D)}(\vec{x}) \leq e^{k\epsilon} f_{A(D')}(\vec{x}) \ .$$

Therefore $\forall S \subseteq R^k$, we have

$$\frac{\mathbb{P}(A(D) \in S)}{\mathbb{P}(A(D') \in S)} = \frac{\int_{\vec{x} \in S} f_{A(D)}(\vec{x})\, d\vec{x}}{\int_{\vec{x} \in S} f_{A(D')}(\vec{x})\, d\vec{x}} \leq e^{k\epsilon} \ ,$$

so the Laplace Mechanism satisfies $k\epsilon$-differential privacy for $Q = (Q_1, \cdots, Q_k)$.
By union bound,

$$\mathbb{P}(||A(D) - Q(D)||_\infty \geq \alpha) \leq \sum_{i=1}^{k} \mathbb{P}(|A_i(D) - Q_i(D)| \geq \alpha) \leq k\beta,$$

the $(\alpha, k\beta)$-accuracy is satisfied. ■

As a corollary, we have

Theorem 11.12 *The Laplace Mechanism satisfies $\frac{k}{n\sigma}$-differential privacy and (α, β)-accuracy with respect to a sequence of k counting queries $Q = (Q_1, \cdots, Q_k)$, where $\alpha = \sigma \ln \frac{k}{\beta}$.*

Given a fixed β, when n is sufficiently large, we expect that $\alpha = O(\frac{1}{\sqrt{n}})$, which is close to the sampling error, or that $\alpha = o(1)$, so that the output gets better accuracy as n grows. To preserve privacy, people usually demand that $\epsilon = O(1)$. These requirements put a limit on k, the available number of queries.

Theorem 11.13 *If a Laplace Mechanism satisfies ϵ-differential privacy and (α, β)-accuracy with respect to a sequence of k counting queries $Q = (Q_1, \cdots, Q_k)$, where $\epsilon = O(1)$, $\alpha = o(1)$, and given $\beta \in (0, 1)$, then it's required that $k = o(\frac{n}{\ln n})$.*

Proof: From $\epsilon = \frac{k}{n\sigma} = O(1)$ we have $k = O(n\sigma)$. From $\alpha = \sigma \ln \frac{k}{\beta} = o(1)$ we have $\sigma = o(\frac{1}{\ln \frac{k}{\beta}}) = o(\frac{1}{\ln k})$. Thus we have $k \ln k = o(n)$, therefore $k = o(\frac{n}{\ln n})$. ■

References

[1] Shipra Agrawal and Navin Goyal(2017). Near-Optimal Regret Bounds for Thompson Sampling.

[2] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy.

Machine Learning **Spring 2021**

Lecture 12: Differential Privacy

Lecturer: Liwei Wang *Scribe: Zhaomeng Deng, Haoyu Jin, Yuke Lou, Yiming Wang, Hanyue Lou*

12.1 Review

In last class we introduced Differential Privacy. In our modeled situation, the true dataset is D. A user poses a query and wants to know $q(D)$. The algorithm $\mathcal{A}$ gives the user an answer $A(D)$ (which is generated from a random distribution).

We often assume that the query $q(D)$ is a *counting query*, which means that:

$$q(D) := \frac{1}{|D|} \sum_{x_i \in D} f(x_i), \quad f(x) \in [0, 1]$$

And we define "ϵ-DP" as below:

Definition 12.1 ($\epsilon-$DP) *Let $\mathcal{A}$ be a randomized algorithm $\mathcal{A} : X^n \to \mathbb{R}$, with input D, we say $\mathcal{A}$ satisfies "ϵ-DP" if for any neighboring datasets $D, D' \in X^n$ and any item $a \in Output(\mathcal{A})$, we have*

$$\Pr[\mathcal{A}(D) = a] \leq \mathrm{e}^{\epsilon} \Pr[\mathcal{A}(D') = a]$$

For any set S and neighboring data set D and D'. We call this "Pure-DP". We can add a Laplacian noise to the true answer before answering the user's query to achieve Pure-DP.

12.2 Approximate DP

Pure-DP is a strong definition. It requires all of the neighboring datasets are bounded by the likelihood ratio. It is also reasonable to only bound algorithm outputs of high probability with a likelihood ratio, and restrict others with a constant. It also can achieve good results in practice. This idea is called Approximate DP($(\epsilon - \delta)-$DP).

Definition 12.2 ($(\epsilon - \delta)-$DP) *Let $\mathcal{A}$ be a randomized algorithm $\mathcal{A} : X^n \to \mathbb{R}$, with input D, we say $\mathcal{A}$ satisfies Approximate DP, or $(\epsilon-\delta)-DP$ if for any neighboring datasets $D, D' \in X^n$ and any set S, we have*

$$\Pr[\mathcal{A}(D) \in S] \leq \mathrm{e}^{\epsilon} \Pr[\mathcal{A}(D') \in S] + \delta$$

Proposition 12.3 (A sufficient condition for $(\epsilon - \delta)-$DP) *Let B(D, D') $= \left\{x, \frac{\Pr(\mathcal{A}(D')=x)}{\Pr(\mathcal{A}(D')=x)} \geq \mathrm{e}^{\epsilon}\right\}$. If $\forall D$, $\Pr\left(\mathcal{A}(D) \in B(D, D')\right) \leq \delta$ then $\mathcal{A}$ preserves $(\epsilon - \delta)-DP$.*

To achieve Approximate DP, we can simply add a Gussian noise $Gussian(\sigma^2)$ to the query answer $q(D)$ to get the output, similar to the Laplacian-Mechanism we used for Pure-DP.

12.3 The Exponential Mechanism

Now let's return to Pure-DP. In last class, to make our Laplacian Mechanism preserve both ϵ-DP and (α, β)-accuracy, where ϵ, α, β are all $O(1)$, we can only allow our user to make $k = O(n)$ queries. This sounds too few. Can we do better? Like $k \gg n$ or even $k = \exp(\text{poly}(n))$?

In the Laplacian Mechanism, we returned close answers with high probability and large-error answers with low probability. Generalizing this we can derive the "Exponential Mechanism".

We define $u(D, x)$ as the Utility Function, or "how accurate the returned value x is, given dataset D". Then we return query result $\mathcal{A}(D)$ according to the distribution:

$$\mathbb{P}(\mathcal{A}(D) = x) = \frac{1}{z} \mathrm{e}^{\frac{u(D,x)}{\sigma}}$$

In which σ is a constant, and z is the normalizing constant of the distribution. We can see that in this distribution, "better" answers have higher probability.

Now we should try to analyze the privacy leakage and accuracy of the Exponential Mechanism.

First define the sensitivity of a given Utility Function $u(D, x)$:

$$\Delta u \triangleq \max_{D, D', x} |u(D, x) - u\left(D', x\right)|$$

Theorem 12.4 *the Exponential Mechanism preservers ϵ-differential privacy, where $\epsilon = \frac{2\Delta u}{\sigma}$.*

Proof:

$$\mathbb{P}(\mathcal{A}(D) = a) = \frac{\exp\left(\frac{u(D,a)}{\sigma}\right)}{\sum_{a \in \Lambda} \exp\left(\frac{u(D,a)}{\sigma}\right)}$$
$$\mathbb{P}\left(\mathcal{A}\left(D'\right) = a\right) = \frac{\exp\left(\frac{u\left(D', a\right)}{\sigma}\right)}{\sum_{a \in \Lambda} \exp\left(\frac{u\left(D', a\right)}{\sigma}\right)}$$

Using $u\left(D', a\right) - \Delta u \leq u(D, a) \leq u\left(D', a\right) + \Delta u$, we have

$$\mathbb{P}(\mathcal{A}(D) = a) \leq \exp\left(\frac{2\Delta u}{\sigma}\right) \mathbb{P}\left(\mathcal{A}\left(D'\right) = a\right)$$

■

we say Exponential Mechanism has $\alpha - \beta$ accuracy, if:

$$\Pr\left(u(D, \mathcal{A}(D, q)) \leq u^* - \alpha\right) \leq \beta$$

where $u^* \triangleq \max\, u(D, x)$.

