Supervised Learning

Phu Sakulwongtana

1 Introduction to Machine Learning Problem

Definition 1.1. (Machine Learning Problem) Define input space and output space $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y}$, respectively. Given the training data points:

$$\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)\} \subset \mathcal{X} \times \mathcal{Y}$$

The goal is to infer the function $f_S(\boldsymbol{x}_i) \approx y_i$, which we can use in the future data. There are 2 types of problems, when: $y \in \{-1, 1\}$, the problem is classification and if $y \in \mathbb{R}$, the problem is regression.

Definition 1.2. (Learning Algorithm) Given the training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m \subset \mathcal{X} \times \mathcal{Y}$. The learning algorithm perform a mapping $S \mapsto f_S$, where the new input can be predicted as $f_S(\boldsymbol{x})$.

Definition 1.3. (Binary Classification) Given the training domain to be $\mathcal{X} = \mathbb{R}^2$ for $\boldsymbol{x} = (x_1, x_2)$ and $\mathcal{Y} = \{0, 1\}$, our predictor is defined as:

$$f(\boldsymbol{x}) = \begin{cases} 0 & \boldsymbol{w}^T\boldsymbol{x} + b > 0 \\ 1 & \boldsymbol{w}^T\boldsymbol{x} + b \le 0 \end{cases}$$

Definition 1.4. (Mean Square Error) In most of the machine learning problem, we would like to find the predictor to minimize the following loss (for m size dataset):

$$\frac{1}{m}\sum_{i=1}^m (y_i - \hat{y}_i)^2 = \frac{1}{m}\sum_{i=1}^m (y_i - \boldsymbol{w}^T\boldsymbol{x})^2$$

This is called mean-square error (MSE).

Lemma 1.1. *Given the input and output dataset, which can be represented in matrix and vector notation:*

$$\boldsymbol{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Given the predictor to be $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$, then the mean-square error can be denoted as:

$$\mathcal{E}_{emp}(S, \boldsymbol{w}) = \frac{1}{m}(\boldsymbol{X}^T\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}^T\boldsymbol{w} - \boldsymbol{y})$$

where the dataset is $S = (\boldsymbol{X}, \boldsymbol{w})$.

Proof. Consider the MSE to be, which we can consider the matrix multiplication:

$$\begin{aligned} \frac{1}{m}\sum_{i=1}^m (y_i - \hat{y}_i)^2 &= \frac{1}{m}\sum_{i=1}^m (y_i - \boldsymbol{w}^T\boldsymbol{x})^2 \\ &= \frac{1}{m}\sum_{i=1}^m \left(y_i - \sum_{j=1}^n w_j x_{ij}\right)^2 \\ &= \frac{1}{m}(\boldsymbol{X}^T\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}^T\boldsymbol{w} - \boldsymbol{y}) \end{aligned}$$

$\square$

Proposition 1.1. *The solution to the mean-square error is given by:*

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Assuming that $\boldsymbol{X}^T\boldsymbol{X}$ is invertible.

Proof. Let's consider the derivative of $\nabla_{\boldsymbol{w}}\mathcal{E}_{\text{emp}}(S, \boldsymbol{w})$, which is given as:

$$\nabla_{\boldsymbol{w}}\Big[(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})\Big] = \boldsymbol{0}$$

$$\iff \left(\sum_{i=1}^{m}\frac{\partial}{\partial w_1}\left(\sum_{j=1}^{n}x_{ij}w_j - y_i\right)^2, \dots, \sum_{i=1}^{m}\frac{\partial}{\partial w_n}\left(\sum_{j=1}^{n}x_{ij}w_j - y_i\right)^2\right)^T = \boldsymbol{0}$$

Let's consider the derivative of each variable w_k

$$\begin{aligned}\frac{\partial \mathcal{E}_{\text{emp}}(S, \boldsymbol{w})}{\partial w_k} &= \frac{2}{m}\sum_{i=1}^{m}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)\frac{\partial}{\partial w_k}\boldsymbol{w}^T\boldsymbol{x}_i \\ &= \frac{2}{m}\sum_{i=1}^{m}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)\boldsymbol{x}_{ik}\end{aligned}$$

Let's consider the simpler case in 2 dimensions with $\boldsymbol{w} = (w_1, w_2)^T$, then setting this to zero gives us:

$$\sum_{i=1}^{m}(x_{ij}x_{i1}w_1 + x_{ik}x_{i2}w_2) = \sum_{i=1}^{m}x_{ik}y_i$$

for $k = 1, 2$. In vector notation, this is equivalent to $\sum_{i=1}^{m}\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{w} = \sum_{i=1}^{m}\boldsymbol{x}_iy_i$ or it is equivalent to matrix notation is: $\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y}$, taking the inverse gives us the required answer. $\square$

Proposition 1.2. *Bias term for the predictor can be added i.e $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b$.*

$$\begin{bmatrix}\boldsymbol{X}^T\boldsymbol{X} & \boldsymbol{X}^T\boldsymbol{1} \\ \boldsymbol{1}^T\boldsymbol{X} & m\end{bmatrix}\begin{bmatrix}\boldsymbol{w} \\ b\end{bmatrix} = \begin{bmatrix}\boldsymbol{x}^T\boldsymbol{y} \\ \boldsymbol{1}^T\boldsymbol{y}\end{bmatrix}$$

For dataset of size m, and $\boldsymbol{1}$ is the vector of elements 1.

Proof. This is equivalent to modify the dataset as $(\boldsymbol{x}^T, 1)$ with the same label y. Furthermore, the weight can be represented as $(\boldsymbol{w}^T, b)$. Now the linear equation (comes from the derivative) is:

$$\begin{aligned}(\boldsymbol{X}^T\boldsymbol{X})\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{1}b = \boldsymbol{X}^T\boldsymbol{y} \\ \boldsymbol{1}^T\boldsymbol{X}\boldsymbol{w} + mb = \boldsymbol{1}^T\boldsymbol{y}\end{aligned}$$

This system of equation can be re-written as the matrix equation in the proposition, and so it is proven. $\square$

Definition 1.5. (Nearest Neighbour) There are difference approach to training the predictor. We consider the set $N(\boldsymbol{x}; k)$ be the set of k-nearest (calculated using metrics) points to the point $\boldsymbol{x}$ and its associated index set $I_{\boldsymbol{x}}$ i.e:

$$I_{\boldsymbol{x}} = \{i : \boldsymbol{x}_i \in N(\boldsymbol{x}; k)\}$$

The predictor function (for classification) is given by:

$$f(\boldsymbol{x}) = \begin{cases}1 & \text{if } |\{y_i = 1 : i \in I_{\boldsymbol{x}}\}| > |\{y_i = 0 : i \in I_{\boldsymbol{x}}\}| \\ 0 & \text{otherwise}\end{cases}$$

On the other hand, the predictor for regression is defined by:

$$f(\boldsymbol{x}) = \frac{1}{k}\sum_{i\in I_{\boldsymbol{x}}}y_i$$

See here for: error vs. loss vs. risk

https://datascience.stackexchange.com/questions/35928/whats-the-difference-between-error-risk-and-loss

1.1 Bayes Estimator

Computes the expectation of the posterior, while MAP computes argmax of posterior

Definition 1.6. (Expected Error) Assuming data is obtained by sampling iid from a fixed and unknown probability density $p(\boldsymbol{x}, y)$. The expected error of the predictor is f is given by:

$$\mathcal{E}(f) = \mathbb{E}\left[(y - f(\boldsymbol{x}))^2\right] = \iint (y - f(\boldsymbol{x}))^2 p(\boldsymbol{x}, y) \text{ d}\boldsymbol{x} \text{ d}y$$

Remark 1. The goal of our learning algorithm is to compute the optimal solution f^* as we have:

$$f^* = \arg\min_f \mathcal{E}(f)$$

However, to compute f^*, we have to know p. Please note that for the binary classification i.e where $\mathcal{Y} = \{0, 1\}$ and for given predictor f, the error $\mathcal{E}(f)$ is the average number of mistake of f.

Proposition 1.3. *The optiaml solution f^* for regression problem $\mathcal{Y} = \mathbb{R}$, with square expected error. We can show that the it is:*

$$f^*(\boldsymbol{x}) = \int_{\mathcal{Y}} y \text{ d}p(y|\boldsymbol{x})$$

We assume that the joint distribution $p(y, \boldsymbol{x})$ can be decomposed as $p(y|\boldsymbol{x})p(\boldsymbol{x})$.

Proof. We have the following decomposition of the probability:

$$\mathcal{E}(f) = \int_{\mathcal{X}} \left\{ \int_{\mathcal{Y}} (y - f(\boldsymbol{x}))^2 \text{ d}p(y|\boldsymbol{x}) \right\} \text{ d}p(\boldsymbol{x})$$

We consider fixed $\boldsymbol{x} = \boldsymbol{x}'$ and given the following stort-hand, we have $e = \mathcal{E}(f(\boldsymbol{x}'))$ and $z = f(\boldsymbol{x}')$, and so we have:

$$e \propto \int_{\mathcal{Y}} (y - z)^2 \text{ d}p(y|\boldsymbol{x}')$$

The differentiation and setting this to zero giving us:

$$\begin{aligned}
\frac{\partial e}{\partial z} &= -2 \int_{\mathcal{Y}} (y - z) \text{ d}p(y|\boldsymbol{x}') \\
\iff 0 &= \int_{\mathcal{Y}} y \text{ d}p(y|\boldsymbol{x}') - z \int_{\mathcal{Y}} \text{ d}p(y|\boldsymbol{x}') \\
&= z - \int_{\mathcal{Y}} y \text{ d}p(y|\boldsymbol{x}')
\end{aligned}$$

This implies that $z = \int_{\mathcal{Y}} y \text{ d}p(y|\boldsymbol{x}')$ and so the optimal predictor is equal to what we required. □

1.2 Bias and Variance of Learning Algorithm

Remark 2. Assuming that there is a relationship between $(\boldsymbol{x}, y)$ in the dataset, which is given by $y = F(\boldsymbol{x}) + \varepsilon$ where $\mathbb{E}[\varepsilon] = 0$ and finite variance. Then the optimal predictor can be shown to be:

$$f^*(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}] = F(\boldsymbol{x})$$

Remark 3. We want to consider the expected error by an arbitrary learner $A_{\mathcal{S}}(\boldsymbol{x})$. The expected error is:

$$\mathcal{E}(A_{\mathcal{S}}(\boldsymbol{x}')) = \mathbb{E}[(y' - A_{\mathcal{S}}(\boldsymbol{x}))^2]$$

where y' is sample from the marginal $p(y|\boldsymbol{x}')$.

Lemma 1.2. *We can show that:*

$$\mathbb{E}[(Z - \mathbb{E}[X])^2] = \mathbb{E}[Z^2] + \mathbb{E}[Z]^2$$

Proof. We have the following:

$$\begin{aligned}\mathbb{E}[(Z-\mathbb{E}[Z])^2] &= \mathbb{E}[Z^2 - 2Z + \mathbb{E}[Z]^2] \\ &= \mathbb{E}[Z^2] - 2\mathbb{E}[Z]^2 + \mathbb{E}[Z]^2 \\ &= \mathbb{E}[Z^2] + \mathbb{E}[Z]^2\end{aligned}$$

□

Proposition 1.4. ***(Decomposing)*** *The square error $\mathcal{E}(A(\boldsymbol{x}'))$ can be decomposed to:*

$$\mathbb{E}[(y-f^*(\boldsymbol{x}'))^2] + (f^*(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2 + \mathbb{E}[(A_{\mathcal{S}}(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2]$$

Proof. We can decomposed the error of the learner $\mathcal{E}(A_{\mathcal{S}}(\boldsymbol{x}'))$ as we have:

$$\begin{aligned}\mathbb{E}[(y'-A_{\mathcal{S}}(\boldsymbol{x}'))^2] &= \mathbb{E}[(y')^2 - 2y'A_{\mathcal{S}}(\boldsymbol{x}') + A_{\mathcal{S}}(\boldsymbol{x}')^2] \\ &= \mathbb{E}[(y'-f^*(\boldsymbol{x}'))^2] + f^*(\boldsymbol{x}')^2 - 2f^*(\boldsymbol{x}')\mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')] + \mathbb{E}[(A_{\mathcal{S}}(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2] + \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')]^2 \\ &= \mathbb{E}[(y-f^*(\boldsymbol{x}'))^2] + (f^*(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2 + \mathbb{E}[(A_{\mathcal{S}}(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2]\end{aligned}$$

Let's show that the second equality is actually equal to the first equation:

$$\begin{aligned}&\mathbb{E}[(y'-\mathbb{E}[y'|\boldsymbol{x}'])^2] + \mathbb{E}[y'|\boldsymbol{x}']^2 - 2\mathbb{E}[y'|\boldsymbol{x}']\mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')] + \mathbb{E}[(A_{\mathcal{S}}(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2] + \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')]^2 \\ =&\mathbb{E}[(y')^2] - \mathbb{E}[y'|\boldsymbol{x}]^2 + \mathbb{E}[y'|\boldsymbol{x}']^2 - 2\mathbb{E}[y'|\boldsymbol{x}']\mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')] + \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')^2] - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')]^2 + \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')]^2 \\ =&\mathbb{E}[(y')^2] - 2\mathbb{E}[y'|\boldsymbol{x}']\mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')] + \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')^2] \\ =&\mathbb{E}[(y'-A_{\mathcal{S}}(\boldsymbol{x}'))^2]\end{aligned}$$

As required. □

Remark 4. **(Bias and Variance Tradeoff)** We can see that each term in the decomposition has the following contribution:

$$\underbrace{\mathbb{E}[(y-f^*(\boldsymbol{x}'))^2]}_{\text{Bayes' Error}} + \underbrace{(f^*(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[(A_{\mathcal{S}}(\boldsymbol{x}') - \mathbb{E}[A_{\mathcal{S}}(\boldsymbol{x}')])^2]}_{\text{Variance}}$$

The bias error describes the discrepancy between the algorithm and truth value. The Bayes error is the irreducible noise. Finally, variance capture the variance of the algorithm between training set. We can have the additional observation:

- Bias and Variance tends to trade-off against one another.
- Many parameter allows better flexibility to fit the data, which lower the bias. However, it also gives rise to high-variance, and vice versa.
- This composition holds for square loss function.

Definition 1.7. (Bayes Estimator for Classification) For C-class classification (Bayes classifier), it is given by:

$$f^*(\boldsymbol{x}) = \operatorname*{arg\,max}_{c\in[C]} p(y=c|\boldsymbol{x})$$

where the loss is 0 if we predict correctly and 1 otherwise. Furthermore, the Bayes optimal error rate is:

$$\int \Big(1 - p(y=f^*(\boldsymbol{x})|\boldsymbol{x})\Big)\,\mathrm{d}p(\boldsymbol{x})$$

Lemma 1.3. *For Z being a random variable with values $[0,1]$ and let $\mathbb{E}[Z]=\mu$ for any $a\in(0,1)$ we have:*

$$\mathbb{P}(Z>1-a) > \frac{\mu-(1-a)}{a}$$

Proof. Recall the Markov's inequality:

$$\begin{aligned}\mathbb{P}(Z \ge a) &\le \frac{\mathbb{E}[Z]}{a} \\ \implies 1 - \mathbb{P}(Z \ge a) &\ge 1 - \frac{\mathbb{E}[Z]}{a}\end{aligned}$$

We consider the following inequality, where we consider:

$$\begin{aligned}\mathbb{P}(1 - Z < a) &\ge 1 - \frac{\mathbb{E}[1-Z]}{a} \\ &= 1 - \frac{1-\mu}{a} = \frac{a - 1 + \mu}{a} = \frac{\mu - (1-a)}{a}\end{aligned}$$

□

Theorem 1.1. ***(No Free-Lunch)*** *Let A be any learning algorithm for binary classifier (where $\mathcal{Y} = \{-1, 1\}$) over domain $\mathcal{X}$. Let $m < |\mathcal{X}|/2$ being a training size. We define the loss of the function f to be:*

$$\mathcal{E}_p(f) = \int_{\mathcal{X}\times\mathcal{Y}} \mathbb{I}[f(\boldsymbol{x}) \ne y] \ \mathrm{d}p(\boldsymbol{x}, y)$$

Then there exists a distribution p such that:

- *There exists a function $f : \mathcal{X} \to \{0, 1\}$ with $\mathcal{E}_p(f) = 0$*
- *For dataset $\mathcal{S} \sim p^m$, we have*
$$\mathbb{P}_{\mathcal{S}\sim p^m}\Big[\mathcal{E}_p(A(\mathcal{S})) > 1/8\Big] \ge 1/7$$