12.4 BLR Mechanism[1]

The Blum-Ligett-Roth (BLR) Mechanism is an data release mechanism that preserves both $\epsilon - d.p. and (\alpha, \beta)$ accuracy. The idea of BLR Mechanism is using random sampling instead of adding random noise.First we introduce the algorithm of BLR Mechanism:

Assumption 12.5 *Data space is discrete and finite.*

Param : $k =$ the number of queries , $\mathcal{X} = \{0,1\}^d, N = |\mathcal{X}|, n = |D|, \varepsilon, (\alpha, \beta), \sigma = \frac{2}{n\varepsilon}$

1. Let $m = \frac{2\log(2k)}{\alpha^2}$

2. For every $\hat{D} \in \mathcal{X}^m$,Output synthetic dataset $\hat{D}$

with probability $\mathcal{P}(\mathcal{A}(D) = \hat{D}) \sim \exp\left(\frac{u(D,\hat{D})}{\sigma}\right)$

where $u(D, \hat{D}) = -\max_{i\in[k]} \mid q_i(D) - q_i(\hat{D}) \mid$

Now we should try to analyze the privacy leakage and accuracy of the BLR Mechanism.

Theorem 12.6 *The privacy loss of BLR Mechanism with the parameters above is ϵ.*

Proof: The database release in BLR Mechanism is to sample from the distribution defined by the exponential mechanism. By the privacy loss analysis of the exponential mechanism in 12.4 and putting the parameters in the formula, we have the privacy loss of BLR Mechanism is $\epsilon = \frac{2\Delta u}{\sigma} = \frac{2\cdot\frac{1}{n}}{\frac{2}{n\epsilon}} = \epsilon$. ■

As for the accuracy, we can define the dataset $\hat{D} \in \chi^m (m = \dfrac{2\log(2k)}{2\alpha^2})$ to be "good" if

$$\max_{i\in[k]} |q_i(D) - q_i(\hat{D})| \leq \frac{\alpha}{2}$$

We can use Probability Method to prove that the "good" dataset $\hat{D}$ exists: Consider those as which are subsets of D, and suppose $\hat{D}$ is randomly drawn from D. By Chernoff bound and union bound,

$$\Pr\left(\exists i \in [k], |q_i(D) - q_i(\hat{D})| > \frac{\alpha}{2}\right) \leq 2k\mathrm{e}^{-\frac{a^2 m}{2}} = 1$$

So $\exists \hat{D} \in \hat{D} \in \chi^m$ satisfies $u(D, \hat{D}) \leq -\frac{\alpha}{2}$.

Correspondingly, we define $\hat{D}$ to be "bad" if $u(D, \hat{D}) \leq u^* - \alpha$, where u^* is the optimal value of $u(D, D')$. By the property of the Exponential Mechanism, each "bad" $\hat{D}$ appears with probability $p_b = \Pr(\mathcal{A}(D) = \hat{D}) \leq \frac{1}{z}\exp(\frac{u^*-\alpha}{\sigma}) \leq \frac{1}{z}\exp(-\frac{\alpha n\epsilon}{2})$. The "good" $\hat{D}$ appears with probability $p_g \geq \frac{1}{z}\exp(-\frac{\alpha n\epsilon}{4})$, so each dataset $\hat{D}$ satisfies $u(D, \hat{D}) \leq -\alpha$ appears with probability $p \leq p_b/p_g \leq \exp(-\frac{\alpha n\epsilon}{4})$.

The total number of dataset is $|\chi^m| = N^m$, we only need to set $N^m \exp(-\frac{\alpha n\epsilon}{4}) \leq \beta$ to satisfiy the and (α, β) accuracy. By this formula, we can calculate the asymptotic boundary of α. Then we have the theorem below:

Theorem 12.7 *The BLR Mechanism satisfies ϵ-DP and (α, β) accuracy, where*

$$\alpha = O\left(\left(\frac{\log k \log N + \log\frac{1}{\beta}}{n\epsilon}\right)^{\frac{1}{3}}\right)$$

12.5 Q&A

At the end of class, XSJ pointed out that many steps in the BLR-Mechanism's proof are very loose, and asked if there was a better bound. The lecturer told us that when you meet such questions about "bounds" when reading papers, there are usually three possible answers:

(a) The mechanism actually has better performance, and a better bound can be proved if the researcher does more math.

(b) This is already the best bound possible for this mechanism. However there exists a better mechanism for this problem that has a better bound.

(c) It is impossible for any mechanism for this problem to exceed this bound.

In this particular case, the answer is (b). There is a better mechanism that supports $\alpha \sim \frac{1}{\sqrt{n}}$.

References

[1] Blum, A., Ligett, K., & Roth, A. (2013). A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM), 60(2)*, 1-25.

Machine Learning **Spring 2021**

Lecture 13: Differential Privacy, Boosting

Lecturer: Liwei Wang *Scribe: Yutong Yin, Haowei Lin, Yuhan Guo, Jingwu Tang, Siyuan Chen*

13.1 Local Differential Privacy

In the past decades, Differential Privacy, which gives strong privacy guarantees for users has risen to popularity. The key idea is that a user is given plausible deniability by adding random noise to their input. We've talked about mechanisms like Laplace mechanism and BLR mechanism to preserve both privacy and accuracy well. However, the setting we have discussed in is *centralized Differential Privacy* setting, where noise is added to the database. This approach to Differential Privacy requires users to have trust the database maintainer to keep their privacy because all the raw personal data is collected by the dataholder.

A stronger privacy guarantee for each individual users can be given in the local setting as there is no need to trust a centralized authority. In Local Differential Privacy setting, each user encodes and perturbs their own inputs before transmitting them to the untrusted server. This server can then compute statistical queries on the input data. We can give the formal definition of LDP proposed by [1] as follows.

Definition 13.1 *We say that an algoirhm A satisfies ϵ-Local Differential Privacy where $\epsilon > 0$ if and only if for any input v and v'*

$$\forall y \in Output(A) : \frac{Pr[A(v) = y]}{Pr[A(v') = y]} \leq e^{\epsilon}$$

where $Output(A)$ denotes every possible output of the algorithm A.

In order to show how to deal with LDP problem, an example was given.

Example 13.2 *Assume there is a survey released for an university to know whether the students are satisfied with their president or not. Respondents should answer "Yes" or "No" with their real names known by the surveyor. The problem is, how to design a mechanism to get meaningful statistics while preserving the privacy of every respondent ?*

The problem could be solved by the following mechanism.