Proof. This prove is abit more involved. Let's start with proving the first point (For now we assume the discrete distribution and finite value of $\mathcal{X}$). Let $C \subset \mathcal{X}$, where $|C| = 2m$. Denote $\mathcal{Y}^C$ being the set of all possible function $f : C \to \mathcal{Y}$ i.e $\{f_1, \dots, f_T\}$ where $T = 2^{2m}$. We can construct the distribution function p_i such that:

$$p_i(\{\boldsymbol{x}, y\}) = \begin{cases} 1/(2m) & \text{if } y = f_i(\boldsymbol{x}) \\ 0 & \text{otherwise}\end{cases}$$

for all $\boldsymbol{x}, y$. Let's consider $\mathcal{E}_{p_i}(f_i)$, which is:

$$\begin{aligned}\mathcal{E}_{p_i}(f_i) &= \sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{y\in\{-1,1\}} \mathbb{I}[f_i(\boldsymbol{x}) \ne y] p_i(\{\boldsymbol{x}, y\}) \\ &= \sum_{\boldsymbol{x}\in C} \mathbb{I}[f_i(\boldsymbol{x}) \ne f_i(\boldsymbol{x})] = 0\end{aligned}$$

And so the first point is proven. Now, consider all possible combination of data points of size m in C i.e $C^m = \{S_1, \dots, S_k\}$ where $k = (2n)^n$. We construct the dataset from f_i as $S^i_j = \{(\boldsymbol{x}, f_i(\boldsymbol{x})) : \boldsymbol{x} \in S_j\}$. Now consider the expected error of an algorithm under correction function f_i i.e

$$\begin{aligned}\mathbb{E}_{S\sim p_i^m}[\mathcal{E}_{p_i}(A(S))] &= \sum_{j=1}^{k} p_i(S^i_j)\mathcal{E}_{p_i}(A(S^i_j)) \\ &= \sum_{j=1}^{k} \frac{1}{(2n)^n}\mathcal{E}_{p_i}(A(S^i_j)) = \frac{1}{k}\sum_{j=1}^{k}\mathcal{E}_{p_i}(A(S^i_j))\end{aligned}$$

Note that for scalar $\alpha_1, \dots, \alpha_m$ we have $\max_l \alpha_l \geq 1/m \sum_{i=1}^m \alpha_i$ and $\min_l \alpha_l \leq 1/m \sum_{i=1}^m$. Consider the value of the function f_i that maximizes the error of the learner (when everything is under f_i):

$$\begin{aligned}\max_{i\in[T]} \mathbb{E}_{S\sim p_i^m}[\mathcal{E}_{p_i}(A(S))] &\geq \frac{1}{T}\sum_{i=1}^T \frac{1}{k}\sum_{j=1}^k \mathcal{E}_{p_i}(A(S_j^i)) \\ &= \frac{1}{k}\sum_{j=1}^k \frac{1}{T}\sum_{i=1}^T \mathcal{E}_{p_i}(A(S_j^i)) \geq \min_{j\in[k]} \frac{1}{T}\sum_{i=1}^T \mathcal{E}_{p_i}(A(S_j^i))\end{aligned}$$

Now, denote a set $S_j' = \{\boldsymbol{v}_1, \dots, \boldsymbol{v}_p\} \subset C$ such that its element doesn't belong in S_j for $j = 1, \dots, k$, consider average expected risk:

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^T \mathcal{E}_{p_i}(A(S_j^i)) &= \frac{1}{T}\sum_{i=1}^T \sum_{\boldsymbol{x}\in\mathcal{X}} \sum_{y\in\{-1,1\}} \mathbb{I}[A(S_j^i)(\boldsymbol{x}) \neq y]\rho_i(\{x, y\}) \\ &\geq \frac{1}{T}\sum_{i=1}^T \sum_{\boldsymbol{v}\in S_j'} \sum_{y\in\{-1,1\}} \mathbb{I}[A(S_j^i)(\boldsymbol{v}) \neq y]\rho_i(\{\boldsymbol{v}, y\}) \\ &= \frac{1}{T}\sum_{i=1}^T \sum_{\boldsymbol{v}\in S_j'} \mathbb{I}[A(S_j^i)(\boldsymbol{v}) \neq f_i(\boldsymbol{v})]\rho_i(\{\boldsymbol{v}, f_i(\boldsymbol{v})\}) \\ &= \frac{1}{T}\sum_{i=1}^T \sum_{\boldsymbol{v}\in S_j'} \frac{1}{2m}\mathbb{I}[A(S_j^i)(\boldsymbol{v}) \neq f_i(\boldsymbol{v})] \\ &\geq \frac{1}{T}\sum_{i=1}^T \sum_{\boldsymbol{v}\in S_j'} \frac{1}{2p}\mathbb{I}[A(S_j^i)(\boldsymbol{v}) \neq f_i(\boldsymbol{v})] \\ &= \frac{1}{2}\frac{1}{p}\sum_{\boldsymbol{v}\in S_j'} \frac{1}{T}\sum_{i=1}^T \mathbb{I}[A(S_j^i)(\boldsymbol{v}) \neq f_i(\boldsymbol{v})] \\ &= \frac{1}{2}\min_{r\in[p]} \frac{1}{T}\sum_{i=1}^T \mathbb{I}[A(S_j^i)(\boldsymbol{v}_r) \neq f_i(\boldsymbol{v}_r)]\end{aligned}$$

Note that $p \geq m$ because the dataset doesn't have to be unique. Before further analysis, for $\mathcal{Y}^C$, we can partion into $T/2$ pairs $(f_i, f_{i'})$ such that $f_i(\boldsymbol{x}) \neq f_{i'}(\boldsymbol{x})$ iff $\boldsymbol{x} = \boldsymbol{v}_r$ for $r \in [p]$, by setting $f_{i'}(v_r) = \neg f_i(v_r)$ where

$$\neg a = \begin{cases} 1 & \text{if } a = -1 \\ -1 & \text{if } a = 1 \end{cases}$$

Please note that $S_j^i = S_j^{i'}$ because the effect of $f_{i'}(x) \neq f_i(x)$ iff $x \notin S_j^i$. Thus, we can see that:

$$\mathbb{I}[A(S_j^i)(\boldsymbol{v}_r) \neq f_i(\boldsymbol{v}_r)] + \mathbb{I}[A(S_j^{i'})(\boldsymbol{v}_r) \neq f_{i'}(\boldsymbol{v}_r)] = 1$$

Let's consider the value inside, by the partion of list of all functions, we have:

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^T \mathbb{I}[A(S_j^i)(\boldsymbol{v}_r) \neq f_i(\boldsymbol{v}_r)] &= \frac{1}{T}\sum_{(i,i')} \mathbb{I}[A(S_j^i)(\boldsymbol{v}_r) \neq f_i(\boldsymbol{v}_r)] + \mathbb{I}[A(S_j^{i'})(\boldsymbol{v}_r) \neq f_{i'}(\boldsymbol{v}_r)] \\ &= \frac{1}{T}\sum_{(i,i')} 1 = \frac{1}{T}\frac{T}{2} = \frac{1}{2}\end{aligned}$$

This implies that:

$$\max_{i\in[T]} \mathbb{E}_{S\sim p_i^m}[\mathcal{E}_{p_i}(A(S))] \geq \min_{j\in[k]} \frac{1}{T}\sum_{i=1}^T \mathcal{E}_{p_i}(A(S_j^i)) \geq \frac{1}{4}$$

This means that for all algorithm A' getting a dataset S of size m, there is a function f and a distribution p over $\mathcal{X} \times \{0, 1\}$ such that:

$$\mathbb{E}_{S\sim p^m}[\mathcal{E}_p(A(S))] \ge \frac{1}{4}$$

and so, using the probabilistic inequality, we have:

$$\begin{aligned}\mathbb{P}\left[\mathcal{E}_p(A(S)) \ge \frac{1}{8}\right] = \mathbb{P}\left[\mathcal{E}_p(A(S)) \ge 1 - \frac{7}{8}\right] &\ge \frac{\mathbb{E}_{S\sim p^m}[8\mathcal{E}_p(A(S))](1-7/8)}{7/8} \\ &\ge \frac{2-1}{7} = \frac{1}{7}\end{aligned}$$

Thus complete the proof. □

Theorem 1.2. *As the number of sample goes to infinity, the error rate is no more than twice of the Bayes error rate for the k-nearest neighbour. Please note that the k-nearest neighbour attemps to approximate:*

$$p(y = c|\boldsymbol{x}) \approx \frac{|\{i : y_i = c, i \in I_{\boldsymbol{x}}\}|}{k}$$

We consider the points that are near the evaluation points and find the class of the neighbours that has the highest frequency.

Proof. (Sketch) We will shorten the notation as $p(c|\boldsymbol{x}) = p(y = c|\boldsymbol{x})$. The expected Bayes classifier (at $\boldsymbol{x}$) is:

$$1 - \max_{c\in[C]} p(c|\boldsymbol{x})$$

The expected error rate of 1-NN at $\boldsymbol{x}$ is given by:

$$\sum_{c=1}^{C} p_{\text{nn}}(c|\boldsymbol{x})[1 - p(c|\boldsymbol{x})]$$

As the number of sequence goes got infinity $m \to \infty$, we have $p(c|\boldsymbol{x}) \approx p_{\text{nn}}(c|\boldsymbol{x})$. Now, we will show that:

$$\sum_{c=1}^{C} p(c|\boldsymbol{x})[1 - p(c|\boldsymbol{x})] \le 2\left[1 - \max_{c\in[C]} p(c|\boldsymbol{x})\right]$$

Let $c^* = \arg\max_{c\in[C]} p(c|\boldsymbol{x})$ and $p^* = p(c^*|\boldsymbol{x})$ observe that $c \in [C]$:

$$\begin{aligned}\sum_{c=1}^{C} p(c|\boldsymbol{x})[1 - p(c|\boldsymbol{x})] &= p^*(1-p^*) + \sum_{c\in[C]\setminus c^*} p(c|\boldsymbol{x})[1 - p(c|\boldsymbol{x})] \\ &\le (C-1)\frac{1-p^*}{C-1}\left[1 - \frac{1-p^*}{C-1}\right] + p^*(1-p^*) \\ &= (1-p^*)\left[1 - \frac{1-p^*}{c-1} + p\right] \\ &\le (1-p^*)\left[1 + p\right] \le 2(1-p^*)\end{aligned}$$

The second inequality comes from the fact that sum is maximized when all $p(c|\boldsymbol{x})$ have the same value. And the last inequality comes from the fact that $p, p^* < 1$. Thus complete the proof. □

Remark 5. One can show that for $k = k(m)$ where m is the size of the dataset, one can show that

$$\mathcal{E}(k - \text{NN}) \to \mathcal{E}(f^*)$$

note that k depends on the data size, as $m \to \infty$ with the condition that $k(m) \to \infty$ and $k(m)/m \to \infty$.

Remark 6. **(Curse of Dimensionality)** The rate of convergence depends exponentially on the input dimension. This problems occure thoughout the ML algorithms. The intuitive is that the volumn increases exponentially with the dimension implies that the number of data required to cover the space to perform estimate also increase exponentially: The ration between volumn unit d-dim ball centered at the origin and 1/2-unit ball at the origin is $(1/2)^d$.

Definition 1.8. (Empirical Risk) We are given only the sample from probability $p(\boldsymbol{x}, y)$. The natural approach is to approximate the expected error using empirical error:

$$\mathcal{E}_{\text{emp}}(\mathcal{S}, f) = \frac{1}{|\mathcal{S}|}\sum_{(\boldsymbol{x},y):\mathcal{S}} (y_i f(\boldsymbol{x}_i))^2$$

Definition 1.9. (Empirical Risk Minimization (with reguarlizer)) If we consider all possible function, we can always find the function with 0 empirical error (remember) this is known as overfitting. To solve this we have to restrict the function space to be $\mathcal{H}$ called hypothesis space, which ERM is defined as:

$$f_{\mathcal{S}} = \operatorname*{arg\,min}_{f\in\mathcal{H}} \mathcal{E}_{\text{emp}}(\mathcal{S}, f)$$

Remark 7. **(Example of Hypothesis Space)** We can consider the following increasing "complexity" as for the regression in 1D as we have:

$$\mathcal{H}_n = \left\{ f(x) = \sum_{l=1}^{n} a_l x^l + b : a_1, \dots, a_n, b \in \mathbb{R} \right\}$$

Choosing the correct model requires a cross-validation. Unless the prior knowledge is avaliable on f^*, we can't expect $f^* \in \mathcal{H}$, while we can't allow too large $\mathcal{H}$ as it leads to the overfitting.

2 Kernel and Regression

2.1 Introduction

Definition 2.1. (Convex Set) A set $\mathcal{X}$ is convex if $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{X}$ is convex if $\alpha\boldsymbol{p} + (1-\alpha)\boldsymbol{q} \in \mathcal{X}$

Definition 2.2. (Convex Function) A function $f : \mathcal{X} \to \mathbb{R}$ is convex iff for all $\boldsymbol{p}, \boldsymbol{q} \in \mathcal{X}$ in convex set and $\alpha \in (0, 1)$ as we have:

$$f(\alpha\boldsymbol{p} + (1-\alpha)\boldsymbol{q}) \le \alpha f(\boldsymbol{p}) + (1-\alpha) f(\boldsymbol{q})$$

A function f is concave if $-f$ is convex. A function is *strictly convex* if we replace $\le$ with $<$.

Remark 8. **(Various Comments on Convex Function)** We have the following results on the convex function, as we have:

- If f and g are convex, then $f + g$ is convex.
- If f is convex and g is affine (linear + constant) then $f(g(\cdot))$ is convex.
- Suppose $\boldsymbol{M}$ is symmetric matrix, then $\boldsymbol{M}$ is positive semi-definite matrix iff $f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x}$ is convex.
- Level set $\{\boldsymbol{x} : f(\boldsymbol{x}) = c\}$ where $c \in \mathbb{R}$ of convex function f is convex.
- For $f : (a, b) \to \mathbb{R}$ if $f'' \ge 0$ then f is convex.
- For $f : \mathcal{X} \subseteq \mathbb{R}^n \to \mathbb{R}$ if $\nabla^2 f(\boldsymbol{x}) \succeq \boldsymbol{0}$ for all $\boldsymbol{x} \in \mathcal{X}$, then f is convex.

2.2 Ridge Regression

Definition 2.3. (Ridge Regression Problem) Given a function $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ with a dataset:

$$\mathcal{S} = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)\} \subset \mathbb{R}^n \times \mathbb{R}$$

Assuming the dataset is generated by the unknown function g i.e $(\boldsymbol{x}, g(\boldsymbol{x}))$. Then suppose that the vector $\boldsymbol{x}_i$ are linearly independent with $m = n$, then there is a unique solution, whose parameter $\boldsymbol{w}$ solves:

$$\boldsymbol{X}\boldsymbol{w} = \boldsymbol{y}$$

where $\boldsymbol{y} = (y_1, \dots, y_m)^T$ and $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m]^T \in \mathbb{R}^{m\times n}$.

Definition 2.4. (Well-Posed) The solution/problem is called well-posed if: the solution exists, uniquem and depends continuously on the data. The regularized theory allows general framework to solve ill-posted problem (we can choose the term to penalize complex function).

Definition 2.5. (Regularized Empirical Error) We minimize the following regularized empirical error, which is given by:

$$\begin{aligned}\mathcal{E}_{\text{emph},\lambda}(\boldsymbol{w}) &= \sum_{j=1}^{m}(y_i - \boldsymbol{w}^T\boldsymbol{x}_i)^2 + \lambda\sum_{i=1}^{n} w_i^2 \\ &= (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \lambda\left\|\boldsymbol{w}\right\|_2^2\end{aligned}$$

We can see that the parameter $\lambda > 0$ defines the trade-off between error and the norm of vector $\boldsymbol{w}$ (which restricts the complexity of the model).

Proposition 2.1. *Solving the regularized empirical error by setting its gradient to* $\mathbf{0}$*, gives us:*

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

Furthermore, we can show that the weight $\boldsymbol{w} = \sum_{i=1}^m \alpha_i\boldsymbol{x}_i$ *and the solution can be written as:*

$$f(\boldsymbol{x}) = \sum_{i=1}^{m}\alpha_i\boldsymbol{x}_i^T\boldsymbol{x}_i$$

where $\boldsymbol{\alpha} = (\boldsymbol{X}\boldsymbol{X}^T + \lambda\boldsymbol{I}_m)^{-1}\boldsymbol{y}$*. This is called dual form, while* $f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x}$ *is called primal form.*

Proof. Starting with the derivative, we have:

$$\nabla\mathcal{E}_{\text{emp},\lambda}(\boldsymbol{w}) = -2\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + 2\lambda\boldsymbol{w} = \mathbf{0}$$

which implies the weight of the first form i.e $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_n)^{-1}\boldsymbol{X}^T\boldsymbol{y}$. Now, we can also see that:

$$\boldsymbol{w} = \frac{\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})}{\lambda}$$

Assume the the dual form of the weight $\boldsymbol{w} = \sum_{i=1}^m \alpha_i\boldsymbol{x}_i$ as we have:

$$\alpha_i = \frac{y_i - \boldsymbol{w}^T\boldsymbol{x}_i}{\lambda} = \frac{y_i - (\sum_{i=1}^m \alpha_i\boldsymbol{x}_i)^T\boldsymbol{x}_i}{\lambda}$$