Randomized Response Although Local Differential Privacy has only recently been gaining traction and popularity, the ideas behind it are much older. The key idea of Local Differential Privacy is *randomized response*, a surveying technique first introduced by Warner in [2]. To answer the question the respondent gives a randomized output, which is truthful for probability p and fake for probability $1 - p$. If p is close to $1 - p$, the output distribution of algorithm A for different inputs will be similar. This preserves respondents' privacy by giving them strong plausible deniability while allowing for computation of accurate population statistics.

Given that respondents comply with the protocol, respondents will answer the question truthfully 55% of the time. An unbiased estimate of the true number of "Yes" answer can therefore be computed by $10(X - 0.45)$ where X refers to the fraction of respondents who answered in the positive. The survey participants are given a privacy guarantee of $\epsilon = \ln(\frac{0.55}{0.45}) = \ln \frac{11}{9}$.

LDP has been seen in practical deployments by major technology organizations such as Apple [3], who use it to collect usage statistics and find commonly used emojis, new words that are not part of the dictionary yet and to improve user behaviour, Google [1], who use it in Chrome to collect commonly chosen homepages, settings, and other web browsing behaviour, and Microsoft [4], who use it for their collection of telemetry data.

13.2 Composition of Privacy Leakage

Previously, we have defined two kinds of differential privacy: ϵ-differential privacy (pure-dp) and (ϵ, δ)-differential privacy (approximate-dp). And we also discussed the composition of privacy leakage in $\epsilon - dp$ scenario.

Theorem 13.3 *For Laplace Mechanism preserving ϵ_0-differential privacy at each round, the total privacy leakage of k rounds is $k\epsilon_0$ (i.e. k rounds in total preserves $k\epsilon_0$-dp).*

Lemma 13.4 *(McDiarmid inequality)Let $X_1, ..., X_m \in \chi^m$ be a set of $m \geq 1$ independent random variables and assume that there exist c1,.....,cm > 0 such that f: $\chi^m \to \mathbb{R}$ satisfies the following conditions:*

$$|f(x_1, ..., x_i, ..., x_m) - f(x_1, ..., x_i^{'}, ..., x_m)| \leq c_i$$

for all i ∈[m] and any points $x_1, ..., x_m, x_i^{'} \in \chi$. Let f(S) denote f($X_1, ..., X_m$), then, for all ϵ >0, the following inequalities hold:

$$Pr[f(S) - E[f(S)] \geq \epsilon] \leq exp(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2})$$

$$Pr[f(S) - E[f(S)] \leq -\epsilon] \leq exp(\frac{-2\epsilon^2}{\sum_{i=1}^m c_i^2})$$

Theorem 13.5 *For all $\epsilon_0, \delta \geq 0$, k rounds of independent Lap(σ), ϵ_0-dp mechanisms satisfies (ϵ, δ)-dp where:*

$$\epsilon = \sqrt{2kln(1/\delta)}\epsilon_0 + 2k\epsilon_0^2$$

Proof: From the Proposition 12.3 in last lecture, we need to proof the bound of f($a_1, ..., a_k$) = $\sum_{i=1}^k ln\frac{p_i(a_i)}{q_i(a_i)}$, where $\frac{p_i(a_i)}{q_i(a_i)}$ denotes $\frac{Pr(A_i(D)=a_i)}{Pr(A_i(D^{'})=a_i)}$. Notice that it is a problem about the sum of k i.i.d. variables, we can estimate its expectation at first and then use concentration inequality to proof its bound.

Because of Lap(σ), we have $-\epsilon_0 \leq ln\frac{p_i(a_i)}{q_i(a_i)} \leq \epsilon_0$. Let g(a) denote $ln\frac{p(a)}{q(a)}$, observe $1 = \int p(a)da = \int q(a)da = \int p(a)e^{-g(a)}da$. So we have:

$$E_{a_i\ p_i}[ln\frac{p_i(a_i)}{q_i(a_i)}] = \int p(a)g(a)da \tag{13.1}$$

$$= 1 - 1 + \int p(a)g(a)da \tag{13.2}$$

$$= \int p(a)[e^{-g(a)} - 1 + g(a)]da \tag{13.3}$$

Let h(x) denote $e^{-x}-1+x$ where x $\in [-\epsilon_0, \epsilon_0]$. Using derivation we have $h(x) \leq max\{h(-\epsilon_0), h(\epsilon_0)\}$. With the same way, we can get $max\{h(-\epsilon_0), h(\epsilon_0)\} \leq 2\epsilon_0^2$. So, $E_{a_i\ p_i}[ln\frac{p_i(a_i)}{q_i(a_i)}] \leq 2\epsilon_0^2$.

For $a_1, ...a_k$ are i.i.d. and $|f(a_1, ...a_i, ..., a_k) - f(a_1, ...a_i', ..., a_k)| = |ln\frac{p_i(a_i)}{q_i(a_i)} - ln\frac{p_i(a_i')}{q_i(a_i')}| \leq 2\epsilon_0$, with Lemma 4.4, we have:

$$Pr[f(a_1, ...a_i, ..., a_k) \geq \epsilon' + 2k\epsilon_0^2] \leq Pr[f(a_1, ...a_i, ..., a_k) - E[f(a_1, ...a_i, ..., a_k)] \geq \epsilon'] \leq exp(\frac{-\epsilon'^2}{2m\epsilon_0^2})$$

Let $\delta = exp(\frac{-\epsilon'^2}{2k\epsilon_0^2})$, we have:

$$\epsilon' = \sqrt{2kln(1/\delta)}\epsilon_0$$

Therefore:

$$Pr[f(a_1, ...a_i, ..., a_k) \geq \sqrt{2kln(1/\delta)}\epsilon_0 + 2k\epsilon_0^2] \leq \delta$$

■

13.3 Multiplicative Weight Updating Mechanism

Accuracy and privacy in the interactive setting. Formally, an *interactive mechanism* $M(x)$ is a stateful randomized algorithm which holds a histogram $x \in R^N$. It receives successive counting queries $f_1, f_2, \cdots \in \mathcal{F}$ one by one, and in each round t, on query f_t , it outputs a (randomized) answer a_t (a function of the input histogram, the internal state, and the mechanism's coins). For privacy guarantees, we always assume that the queries are given to the mechanism in an adversarial and adaptive fashion by a randomized algorithm A called the *adversary*. For accuracy guarantees, while we usually consider adaptive adversarial, we will also consider non-adaptive adversarial queries chosen in advance–we still consider such a mechanism to be interactive, because it does not know in advance what these queries will be. The main query class we consider throughout this work is the class $\mathcal{F}$ of all counting queries, as well as sub-classes of it.

For the interactive setting, there is a privacy-preserving interactive mechanism which is called private multiplicative weight updating (PMW) mechanism.