Now solving for the value of y_i, which we have:

$$\begin{aligned}y &= \left(\sum_{j=1}^{m}\alpha_j\boldsymbol{x}_j\right)^T\boldsymbol{x}_i + \lambda\alpha_i \\ &= \sum_{j=1}^{m}(\alpha\boldsymbol{x}_j^T\boldsymbol{x}_j + \lambda\alpha_j\delta_{ij}) = \sum_{j=1}^{m}(\boldsymbol{x}_j^T\boldsymbol{x}_j + \lambda\delta_{ij})\boldsymbol{\alpha}\end{aligned}$$

and so we have $(\boldsymbol{X}\boldsymbol{X}^T + \lambda\boldsymbol{I}_m)\boldsymbol{\alpha} = \boldsymbol{y}$ □

Remark 9. **(Advantage of Dual Form)** The dual form allow us to gain a computational advantage for both training and testing time:

- *Training Time*: Solving $\boldsymbol{w}$ in the primal function requires $\mathcal{O}(mn^2 + n^3)$ operations while solving for dual form $\mathcal{O}(nm^2 + m^3)$ if $m \ll n$ then it is more efficient that primal.
- *Testing Time*: Computing $f(\boldsymbol{x})$ in test vector $\boldsymbol{x}$ in the primal form requires $\mathcal{O}(n)$ operations but the dual form requires $\mathcal{O}(nm)$ operations.

2.3 Basis/Kernel Functions

Definition 2.6. (Basis/Feature Function) We have the function $\boldsymbol{\phi} : \mathbb{R}^n \to \mathbb{R}^N$ as we have:

$$\boldsymbol{\phi}(\boldsymbol{x}) = \Big(\boldsymbol{\phi}_1(\boldsymbol{x}), \dots, \boldsymbol{\phi}_N(\boldsymbol{x})\Big)^T$$

for $\boldsymbol{x} \in \mathbb{R}^n$, where we call $\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_N$ are called basis function and $\boldsymbol{\phi}(\boldsymbol{x})$ is called feature vector, and feature space is defined by: $\{\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{R}^n\}$

Remark 10. We can use the feature map of the data instead of real data $\boldsymbol{\phi}(\boldsymbol{x})$. This gives us the many advantages, for example:

- The map: $\boldsymbol{\phi}(\boldsymbol{x}) = (\boldsymbol{x}, 1)^T$ allow us to have the bias terms.
- The map: $\boldsymbol{\phi}(\boldsymbol{x}) = (\boldsymbol{x}_1 x_2)^T$ allow us to consider the interaction between inputs (individual elements).

We can also consider the second order correlation if $\boldsymbol{x} \in \mathbb{R}^n$ as:

$$\boldsymbol{\phi}(\boldsymbol{x}) = (x_1x_1, x_1x_2, \dots, x_1x_n, x_2x_2, x_2x_3, \dots, x_2x_n, \dots, x_nx_n)^T$$

now the feature vector has the size of $(n^2 + n)/2$. However, if we consider the inner product, we will have:

$$\begin{aligned}
\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{t}) \rangle &= (x_1x_1, x_1x_2, \dots, x_nx_n)^T(t_1t_1, t_1t_2, \dots, t_nt_n) \\
&= (x_1t_1 + \cdots + x_nt_n)(x_1t_1 + \cdots + x_nt_n) \\
&= (\boldsymbol{x}^T\boldsymbol{t})^T
\end{aligned}$$

How we circumvent computation into new feature space with kernel trick

Note that $\mathcal{O}(n)$ but the native computation will take $\mathcal{O}(n^2)$. This leads to decrease the computation complexity (please see the dual form too).

Definition 2.7. (Kernel Function) Given a feature map $\boldsymbol{\phi}$, we define the asssociated kernel function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as we have:

$$k(\boldsymbol{x}, \boldsymbol{t}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{t}) \rangle$$

Please note that the computing $k(\boldsymbol{x}, \boldsymbol{t})$, which it doesn't depends on computing $\boldsymbol{\phi}(\boldsymbol{x})$.

Remark 11. **(Feature Map not Unique)** The feature map isn't unique. Consider the $\boldsymbol{\phi}$ that is associated with kernel k, and so $\hat{\boldsymbol{\phi}} = \boldsymbol{U\phi}$ where $U \in \mathbb{R}^{N\times N}$. The feature can be difference in values and dimension but gives rise to the same kernel:

$$(\boldsymbol{U\phi})^T(\boldsymbol{U\phi}) = \boldsymbol{\phi}^T\boldsymbol{\phi}$$

Theorem 2.1. ***(Representor)*** *Consider the loss to be:*

$$\boxed{\mathcal{E}_{emp,\lambda}(\boldsymbol{w}) = \sum_{i=1}^{m} V(y_i, \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle) + \lambda \langle \boldsymbol{w}, \boldsymbol{w}\rangle}$$

We have expressed our loss as a dot product of transformed features and their weights using kernels

where $V : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function. If V is differentiable with respected to its second argument and $\boldsymbol{w}$ is a minimizer of $\mathcal{E}_\lambda$, then $\boldsymbol{w}$ has the form of:

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i) \implies f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle = \sum_{i=1}^{m} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x})$$

Proof. The proof is similar to the dual form. Setting the derivative of $\mathcal{E}_\lambda$ with respected to zero and we have:

$$\sum_{i=1}^{m} V'(y_i, \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle)\boldsymbol{\phi}(\boldsymbol{x}_i) + 2\lambda \boldsymbol{w} = 0$$

Compared to $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)$, we can see that:

$$\alpha_i = \frac{1}{2\lambda} V'(y_i, \langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle)$$

From the definition of $\boldsymbol{w}$, we can see that:

$$\alpha_i = \frac{1}{2\lambda} V'\left(y_i, \sum_{j=1}^{m} k(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j\right)$$

for $i = 1, \dots, m$. Finding $\boldsymbol{\alpha}$ is done by solving the following optimization problem

$$\boxed{\arg\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} V(y_i, (\boldsymbol{K}\boldsymbol{\alpha})_i) + \boldsymbol{\alpha}^T \boldsymbol{K}\boldsymbol{\alpha}}$$

alpha is like our weight vector for the transformed feature space => we need to find it

□

Definition 2.8. (Positive Semi-Definite Kernel) The kernel $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is positive semi-definite if it is symmetrix and given the set of points $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$, the matrix:

$$\begin{bmatrix} k(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix}$$

is positive semi-definite. symmetric matrix with non-negative eigenvalues i.e. +ve energy

Theorem 2.2. *Kernel k is positive definite iff:*

$$k(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{t})\rangle$$

for $\boldsymbol{x}, \boldsymbol{t} \in \mathbb{R}^n$ for some feature map $\boldsymbol{\phi} : \mathbb{R}^n \to \mathcal{W}$ for Hilber space $\mathcal{W}$

Proof. We will consider only one direction. If $k(\boldsymbol{x}, \boldsymbol{t}) = \langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{t})\rangle$, then we have:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} c_i c_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left\langle \sum_{i=1}^{m} c_i \boldsymbol{\phi}(\boldsymbol{x}_i), \sum_{j=1}^{m} c_j \boldsymbol{\phi}(\boldsymbol{x}_j) \right\rangle = \left\| \sum_{i=1}^{m} c_i \boldsymbol{\phi}(\boldsymbol{x}_i) \right\|^2 \geq 0$$

□

Definition 2.9. (Polynomial Kernel) If $p : \mathbb{R} \to \mathbb{R}$ is a polynomial with non-negative coefficient then $k(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{x}^T \boldsymbol{t})$ where $\boldsymbol{x}, \boldsymbol{t} \in \mathbb{R}^n$ and k positive semi-definite kernel.

Proposition 2.2. *If $\boldsymbol{A}$ is an $n \times n$ positive semi-definite matrix, the function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ defined by:*

$$k(\boldsymbol{x}, \boldsymbol{t}) = \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{t}$$

is a generalized linear kernel and it is a positive semi-definite kernel.

Proof. Since $\boldsymbol{A}$ is positive semi-definite, we can write $\boldsymbol{A}$ in the form of $\boldsymbol{A} = \boldsymbol{R}\boldsymbol{R}^T$ for some $\boldsymbol{R} \in \mathbb{R}^{n\times n}$. Thus, k is represented by a feature map $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{R}^T\boldsymbol{x}$. As we can see that:

$$\begin{aligned}\sum_{ij} c_i c_j \boldsymbol{x}_i^T \boldsymbol{A}\boldsymbol{x}_j &= \sum_{ij} c_i c_j (\boldsymbol{R}^T\boldsymbol{x}_i)^T(\boldsymbol{R}^T\boldsymbol{x}_j) \\ &= \sum_i c_i [\boldsymbol{R}^T\boldsymbol{x}_i]^T \left[\sum_j c_j(\boldsymbol{R}^T\boldsymbol{x}_j)\right] = \left\|\sum_i c_i \boldsymbol{R}^T\boldsymbol{x}_i\right\|^2 \ge 0\end{aligned}$$

□

Proposition 2.3. *If $k : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is a positve semi-definite kernel and $\boldsymbol{\phi} : \mathbb{R}^n \to \mathbb{R}^N$.*

$$\tilde{k}(\boldsymbol{x}, \boldsymbol{t}) = k(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{t}))$$

The kernel $\tilde{k}$ defined to be $\tilde{k} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a positive definite kernel.

Proposition 2.4. *Given a positive semi-definite kernels k_1 and k_2, ak_1 is a positive semi-definite kernel if $a > 0$ and $k_1 + k_2$ is also a positive definite kernel.*

Proposition 2.5. *We consider the following combination of kernel k_1 and k_2 are given as:*

$$k(\boldsymbol{x}, \boldsymbol{t}) = k_1(\boldsymbol{x}, \boldsymbol{t})k_2(\boldsymbol{x}, \boldsymbol{t})$$

where $\boldsymbol{x}, \boldsymbol{t} \in \mathbb{R}^d$ is a kernel.

Proof. For the product of kernel, we have:

- We want to show that for positive semi-definite $\boldsymbol{A}$ and $\boldsymbol{B}$ where $\boldsymbol{C} = A \odot \boldsymbol{B}$ is a positive semi-definite.
- Since $\boldsymbol{A}$ and $\boldsymbol{B}$ are positive semi-definite, where it can be factorized as $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{U}^T$ and $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{V}^T$ for $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{n\times n}$ as we have:

$$\begin{aligned}\sum_{i=1}^n\sum_{j=1}^n z_i z_j C_{ij} &= \sum_{i=1}^n\sum_{j=1}^n z_i z_j \left(\sum_{r=1}^n U_{ir}U_{jr}\right)\left(\sum_{s=1}^n V_{is}V_{js}\right) \\ &= \sum_{i=1}^n\sum_{j=1}^n\sum_{r=1}^n\sum_{s=1}^n z_i z_j U_{ir}U_{jr}V_{is}V_{js} \\ &= \sum_{r=1}^n\sum_{s=1}^n\sum_{i=1}^n\sum_{j=1}^n z_i z_j U_{ir}U_{jr}V_{is}V_{js} \\ &= \sum_{r=1}^n\sum_{s=1}^n\sum_{i=1}^n z_i U_{ir}V_{is}\sum_{j=1}^n z_j U_{ji}V_{js} = \sum_{r=1}^n\sum_{s=1}^n\left(\sum_{i=1}^n z_i U_{ir}V_{is}\right)^2 \ge 0\end{aligned}$$

  Thus complete the proof. This proves the polynomial kernel is positive definite kernel.

□

Remark 12. **(Several Kernels)** We have the following positve definite kernel, where we have $a \ge 0$:

- $k(\boldsymbol{x}, \boldsymbol{t}) = (\boldsymbol{x}^T\boldsymbol{t})^r$
- $k(\boldsymbol{x}, \boldsymbol{t}) = (a + \boldsymbol{x}^T\boldsymbol{t})^r$
- $k(\boldsymbol{x}, \boldsymbol{t}) = \sum_{i=1}^d (a^i/i!)(\boldsymbol{x}^T\boldsymbol{t})^r$

- Gaussian Kernel: $k(\boldsymbol{x},\boldsymbol{t}) = \exp(-\beta\left\|\boldsymbol{x}-\boldsymbol{t}\right\|^2)$ for $\beta > 0$ the data $\boldsymbol{x},\boldsymbol{t}\in\mathbb{R}^n$ (It has infinite dimensional feature map)
- ANOVA kernel: $k(\boldsymbol{x},\boldsymbol{t}) = \prod_{i=1}^n(1+x_it_i)$

Remark 13. Consider the following polynomial kernel as we have:

$$\sum_{i=1}^d \frac{a^i}{i!}(\boldsymbol{x}^T\boldsymbol{t})^i$$

Suppose we have $r=\infty$, this can converge uniformly to $\exp(a\boldsymbol{x}^T\boldsymbol{t})$ showing that it is a kernel, where if $n=1$, the feature map is:

$$\phi = \left(1, \sqrt{2}x, \sqrt{\frac{a}{2}}x^2, \sqrt{\frac{a^3}{6}}x^3, \cdots\right) = \left(\sqrt{\frac{a^i}{i!}} : i\in\mathbb{N}\right)$$

Definition 2.10. (Transition Invariance/Radial Kernel) We say that a kernel $k:\mathbb{R}^d\times\mathbb{R}^d\to\mathbb{R}$ is:

- *Transition Invariance*: If the kernel has the form:

$$k(\boldsymbol{x},\boldsymbol{t}) = H(\boldsymbol{x}-\boldsymbol{t})$$

  for all $\boldsymbol{x},\boldsymbol{t}\in\mathbb{R}^d$ where $H:\mathbb{R}^d\to\mathbb{R}$ is a differentiable function.

- *Radial*, if kernel has the form:

$$k(\boldsymbol{x},\boldsymbol{t}) = h(\|\boldsymbol{x}-\boldsymbol{t}\|)$$

represents an infinite-dimensional feature space with all polynomial powers of x

  for all $\boldsymbol{x},\boldsymbol{t}\in\mathbb{R}^d$ where $h:[0,\infty)\to[0,\theta)$ is the differentiable function.

Remark 14. The important example of a radial kernel in the Gaussian kernel as we have:

$$k(\boldsymbol{x},\boldsymbol{t}) = \exp(-\beta\left\|\boldsymbol{x}-\boldsymbol{t}\right\|^2)$$

which is a product of 2 kernel as $k(\boldsymbol{x},\boldsymbol{t}) = \exp(-\beta(\boldsymbol{x}^T\boldsymbol{x}+\boldsymbol{t}^T\boldsymbol{t}))\exp(2\beta\boldsymbol{x}^T\boldsymbol{t})$

Remark 15. **(Ridge Regression with Feature Map)** Given the dataset $\boldsymbol{X}\in\mathbb{R}^{m\times n}$ and $\boldsymbol{y}\in\mathbb{R}^{m\times 1}$. Starting with the basis function $\phi_1,\dots,\phi_N$ where $\phi_i:\mathbb{R}^n\to\mathbb{R}$ with the map:

$$\boldsymbol{\Phi} = \begin{bmatrix}\phi_1(\boldsymbol{x}_1) & \cdots & \phi_N(\boldsymbol{x}_1)\\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_m) & \cdots & \phi_N(\boldsymbol{x}_m)\end{bmatrix}\in\mathbb{R}^{m\times N}$$

We have the regression coefficient as we have $\boldsymbol{w} = (\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\lambda\boldsymbol{I}_N)^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$ Here we calculate inside the new feature space

Remark 16. **(Kernel Ridge Regression)** Given the same setting, a kernel function $\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}$, where the kernel matrix is given by:

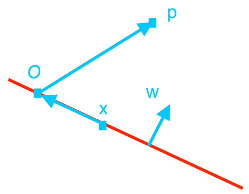
$$\boldsymbol{K} = \begin{bmatrix}k(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_1,\boldsymbol{x}_n)\\ \vdots & \ddots & \vdots \\ k(\boldsymbol{x}_n,\boldsymbol{x}_1) & \cdots & k(\boldsymbol{x}_n,\boldsymbol{x}_n)\end{bmatrix}\in\mathbb{R}^{m\times m}$$

Regression coefficient is then given by $\boldsymbol{\alpha} = (\boldsymbol{K}+\lambda\boldsymbol{I}_m)^{-1}\boldsymbol{y}$ as the function is:

$$\hat{y}(\boldsymbol{x}) = \sum_{i=1}^m \alpha_i k(\boldsymbol{x}_i,\boldsymbol{x})$$

Here we don't

Take to p dot w to get vector p normal to plane. Divide by ||w|| to nromalise



3 Support Vector Machine

3.1 Forming Problems

Dot any point, x, on this line, with w and you will get 0
Dot any point, x, on this line, with w and you will get 0

Definition 3.1. (Seperating Hyperplane) Let the dataset be $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m} \in \mathbb{R}^n \times \{-1, 1\}$. The hyperplane is the set such that:

$$\mathcal{H}_{\boldsymbol{w},b} = \left\{\boldsymbol{x} \in \mathbb{R}^n : \boldsymbol{w}^T\boldsymbol{x} + b = 0\right\}$$

Definition 3.2. (Linearly Separatable) The data are linearly separatable if there exists $\boldsymbol{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ such that:

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) > 0$$

for $i = 1, \dots, m$, which we call $\mathcal{H}_{\boldsymbol{w},b}$ a separating hyperplane. Note that it is a strict inequality.