Parameters: Data universe $\mathcal{X}$ with $|\mathcal{X}| = N$. D is a subset of $\mathcal{X}$ with $|D| = n$. The number of queries k, $(\epsilon, \delta) - dp, (\alpha, \beta) - acc$. Put $\sigma = \frac{10 \cdot \log(1/\delta) \log^{1/4} N}{\sqrt{n}\epsilon}, \eta = \frac{\log^{1/4} N}{\sqrt{n}}, T = 4\sigma \cdot (\log k + \log 1/\beta)$.
Input: Database $D \in \mathcal{X}^n$ corresponding to a histogram $x \in \mathbb{R}^N$
Algorithm:
Initialize: Set $x_0 = (1/N, \cdots, 1/N), m = 0$.
For $t = 1, 2, \cdots, k$:

1. Receive f_t
2. $d_t \leftarrow < f_t, x > - < f_t, x_{t-1} >$
3. $\hat{d}_t \leftarrow d_t + A_t, A_t \sim Lap(\sigma)$
4. If $\hat{d}_t > T$: For all i *s.t.* $f_t[i] = 1, x_t[i] \leftarrow x_{t-1}[i]e^{\eta}$; otherwise, $x_t[i] \leftarrow x_{t-1}[i]$.
   Else if $\hat{d}_t < -T$: For all i: $f_t[i] = 0$,$x_t[i] \leftarrow x_{t-1}[i]e^{\eta}$; otherwise, $x_t[i] \leftarrow x_{t-1}[i]$.
   Else $x_t \leftarrow x_{t-1}$.
   Normalize, $x_t[i] \leftarrow \frac{x_t[i]}{\sum_{i \in \mathcal{X}} x_t[i]}$.Update $m \leftarrow m + 1$.
5. If $m > n\sqrt{\log N}$: abort and output 'failure'.
   Else if $|\hat{d}_t| > T$, output the noisy answer $< f_t, x > + A_t$. Else, output $< f_t, x_{t-1} >$.

In the t-th round, after the t-th query f_t has been specified, we compute a noisy answer bat by adding (properly scaled) Laplace noise to $f_t(x)$—the "true" answer on the real database. We then compare this noisy answer with the answer given by the previous round's database $f_t(x_{t-1})$. If the answers are "close", then this is a "lazy" round, and we simply output $f_t(x_{t-1})$ and set $x_t \leftarrow x_{t-1}$. If the answers are "far", then this is an "update" round and we need to update or "improve" x_t using a multiplicative weights re-weighting. The intuition is that the re-weighting brings x_t "closer" to an accurate answer on f_t. In a nutshell, this is all the algorithm does. The only additional step required is bounding the number of "update" rounds: if the total number of update rounds grows to be larger than (roughly) n, then the mechanism fails and terminates. This will be a low probability event.

We use a_t to denote the true answer on the database on query t, and $\hat{a_t}$ denotes this same answer with noise added to it. We use d_t to denote the difference between the true answer a_t and the answer given by x_{t-1}, and $\hat{d_t}$ to denote the difference between the *noisy* answer and the answer given by x_{t-1}. The boolean variable w_t denotes whether the noisy difference was large or small. If $\hat{d_t}$ is smaller (in absolute value) than $\approx 1/\sqrt{n}$, then this round is lazy. If $\hat{d_t}$ is larger than threshold then this is an *update* round and we set $m \leftarrow m + 1$.

Definition 13.6 *We say that a mechanism M is $(\alpha, \beta, k)-$(adaptively) accurate for a database x, if when it is run for k rounds, for any (adaptively chosen) counting queries, with all but β probability over the mechanism's coins $\forall t \in [k], |a_t - f_t(x)| \leq \alpha$*

Theorem 13.7 *Let X be a data universe of size N. For any $k, \varepsilon, \delta, \beta > 0$, the Private Multiplicative Weights Mechanism above is an $(\epsilon, \delta)-$differentially private interactive mechanism. For any database of size n, the mechanism is $(\alpha, \beta, k)-$accurate for (adaptive) counting queries over X, where*

$$\alpha = O(\varepsilon^{-1} n^{-1/2} \cdot \log(1/\delta) \log^{1/4}(N) \cdot (\log k + \log(1/\beta)))$$

The running time in answering each query is $N \cdot poly(n) \cdot polylog(1/\beta, 1/\varepsilon, 1/\delta)$.

The proof of the theorem can be found in [6].

13.4 Generalization of Boosting

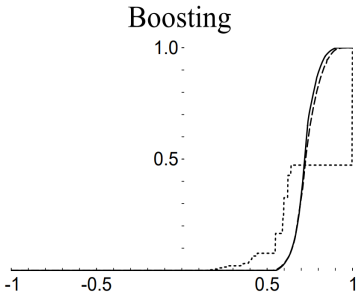
In this section, we will discuss the generalization of boosting algorithm based on the margin theory.

We have known that boosting works by voting over a set of base hypotheses, where the result hypothesis as a weighted combination of base hypotheses. As long as the base hypotheses are not so bad, the training error will exponentially decay to zero with respect to the boosting iterations. Considering the test/generalization error of boosting, it is surprising to find out that after the training error is reduced to zero, the test error continues to go down with additional training. This fact is counter-intuitive because most of the time, we assume that the 'over-fitting' on training data will result in bad performance of generalization.

To explain this fact, the main observation is that boosting continues to enlarge the *margin* even after the training error is reduced to zero. Consider a binary classification task with sample space $\mathcal{X} \times \{1, -1\}$, distribution $\mathcal{D}$ on $\mathcal{X} \times \{1, -1\}$, hypothesis space $\mathcal{H}$, the voted classifier given by boosting $f(x) = \Sigma_{i=1}^{N} \alpha_i h_i(x)$, where the weights αs are normalized to have sum 1. We define the *margin* of a sample $(x, y) \in \mathcal{X} \times \{1, -1\}$ as $yf(x)$. It is easy to see that *margin* is a number in $[-1, 1]$, and that a sample is classified correctly if and only if it has positive *margin*. Moreover, a large *margin* can be interpreted a "confident" classification.

To visualize the observation, we plot the fraction of examples whose margin is at most θ as a function of $\theta \in [-1, 1]$, we refer to these graph as *margin distribution graph*. In experiments, we observe that as the

training progresses, the boosting algorithm tends to increase the margins associated with examples and converge to a margin distribution in which most examples has large margins. The graph below is a demo of *margin distribution graph* of a binary classification, where the short-dashed, long-dashed and solid curves correspond to the margin distribution after 5, 100, 1000 iterations.



Now, we formalize the bound of generalization error of boosting by giving the following theorem given by Robert E. Schapire and Yoav Freund in [7].

Theorem 13.8 *Let S be a sample of n examples chosen independently at random according to $\mathcal{D}$. Assume that the base hypothesis space $\mathcal{H}$ is finite, and let $\delta > 0$. Then with probability at least $1 - \delta$ over random choice of the training set S, every voted classifier f satisfies the following bound for all $\theta > 0$:*

$$\boldsymbol{P}_{\mathcal{D}}[yf(x) \leq 0] \leq \boldsymbol{P}_S[yf(x) \leq \theta] + O(\frac{1}{\sqrt{n}}(\frac{log\ n\ log\ |\mathcal{H}|}{\theta^2} + log(1/\delta))^{1/2})$$

This theorem states that if the voting classifier generates a good margin distribution, that is, most training examples have large marins so that $P_S(yf(x) \leq \theta)$ is small for not too small θ, then the generalization error is also small. In [7], it has also been shown that for the AdaBoost algorithm $P_S(yf(x) \leq \theta)$ decreases to zero exponentially fast with respect to the number of boosting iterations if θ is not too large. These results imply that the excellent performance of AdaBoost is due to its good margin distribution.

Breiman[8] also proved an upper bound for the generalization error of voting classifiers. This bound depends only on the minimum margin, not on the entire margin distribution.

Theorem 13.9 *Let θ_0 be the minimum margin defined as*

$$\theta_0 = min\{yf(x) : (x, y) \in S\}$$

, where S is the training set. If

$$\mathcal{H} < \infty,\ \theta_0 > 4\sqrt{\frac{2}{|\mathcal{H}|}},\ R = \frac{32log(2|\mathcal{H}|)}{n\theta_0^2} \leq 2n$$

then for any $\delta > 0$, with probability at least $1 - \delta$ over the random choice of the training set S of n examples, every voting classifier f satisfies the following bounds:

$$\boldsymbol{P}_D(yf(x) \leq 0) \leq R(log(2n) + log\frac{1}{R} + 1) + \frac{1}{n}log(\frac{|\mathcal{H}|}{\delta})$$

Breiman pointed out that this bound is sharper that that of Schapire's, as the second term of this bound is $O(\frac{log\ n}{n})$ while that in Schapire's is $O(\sqrt{\frac{log\ n}{n}})$.

References

[1] Erlingsson U, Pihur V, Korolova A. Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. pp. 1054–1067. ACM (2014).

[2] Warner S.L. Randomized response: A survey technique for eliminating evasive answer bias. Journal of the American Statistical Association 60(309), 63–69 (1965).

[3] Apple Inc. Differential Privacy Team: Learning with privacy at scale (2017).

[4] Ding B, Kulkarni J, Yekhanin S. Collecting telemetry data privately. In: Advances in Neural Information Processing Systems. pp. 3571–3580 (2017).

[5] Bebensee B. Local differential privacy: a tutorial[J]. arXiv preprint arXiv:1907.11908, 2019.

[6] Hardt M, Rothblum G.N. A multiplicative weights mechanism for privacy-preserving data analysis. in Proc. Ann. IEEE Symp. Found. Comput. Sci. (FOCS), Las Vegas, NV, 2010, pp. 61-70.

[7] Bartlett P, Freund Y, Lee W S, et al. Boosting the margin: A new explanation for the effectiveness of voting methods[J]. The annals of statistics, 1998, 26(5): 1651-1686.

[8] Breiman L. Prediction games and arcing algorithms[J]. Neural computation, 1999, 11(7): 1493-1517.

Machine Learning **Spring 2021**

Lecture 14: Reinforcement Learning with Known Model

Lecturer: Liwei Wang *Scribe: Jingrui Xing, Qizhi Chen, Weijie Peng, Hao Liu, Yuheng Wu*

14.1 Reinforcement Learning: Basic Concepts

Simply put, **reinforcement learning** is the process of training models to take a sequence of decisions in an environment that optimize the reward, usually in large-scale problems where dynamic programming is inapplicable. Virtually all RL problems can be formalized as **Markov Decision Process** (MDP).

14.1.1 Markov Decision Process

Definition 14.1 *A Markov Decision Process is a 4-tuple (S, A, P, R), where*

- *S is the set of states;*
- *$A(s)$ is the set of available actions from state $s \in S$;*
- *$P(s'|s, a) = \mathbb{P}(S_{t+1} = s'|S_t = s, a_t = a)$ is the transition probability that, in time t, the action $a \in A$ in state s would lead to state s' in time $t + 1$;*
- *$R(s, a)$ is the **expected** reward received by taking action a at state s;*

In the case when the decision-making process has infinite horizon, this tuple also includes a discouting factor $\gamma \in (0, 1)$. For finite horizon, $\gamma = 1$.

The goal of an MDP agent is to maximize the **long term reward**, defined as $R \triangleq R(s_{t+1}, a_{t+1}) + \gamma R(s_{t+2}, a_{t+2}) + \gamma^2 R(s_{t+3}, a_{t+3}) + \cdots$. Yet, in most cases, instead of directly dealing with discrete set of actions, we are dealing with policies.

A **policy** is defined as $\pi : S \to \Delta(A)$, where $\pi(s)$ gives the probability vector of choosing actions in A at state s. For short, we write $\pi(s)$ for a random variable of taking actions with the corresponding probabilities.

There are two tasks in an MDP control problem: how to evaluate a given policy $\pi : S \to A$; how to find the optimal policy π^*. The first task involves two **value functions**. The second task would be discussed in the following chapters.

14.1.2 State Value Function

Definition 14.2 *For a given policy π, at a certain state s, the state value function is defined as*

$$v^\pi(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \Big| s_0 = s\right]$$

This is the expected return when starting from s and following policy π. The randomness comes from the random action distribution and the transition probability.

14.1.3 State-Action Value Function

Definition 14.3 *For a certain state s, a feasible action a in state s, and given policy π, the state-action value function is defined as*

$$Q^{\pi}(s,a) = \mathbb{E}\left[R(s,a) + \sum_{t=1}^{\infty} \gamma^t R(s_t, \pi(s_t)) \Big| s_0 = s, a_0 = a\right]$$

This is the expected return when starting from s, taking action a, then following policy π.

Note that a does not necessarily follows π. Moreover, $Q^{\pi}(s,a) = \mathbb{E}_{s_1}\left[R(s,a) + \gamma V^{\pi}(s_1,a_1) \big| s_0 = s, a_0 = a\right]$. Therefore, if a follows π, then $\mathbb{E}[Q^{\pi}(s,a)] = V^{\pi}(s)$.

14.1.4 Optimal Policy

With the definition of the two value functions, we now give a formal definition of the optimal policy.

Definition 14.4 *A optimal policy π^* satisfies: for any policy π and any state s, it holds $V^{\pi^*}(s) \geq V^{\pi}(s)$.*

In the following chapters, we will discuss the existence of optimal policy and its computation.

14.2 Bellman Expectation Equation and Policy Evaluation

14.2.1 Bellman Expectation Equation

From the definition of the value function, we can generally see that it can be expressed in the form of a transition function. More specifically,

$$v^{\pi}(s) = \mathrm{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid s_t = s\right] \tag{14.1}$$

$$= \mathrm{E}\left[R_t + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \mid s_t = s\right] \tag{14.2}$$

$$= \mathrm{E}\left[R_t \mid s_t = s\right] + \gamma \mathrm{E}_{s_{t+1} \sim P(s,\pi(s))}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k}\right] \tag{14.3}$$

$$= R(s,\pi(s)) + \gamma \sum_{s'} Pr[s_{t+1} = s' | s_t = s, a_t = \pi(s)] \cdot v^{\pi}(s') \tag{14.4}$$

where R_t represents reward at time t (which depends on the state and action at time t), and $R(s,a)$ represents the expectation of reward on state s and action a. Thus, we have

$$v^{\pi}(s) = R(s,\pi(s)) + \gamma \sum_{s'} P(s'|s,\pi(s)) \cdot v^{\pi}(s'), \forall s \in S \tag{14.5}$$

which is known as **Bellman Expectation Equation**.

14.2.2 Evaluating the value function

Firstly, we define an operator according to Bellman Expectation Equation.

Definition 14.5 (Bellman Expectation Operator) *The Bellman Expectation Operator Φ is defined as*

$$\begin{aligned}\Phi : \mathbb{R}^N &\to \mathbb{R}^N \\ \mathbf{v} &\mapsto \mathbf{v}'\end{aligned}$$

such that

$$\mathbf{v}'(s) = R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) \cdot \mathbf{v}(s'), \forall s \in S \tag{14.6}$$

where $N = |S|$.

To further discuss the property of Φ, we introduce what a γ-contraction mapping is.