Proposition 3.1. ***(Finding A distance from Plane)*** *If $\mathcal{H}_{\boldsymbol{w},b}$ is a hyperplane, we also define the distance from a point $\boldsymbol{x}$ to be:*

$$\boxed{\frac{\boldsymbol{w}^T\boldsymbol{x} + b}{\|\boldsymbol{w}\|}}$$

Proof. We consider the projection from the point $\boldsymbol{x}$ to $\mathcal{H}_{\boldsymbol{w},b}$ as we have:

$$\boldsymbol{p} = \boldsymbol{x} - \frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}$$

To show that $\boldsymbol{p}$ is indeed a projection:

- We will have to show that $\boldsymbol{p}$ is on hyperplane

$$\boldsymbol{w}^T\boldsymbol{p} + b = \boldsymbol{w}^T\boldsymbol{x} - \frac{\boldsymbol{w}^T\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2} + b = 0$$

- $\boldsymbol{x} - p$ is orthogonal to $\boldsymbol{p} - \boldsymbol{x}'$ where $\boldsymbol{x}'$ is any point from on the hyperplane:

$$\begin{aligned}
(\boldsymbol{p} - \boldsymbol{x})^T(\boldsymbol{p} - \boldsymbol{x}') &= \left\langle -\frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}, \boldsymbol{p} - \boldsymbol{x}' \right\rangle \\
&= \left\langle -\frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}, \boldsymbol{x} - \frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2} - \boldsymbol{x}' \right\rangle \\
&= \left\langle -\frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}, \boldsymbol{x} - \boldsymbol{x}' \right\rangle + \left\langle \frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}, \frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2} \right\rangle \\
&= \left\langle -\frac{\boldsymbol{w}(b + \boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}, \boldsymbol{x} - \boldsymbol{x}' \right\rangle + \frac{\|\boldsymbol{w}\|^2(b + \boldsymbol{w}^T\boldsymbol{x})^2}{\|\boldsymbol{w}\|^4} \\
&= -\frac{b + \boldsymbol{w}^T\boldsymbol{x}}{\|\boldsymbol{w}\|^2}\langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{x}' \rangle + \frac{(b + \boldsymbol{w}^T\boldsymbol{x})^2}{\|\boldsymbol{w}\|^2} \\
&= -\frac{(b + \boldsymbol{w}^T\boldsymbol{x})(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{x}')}{\|\boldsymbol{w}\|^2}\langle \boldsymbol{w}, \boldsymbol{x} - \boldsymbol{x}' \rangle + \frac{(b + \boldsymbol{w}^T\boldsymbol{x})^2}{\|\boldsymbol{w}\|^2} \\
&= -\frac{b(\boldsymbol{w}^T\boldsymbol{x}) - b(\boldsymbol{w}^T\boldsymbol{x}') + (\boldsymbol{w}^T\boldsymbol{x})^2 - (\boldsymbol{w}^T\boldsymbol{x})(\boldsymbol{w}^T\boldsymbol{x}')}{\|\boldsymbol{w}\|^2} + \frac{(b + \boldsymbol{w}^T\boldsymbol{x})^2}{\|\boldsymbol{w}\|^2} \\
&= -\frac{b(\boldsymbol{w}^T\boldsymbol{x}) + b^2 + (\boldsymbol{w}^T\boldsymbol{x})^2 + (\boldsymbol{w}^T\boldsymbol{x})b}{\|\boldsymbol{w}\|^2} + \frac{(b + \boldsymbol{w}^T\boldsymbol{x})^2}{\|\boldsymbol{w}\|^2} = 0
\end{aligned}$$

Please note that $\boldsymbol{w}^T\boldsymbol{x}' + b = 0$.

Now, we are left to find the distance between $\boldsymbol{p}$ and $\boldsymbol{x}$, which we can find it to be:

$$\sqrt{(\boldsymbol{p}-\boldsymbol{x})^T(\boldsymbol{p}-\boldsymbol{x})} = \sqrt{\left\langle \frac{\boldsymbol{w}(b+\boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2}, \frac{\boldsymbol{w}(b+\boldsymbol{w}^T\boldsymbol{x})}{\|\boldsymbol{w}\|^2} \right\rangle} = \frac{|b+\boldsymbol{x}^T\boldsymbol{w}|}{\|\boldsymbol{w}\|}$$

Thus complete the proof. □

Definition 3.3. (Margin) As we have the distance from a point $\boldsymbol{x}$ to the plane $\mathcal{H}_{\boldsymbol{w},b}$ to be $\rho_{\boldsymbol{x}}(\boldsymbol{w}, b)$. If $\mathcal{H}_{\boldsymbol{w},b}$ separates the training set S, we define a margin as:

$$\rho_S(\boldsymbol{w}, b) = \min_{i\in[m]} \rho_{\boldsymbol{x}_i}(\boldsymbol{w}, b)$$

Definition 3.4. (Optimal Separating Hyper-Planes) We want to find the weight and bias of a separating hyperplane such that the the margin is maximized :

$$\boxed{\rho(S) = \max_{\boldsymbol{w},b} \min_{i\in[m]} \left\{ \frac{y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)}{\|\boldsymbol{w}\|} : y_j(\boldsymbol{w}^T\boldsymbol{x}_j + b) > 0 \text{ for } j \in [m] \right\}}$$

Furthermore, to get the unqiue $\boldsymbol{w}, b$, we may consider 2 choices:

- Set $\|\boldsymbol{w}\| = 1$, so $\rho_{\boldsymbol{x}}(\boldsymbol{w}, b) = \left|\boldsymbol{w}^T\boldsymbol{x} + b\right|$ and so:

$$\rho_S = \min_{i\in[m]} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b)$$

- Choose $\|\boldsymbol{w}\|$ such that $\rho_S(\boldsymbol{w}, b) = 1/\|\boldsymbol{w}\|$ or:

$$\min_{i\in[m]} y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) = 1$$

We simplify the problem...

We will consider the second case.

Proposition 3.2. *The optimal separating hyperplane is equivalent to following optimization problem:*

Set the contraint so that we can ignore the numerator and focus on maximising 1 / ||w||

Just make sure that the plane separates the classes at least by a margin of 1

$$\boxed{\begin{aligned} \min_{\boldsymbol{w},b} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \\ \text{s.t} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 \end{aligned}}$$

minimising ||w|| =√w• w

and set up a Lagrangian to minimise

for $\boldsymbol{w} \in \mathbb{R}^n$. The quantity $1/\|\boldsymbol{w}\|$ is the margin of optimal separating hyperplane.

Proof. We have following the second case:

$$\begin{aligned} \rho(S) &= \max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\boldsymbol{w}\|} : \min_{j\in[m]} \left\{ y_j(\boldsymbol{w}^T\boldsymbol{x}_j + b) \right\} = 0, y_k(\boldsymbol{w}^T\boldsymbol{x}_k + b) > 0 \text{ for } k \in [m] \right\} \\ &= \max_{\boldsymbol{w},b} \left\{ \frac{1}{\|\boldsymbol{w}\|} : \left\{ y_j(\boldsymbol{w}^T\boldsymbol{x}_j + b) \right\} \geq 1 \right\} = \frac{1}{\min_{\boldsymbol{w},b} \left\{ \|\boldsymbol{x}\| : \left\{ y_j(\boldsymbol{w}^T\boldsymbol{x}_j + b) \right\} \geq 1 \right\}} \end{aligned}$$

□

Proposition 3.3. *To minimize a differentiable convex function $f(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}$ subjected to linear inequality $\boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{c}$. We may solve the problem with Lagragian:*

$$\boxed{L(\boldsymbol{x}, \boldsymbol{\alpha}) = f(\boldsymbol{x}) - \boldsymbol{\alpha}^T(\boldsymbol{A}\boldsymbol{x} - c)}$$

If the optimization problem is feasible that is $\{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}\} \neq \emptyset$, we can show that:

$$\max_{\boldsymbol{\alpha}\geq\boldsymbol{0}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\alpha}) = \min_{\boldsymbol{x}} f(\boldsymbol{x}) \ \ s.t\ \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{c}$$