Definition 14.6 (α-contraction mapping) *Function $\phi : X \to X$ is a α-contraction mapping w.r.t. $||\cdot||_p$, if $\forall u, v \in X$,*

$$||\phi(u), \phi(v)||_p < \alpha ||u, v||_p \tag{14.7}$$

where $\alpha \in [0, 1)$

Theorem 14.7 (Banach Fixed Point Theorem) *Every contraction mapping has a unique fixed point.*

Now we point out that Φ is a γ-contraction mapping w.r.t. $||\cdot||_\infty$.

Theorem 14.8 *Bellman Expectation Operator Φ is a γ-contracion mapping w.r.t. $||\cdot||_\infty$.*

Proof: $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^N$, let $\mathbf{u}' = \Phi(\mathbf{u}), \mathbf{v}' = \Phi(\mathbf{v})$. We have $\mathbf{u}' - \mathbf{v}' = \gamma \sum_{s'} P(s'|s, \pi(s)) \cdot (\mathbf{u}(s') - \mathbf{v}(s')) \leq \gamma \sum_{s'} P(s'|s, \pi(s)) \cdot ||\mathbf{u} - \mathbf{v}||_\infty = \gamma ||\mathbf{u} - \mathbf{v}||_\infty$. ■

Now we know that Φ converge to a unique point. Thus, we can use the following algorithm to evaluate the value function.

Algorithm 1 Solving Bellman Expectation Equation via Iteration

Output: The value function vector $\mathbf{v}^\pi$ w.r.t. a given policy π
  Stochastically initialize $\mathbf{v}_0$
  $k \leftarrow 0$
  repeat
    $\mathbf{v}_{k+1} \leftarrow \Phi(\mathbf{v}_k)$
    $k \leftarrow k + 1$
  until Convergence. Denote the result as $\mathbf{v}^\pi$

14.2.3 Greedy Method

Now given a policy π, we can get $\mathbf{v}^{\pi}$ through the above algorithm. Then in order to get an improved policy π', we employ a greedy method:

$$\forall s \in S, \pi'(s) = \underset{a}{\operatorname{argmax}} \left[R(S,a) + \sum_{s'} P\left(s' \mid s, a\right) \mathbf{v}^{\pi}\left(s'\right) \right]$$

Theorem 14.9 $\mathbf{v}^{\pi}(s) \leq \mathbf{v}^{\pi'}(s)$ *for all* s

Proof:

$$\begin{aligned} \mathbf{v}^{\pi}(s) &= R(s,\pi(s)) + \gamma \sum_{s'} P\left(s' \mid s, \pi(s)\right) \mathbf{v}^{\pi}\left(s'\right) \\ &\leq R\left(s,\pi'(s)\right) + \gamma \sum_{s'} P\left(s' \mid s, \pi'(s)\right) \mathbf{v}^{\pi}\left(s'\right) \end{aligned}$$

Written in matrix form as follows:

$$\mathbf{R}_{\pi'} + \gamma \mathbf{P}_{\pi'} \mathbf{v}^{\pi} \geq \mathbf{R}_{\pi} + \gamma \mathbf{P}_{\pi} \mathbf{v}^{\pi} = \mathbf{v}^{\pi}$$

$$\mathbf{R}_{\pi'} \geq \left(\mathbf{I} - \gamma \mathbf{P}_{\pi'}\right) \mathbf{v}^{\pi}$$

And we have:

$$\|\mathbf{P}\|_{\infty} = \max_{s} \sum_{s'} |\mathbf{P}_{ss'}| = \max_{s} \sum_{s'} \mathbb{P}\left[s' \mid s, \pi(s)\right] = 1$$

Then $\|\gamma \mathbf{P}\|_{\infty} = \gamma < 1$. Since the radius of convergence of the power series $(1-x)^{-1}$ is 1, we can use its expansion and write

$$\left(\mathbf{I} - \gamma \mathbf{P}_{\pi'}\right)^{-1} = \sum_{k=0}^{\infty} \left(\gamma \mathbf{P}_{\pi'}\right)^{k}.$$

Thus, if $\mathbf{Z} = (\mathbf{Y} - \mathbf{X}) \geq \mathbf{0}$, then $\left(\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}}\right)^{-1} \mathbf{Z} = \sum_{k=0}^{\infty} \left(\gamma \mathbf{P}_{\pi_{n+1}}\right)^{k} \mathbf{Z} \geq \mathbf{0}$, which means $\left(\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}}\right)^{-1}$ preserves ordering. Then $\mathbf{v}^{\pi'} = \left(\mathbf{I} - \gamma \mathbf{P}_{\pi_{n+1}}\right)^{-1} \mathbf{R}_{\pi_{n+1}} \geq \mathbf{v}^{\pi}$ ■

Now our goal is to find out if there exists an unique optimal policy π^*, which has the property that: $\forall s, \mathbf{v}^{\pi^*}(s) \geq \mathbf{v}^{\pi}(s)$

14.3 Finding the Optimal Policy

In the above section we've discussed about the value function. Now, let's turn to the problem of finding the optimal policy.

We've already known the definition of an optimal policy. Now we assume such policy exists, denoted as π^*.

Definition 14.10 (Policy Iteration) *Using the Greedy Method above, we can obtain a method of iteration: start from some policy* π_0*, and compute* v^{π_0}*, get the "greedy policy" of* π_0 *(denoted as* π_1 *), and compute* V^{π_1}*; ... We know the value function of* π_k *should increase during the iteration. This approach is called policy iteration, the value function of* π_k *will converge to* v^**.*

And based on Bellman Expectation Equation, we may guess that,

$$v^{\pi^*}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) v^{\pi^*}(s') \right], \forall s \in S \tag{14.8}$$

This is called the **Bellman Optimal Equation**. Similarly, we define an op operator according to this equation.

Definition 14.11 (Bellman Operator) *The Bellman Operator Φ^* is defined as*

$$\Phi^* : \mathbb{R}^N \to \mathbb{R}^N$$
$$\mathbf{v} \mapsto \mathbf{v}'$$

such that

$$\mathbf{v}'(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \mathbf{v}(s') \right], \forall s \in S \tag{14.9}$$

where $N = |S|$.

We point out that Φ^* is also a contraction mapping.

Theorem 14.12 *Bellman Operator Φ^* is a γ-contraction mapping w.r.t. $||\cdot||_\infty$.*

Proof: $\forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^N$, let $\mathbf{u}' = \Phi^*(\mathbf{u}), \mathbf{v}' = \Phi^*(\mathbf{v})$. For a state $s \in S$, let

$$a = \operatorname*{argmax}_b \left[R(s,b) + \gamma \sum_{s'} P(s'|s,b) \mathbf{v}(s') \right]$$
$$a' = \operatorname*{argmax}_b \left[R(s,b) + \gamma \sum_{s'} P(s'|s,b) \mathbf{u}(s') \right]$$

We have

$$\begin{aligned}
\mathbf{v}'(s) - \mathbf{u}'(s) &= \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \mathbf{v}(s') \right] - \left[R(s,a') + \gamma \sum_{s'} P(s'|s,a') \mathbf{u}(s') \right] \\
&\leq \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \mathbf{v}(s') \right] - \left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \mathbf{u}(s') \right] \\
&= \gamma \sum_{s'} P(s'|s,a)(\mathbf{v}(s') - \mathbf{u}(s')) \\
&\leq \gamma \sum_{s'} P(s'|s,a) ||\mathbf{v} - \mathbf{u}||_\infty \\
&= \gamma ||\mathbf{v} - \mathbf{u}||_\infty
\end{aligned}$$

Similarly we show that $\mathbf{u}'(s) - \mathbf{v}'(s) \leq \gamma ||\mathbf{v} - \mathbf{u}||_\infty$. Thus, we have $||\mathbf{v}' - \mathbf{u}'||_\infty \leq \gamma ||\mathbf{v} - \mathbf{u}||_\infty$. ■

Now we know that Φ^* converge to a unique fixed point. So similar to Algorithm 1, we can use a iteration approach to compute the unique fixed point v^* of Φ^*

We have following results.

Algorithm 2 Value Iteration

Output: The unique fixed point $\mathbf{v}^*$ of Φ^*
  Stochastically choose a policy π_0 ,let $\mathbf{v}_0$ be its value function
  $k \leftarrow 0$
  repeat
    $\mathbf{v}_{k+1} \leftarrow \Phi^*(\mathbf{v}_k)$
    $k \leftarrow k+1$
  until Convergence. Denote the result as $\mathbf{v}^*$

Theorem 14.13 *Let π_0 be an arbitrary policy and $v_0 = v^{\pi_0}$. Then $(\Phi^*(v_0))(s) \geq v_0(s), \forall s \in S$*

Proof: Note that for all $s \in S$,

$$v_0(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) v_0(s')$$

by the Bellman Expectation Equation, and

$$(\Phi^*(v_0))(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_0(s')]$$

by the definition of Φ^*.

Comparing the right sides immediately yields $(\Phi^*(v_0))(s) \geq v_0(s)$ ■

Theorem 14.14 *Let π_0 be an arbitrary policy and $v_0 = v^{\pi_0}$ and $v_{k+1} = \Phi^*(v_k)$. Then $v_{k+1}(s) \geq v_k(s), \forall s \in S$*

Proof: we can prove this by induction since we already have $(\Phi^*(v_0))(s) \geq v_0(s), \forall s \in S$

suppose we have $v_k(s) \geq v_{k-1}(s), \forall s \in S$

$$v_{k+1} = (\Phi^*(v_k))(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_k(s')]$$

$$v_k = (\Phi^*(v_{k-1}))(s) = \max_{a \in A} [R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) v_{k-1}(s')]$$

Comparing the right sides and it's trivial that $v_{k+1}(s) \geq v_k(s)$ ■

And the following fact is trivial.

Theorem 14.15 *Let π_0 be an arbitrary policy and $v_0 = v^{\pi_0}$ and $v_{k+1} = \Phi^*(v_k)$. Then v_k converges to v^* which satisfies the Bellman Optimal Equation.*

Theorem 14.16 *Let v^* is the fixed point of Φ^*. Then there exists a policy π^* whose value function is v^*.*

Proof: let $\pi^*(s) := \text{argmax}_a [R(s,a) + \gamma \sum_{s'} P(s'|s,a) v^*(s')]$

$$\begin{aligned} v_{\pi^*}(s) &= R(s,\pi^*(s)) + \gamma \sum_{s'} P(s'|s,\pi^*(s)) v_{\pi^*}(s') \\ &= R(s,\pi^*(s)) + \gamma \sum_{s'} P(s'|s,\pi^*(s))[R(s',\pi^*(s')) + \gamma \sum_{s''} P(s''|s',\pi^*(s')) v_{\pi^*}(s'')] \\ &= \cdots \\ &= v^*(s) \end{aligned}$$

■

Now we have proved the existence of π^* and gives a way to calculate it.And finally we have:

Theorem 14.17 *Let v^* be the value function of the optimal policy, assume $||v_k - v^*||_\infty \leq \delta$. Let π_k be the greedy policy w.r.t. v_k, then $||v^{\pi_k} - v^*||_\infty \leq \frac{\gamma}{1-\gamma}\delta$.*

Proof: According to the assumption $||v_k - v^*||_\infty \leq \delta$ and that the greedy strategy is to choose $\pi_k(s) := \text{argmax}_a[R(s,a) + \gamma \sum_{s'} P(s'|s,a) v_k(s')]$, so we know that if we choose a wrong action due to the $\delta-$difference at state s, time t, and follow the best strategy since then, we would only be $\gamma\delta$ less than the best strategy in expectation.

By the same argument, if we make two mistakes successively, then our expectation value drops at most $\gamma\delta + \gamma^2\delta$

So we get $||v^{\pi_k} - v^*||_\infty \leq (\gamma + \gamma^2 + \cdots)\delta = \frac{\gamma}{1-\gamma}\delta$ ■

References

Machine Learning **Spring 2021**

Lecture 15: Reinforcement Learning

Lecturer: Liwei Wang *Scribe: Mingwei Yang, Hongyu Yan*

15.1 Monte Carlo Decision Process(MDP)

Monte Carlo methods are ways of solving the reinforcement learning problem based on averaging sample returns. Here we define Monte Carlo methods only for episodic tasks. For all $s \in S$, there are two Monte Carlo (MC) methods, *first-visit MC* method and *every-visit MC* method.

First-visit MC: Estimate the value of a state as the average of the returns that have followed the first visits to the state, where a first visit is the first time during a trial that the state is visited.
Every-visit MC: Estimate the value of a state as the average of the returns that have followed all visits to the state.

It can be trivially implied that the *first-visit MC* method is a sound method, since the samples are of independently identically distributed estimate of $v^{\pi}(s)$ with finite variance. By the law of large numbers, the sequence of averages of these estimates converges to their expected value. Each average is itself an unbiased estimate, and the standard deviation of its error falls as $\frac{1}{\sqrt{N}}$, where N is the number of returns averaged. *Every-visit MC* is less straightforward. Surprisingly, however, the *every-visit MC* method is also a sound method.

Lemma 15.1 *Suppose $v_1, v_2 \cdots$ are random variables and have common mean. Let N take positive integers. Suppose $\mathbb{E}[v_j|N \geq j] = \mathbb{E}[v_1]$. Then*

$$\frac{\mathbb{E}\left[\sum_{j=1}^{N} v_j\right]}{\mathbb{E}[N]} = \mathbb{E}[v_1]$$

Proof:

$$\mathop{\mathbb{E}}_{N,v}\left[\sum_{j=1}^{N} v_j\right] = \sum_{n=1}^{\infty} \Pr[N=n]\mathbb{E}\left[\sum_{j=1}^{n} v_j | N=n\right] \tag{15.1}$$

$$= \sum_{n=1}^{\infty}\sum_{j=1}^{n} \Pr[N=n]\mathbb{E}[v_j|N=n] \tag{15.2}$$

$$= \sum_{j=1}^{\infty} \Pr[N \geq j]\mathbb{E}[v_j|N=j] \tag{15.3}$$

$$= \mathbb{E}[N] \cdot \mathbb{E}[v_j] \tag{15.4}$$

■

Theorem 15.2 *The* Every-visit MC *method is a sound method for policy evaluation.*