Then set up in dual form:
– can set up with kernel trick
– unlocks computational efficiency if n < d
– regularisation terms usually appear as dual form anyways

And there is a necessary and sufficient condition called KKT for a solution $(\boldsymbol{\alpha}^*\boldsymbol{z}^*)$:

- $\boldsymbol{A}\boldsymbol{x}^* \leq \boldsymbol{c}$
- $\boldsymbol{\alpha}^* \geq \boldsymbol{0}$
- $\nabla_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\alpha})|_{\boldsymbol{x}^*} = \boldsymbol{0}$
- $(\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{c})_i \boldsymbol{\alpha}_i^* = \boldsymbol{0}_i$ *for* $i \in [m]$

Proposition 3.4. *The dual form for the SVM is:*

$$\boxed{\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{A}\boldsymbol{\alpha} + \sum_{i=1}^m \alpha_i \\ \text{s.t} \quad & \sum_{i=1}^m y_i \alpha_i = 0 \text{ for } i \in [m] \\ & \alpha_i \geq 0 \end{aligned}}$$

Solve using Lagrangian as below

where $\boldsymbol{A} = (y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j : i, j \in [m])$*. The solution to the primal problem is:*

$$\boxed{\boldsymbol{w}^* = \sum_{i=1}^m \alpha_i^* y_i \boldsymbol{x}_i}$$

as the weight is the linear combination of the data. Finally the variable b^* *can be determine by find the weight* $\boldsymbol{x}_j$ *that satisfies the condition:*

$$y_i((\boldsymbol{w}^*)^T \boldsymbol{x}_i + b) - 1 = 0$$

Then we bias can be found by rearrange as we have $b^* = y_i - (\boldsymbol{w}^*)^T \boldsymbol{x}_j$*. The point that satisfies this conditon is called* support vector.

Proof. We consider the Lagragian to be:

$$L(\boldsymbol{w}, b; \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w} - \sum_{i=1}^m \alpha_i [y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) - 1]$$

where $\alpha_i \geq 0$ is Lagragian multipler. Let's minimize L over $\boldsymbol{w}$ and b and maximized over $\boldsymbol{\alpha}$ with $\boldsymbol{\alpha} \geq \boldsymbol{0}$. We can see that the partial derivative is:

$$\begin{aligned} \frac{\partial L}{\partial b} &= -\sum_{i=1}^m y_i \alpha_i = 0 \\ \frac{\partial L}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^m \alpha_i y_i \boldsymbol{x}_i = 0 \implies \boldsymbol{w} = \sum_{i=1}^m \alpha_i y_i \boldsymbol{x}_i \end{aligned}$$

Now, we can see that the optimal weight will have the linear combination term. Let's plugging this back into Lagragian and we have:

$$\frac{1}{2}\underbrace{\boldsymbol{w}^T \boldsymbol{w}}_{\boldsymbol{\alpha}^T \boldsymbol{A}\boldsymbol{\alpha}} - \underbrace{\sum_{i=1}^m \alpha_i y_i \boldsymbol{w}^T \boldsymbol{x}_i}_{\boldsymbol{\alpha}^T \boldsymbol{A}\boldsymbol{\alpha}} - b\underbrace{\sum_{i=1}^m \alpha_i y_i}_{0} + \sum_{i=1}^m \alpha_i$$

□

Remark 17. The new point $\boldsymbol{x}$ can be classified as:

$$\boxed{\text{sign}\left(\sum_{i=1}^m y_i \alpha_i^* \boldsymbol{x}_i^T \boldsymbol{x}_i + b^*\right)}$$

One can show that the expected generalization error of SVM trained on $m - 1$ sample is bounded by n_{sv}/m, where n_{sv} is the number of support vector.

The parameter C>0 controls the trade-off between maximising the size of the margin and ensuring that the data points x are correctly positioned on the appropriate side of the margin.

Remark 18. **(Linear Non-Separatable Case)** We would like to minimize the following objective function:

$$\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^m V_{\text{mc}}(y_i, \boldsymbol{w}^T\boldsymbol{x}_i + b)$$

as we have $V_{\text{mc}}(y, \hat{y}) = \mathbb{I}[y = \text{sign}(\hat{y})]$ but it is NP-Hard and so we will have to convexify the problem by consider the hinge loss, instead:

$$V_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - h\hat{y})$$

This will gives us the convex optimization.

Proposition 3.5. *The hinge loss can be reformulated using the slack variable and gives us the following optimization problem:*

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^m \xi_i \\ \text{s.t} \quad & y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \ge 1 - \xi_i \\ & \xi_i \ge 0 \textit{ for } i \in i = 1, \dots, m \end{aligned}$$

This would in turn, gives us the following dual problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{A}\boldsymbol{\alpha} + \sum_{i=1}^m \alpha_i \\ \text{s.t} \quad & \sum_{i=1}^m y_i\alpha_i = 0 \text{ for } i \in [m] \\ & 0 \le \alpha_i \le C \end{aligned}$$

We will consider the implication of KKT conditon afterward.

Proof. We now have the following Lagragian to be:

$$L(\boldsymbol{w}, b; \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i[y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) - 1] - \sum_{i=1}^m \beta_i\xi_i$$

where $\alpha_i, \beta_i \ge 0$ are Lagragian multipler. We minimize L over $(\boldsymbol{w}, \boldsymbol{\xi}, b)$ and maxmize L with respected to the variables as:

$$\begin{aligned} \frac{\partial L}{\partial b} &= -\sum_{i=1}^m y_i\alpha_i = 0 \\ \frac{\partial L}{\partial \boldsymbol{w}} &= \boldsymbol{w} - \sum_{i=1}^m \alpha_i y_i \boldsymbol{x}_i = 0 \implies \boldsymbol{w} = \sum_{i=1}^m \alpha_i y_i \boldsymbol{x}_i \\ \frac{\partial L}{\partial \xi_i} &= c - \alpha_i - \beta_i = 0 \implies 0 \le \alpha_i \le C \end{aligned}$$

Plugging this back gives us the dual form. Please note that both $\alpha_i, \beta_i \ge 0$ □

Remark 19. **(Interpretation of The Results)** The dual problem is similar to the eariler linear separatable case, as we have additional box constraint. The weight is given as:

$$\boldsymbol{w}^* = \sum_{i=1}^m \alpha_i^* y_i \boldsymbol{x}_i$$

where $\boldsymbol{b}^*$ is the same. For a new KKT conditon, we have:

$$\begin{aligned} &\alpha_i^*(y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* - 1 + \xi_i^*) = 0 \\ &(C - \alpha_i^*)\xi_i^* = 0 \end{aligned}$$

where the second equation follows from $\beta_i^* = C - \alpha_i^*$. There are difference points to consider:

- $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* > 1$ implies that $\alpha_i^* = 0$ where the point isn't support vector.
- $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* < 1$ implies that $\alpha_i^* = C$ where the point is a support vector slack ξ_i^* outlier.
- $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* = 1$ implies that $\alpha_i^* \in [0, C]$ and if $\alpha_i^* > 0$, it is a support vector on a margin.

On the otherhand, we have:

- $\alpha_i^* = 0$ then we have $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* \geq 1$ and $\xi_i^* = 0$
- $\alpha_i^* \in (0, C)$ then we have $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* = 1$ and $\xi_i^* = 0$
- $\alpha_i^* = C$ then we have $y_i(\boldsymbol{w}^*)^T\boldsymbol{x}_i + b^* \leq 1$ and $\xi_i^* \geq 0$

Remark 20. The role of parameter C is that:

- The parameter C controls the trade-off between $\|\boldsymbol{w}\|^2$ and the training error $\sum_{i=1}^m \xi_i$
- The value of α_i^* is piecewise quadratic of C
- C is selected by minimizing leave-one-out (LOO) cross-validation error.

To compute the LOO error, we need to retrain the SVM no more than the number of support vector making it fast to train. One can observe that we can use the n_{sv}/m as an upper bound on LOO error.

Definition 3.5. (Kernelized SVM) Given the feature map $\boldsymbol{\phi}(\boldsymbol{x}) : \mathcal{X} \to \mathcal{W}$, we can replace $\boldsymbol{x}$ with $\boldsymbol{\phi}(\boldsymbol{x})$ and $\boldsymbol{x}^T\boldsymbol{t}$ by $\langle \boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{t})\rangle$. The result function is:

$$\boxed{f(\boldsymbol{x}) = \sum_{i=1}^m y_i\alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b}$$

The parameter can be found using the matrix $\boldsymbol{A} = (y_iy_jk