Proof: For trajectories $k = 1, 2 \cdots K$, let $v_{k,j}$ be the return of state s of k-th trajectory for the j-th visit. Then the *Every-visit MC* method has the estimation

$$estimation = \frac{\sum_{k=1}^{K} \sum_j v_{k,j}}{\sum_{k=1}^{K} N(k)}$$

where $N(k) = \#$ visits of s in trajectory k.
Let's define E as the limit of estimation, e.g.,

$$E := \lim_{K\to\infty} \frac{\sum_{k=1}^{K} \sum_j v_{k,j}}{\sum_{k=1}^{K} N(K)} = \lim_{K\to\infty} \frac{\frac{1}{m(K)} \sum_{k=1}^{K} \sum_j v_{k,j}}{\frac{1}{m(K)} \sum_{k=1}^{K} N(k)} = \frac{\lim_{K\to\infty} \frac{1}{m(K)} \sum_{k=1}^{K} \sum_j v_{k,j}}{\lim_{K\to\infty} \frac{1}{m(K)} \sum_{k=1}^{K} N(k)}$$

where $m(K) = \#$ trajectories in which s is visited.
To prove the method is a sound method, it is sufficient to prove

$$E = v^{\pi}(s)$$

Let N be the (random) number of visits of s in an trajectory, v_j be the (random) return of s for the j-th visit. Clearly, $v_1, v_2 \cdots$ have common mean. According to **lemma 15.1**

$$E = \frac{\mathbb{E}\left[\sum_{j=1}^{N} v_j | N \geq 1\right]}{\mathbb{E}[N|N \geq 1]} = \mathbb{E}[v_1|N \geq 1] = \mathbb{E}[v_1] = v^{\pi}(s)$$

■

Suppose our state sequence in a trajectory is $S_1, S_2, \cdots, S_t \cdots$. We need to check if S_t is not in the sequence $S_1, S_2, \cdots, S_{t-1}$ in *First-visit MC* method while we don't need to check it in *Every-visit MC* method. In practice, *Every-visit MC* is more data efficient.

15.2 Temporal-Difference Learning (TD-Learning)

15.2.1 TD(0) Algorithm

Here we present TD(0) algorithm, for evaluating a policy in the case where the environment model is unknown. The algorithm is based on Bellman's equations giving the value of a policy π:

$$\begin{aligned} v^{\pi}(s) &= \mathbb{E}[R(s,a)] + \gamma \sum_{s'} \Pr[s'|s, \pi(s)] v^{\pi}(s') \\ &= \mathbb{E}\left[R(s, \pi(s)) + \gamma \cdot v^{\pi}(s')|s\right] \end{aligned}$$

Suppose during the sampling process, we sample a new state s_{t+1} and get reward R_t when taking a action according to π in state s_t. The TD(0) algorithm updates the policy values according to the following:

$$\begin{aligned} v(s_t) &\leftarrow (1 - \alpha_t) v(s_t) + \alpha_t \left[R_t + \gamma \cdot v(s_{t+1})\right] \\ &= v(s_t) + \alpha_t \left[R_t + \gamma \cdot v(s_{t+1}) - v(s_t)\right] \end{aligned}$$

where $\alpha_t \in (0, 1)$. The term $[R_t + \gamma \cdot v(s_{t+1}) - v(s_t)]$ is called the temporal difference of v values, which justifies the name of the algorithm.

Theorem 15.3 *Assume that* $\sum_{t=0}^{\infty} \alpha_t = \infty$, *and* $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$. *Then, the TD(0) algorithm converges to* $v(s)$.

15.2.2 TD(λ) Algorithm

The idea of TD(λ) consists of using multiple steps ahead, where TD(0) only uses one step ahead. Thus, for $n > 1$ steps, we would have the update

$$v(s_t) \leftarrow v(s_t) + \alpha_t(G_t^{(n)} - v(s_t))$$

where $G_t^{(n)}$ is defined by

$$G_t^{(n)} = R_t + \gamma R_{t+1} + \cdots + \gamma^{n-1} R_{t+n-1} + \gamma^n v(s_{t+n})$$

Comparing to TD(0), the normal defined TD(n) uses more information in each iteration. However it has longer delays, e.g., the current state would be updated only after n steps. Moreover, we could only update state once a time. But TD(λ) beautifully solves those problems by using G_t^{λ} instead of $G_t^{(n)}$, where

$$G_t^{\lambda} = (1-\lambda) \sum_{n=0}^{\infty} \lambda^n G_t^{(n)}, \quad \lambda \in [0,1]$$

is based on the geometric distribution over all $G_t^{(n)}$. Thus, the main update becomes

$$v(s_t) \leftarrow v(s_t) + \alpha_t(G_t^{\lambda} - v(s_t))$$

and we get the main steps of TD(λ) algorithm:

1. For all $s \in S$, $E_0(s) = 0$, $E_t(s) = \gamma\lambda E_{t-1}(s) + I[s_t = s]$.
2. $\delta_t \leftarrow R_t + \gamma \cdot v(s_{t+1}) - v(s_t)$.
3. For all $s \in S$, $v(s) \leftarrow v(s) + \alpha_t \cdot \delta_t E_t(s)$.

15.3 Q-learning

To find the optimal policy in the unknown model, e.g., the transition and reward probabilities are unknown, we can't directly apply iteration based on the Bellman Optimal Equation. Note that the optimal policy or policy value can be straightforwardly derived from state-action value function Q via: $\pi^*(s) = \arg\max_{a \in A} Q(s,a)$ and $v^*(s) = \max_{a \in A} Q(s,a)$.

The Q-learning algorithm is based on the equations giving the optimal state-action function Q^*:

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}[R(s,a)] + \gamma \sum_{s' \in S} \Pr[s'|s,a] v^*(s') \\ &= \mathbb{E}_{s'} \left[R(s,a) + \gamma \max_{a \in A} Q^*(s',a) \right] \end{aligned}$$

Similar to TD-learning, we update $Q(s, a)$ with real experience as follows:

$$\begin{aligned} Q(s_t, a_t) &\leftarrow (1 - \alpha_t)Q(s_t, a_t) + \alpha_t \left[R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a')\right] \\ &= Q(s_t, a_t) + \alpha_t \left[R(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \end{aligned}$$

It's remain to specify how to choose actions. Note that random selection is not a good idea, since it doesn't update the current best Q_t frequently, especially when there are a lot of actions. On the other hand, a bad initial value can invalidate the "Choose only the current $\arg\max_a Q_t(s, a)$" method, which can be viewed as keeping doing exploitation but ignoring exploration.

15.3.1 Action Selection Policy

To select actions, a standard choice in reinforcement learning is the so called ϵ-*greedy policy*, which consists of selecting with probability $(1 - \epsilon)$ the greedy action from state s, that is, $\arg\max_{a \in A} Q_t(s, a)$, and with probability ϵ a random action from s, for some $\epsilon \in (0, 1)$. Thus method ensures that if enough trials are done, each action will be tried an infinite number of times, thus ensuring optimal action are discoverd.

We generalize the different types of the action selection policies into on-policy learning and off-policy learning.

15.4 On-Policy and Off-Policy Learning

Reinforcement learning algorithms include two components: a *learning policy*, which determines the action to take, and an *update rule*, which defines the new estimate of the optimal value function.

For an *off-policy learning algorithm*, the update rule does not necessarily depend on the learning policy. Q-learning is an off-policy algorithm since its update rule is based on the max operator and the comparison of all possible actions a', yet its learning policy is based on the ϵ-greedy policy.

On-Policy learning algorithms are the algorithms that evaluate and improve the same policy which is being used to select actions. Some examples of On-Policy algorithms are Policy Iteration, Value Iteration and Monte Carlo for On-policy.

References

[1] Mehryar Mohri and Afshin Rostamizadeh and Ameet Talwalkar. The Foundation of Machine Learning.

[2] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction.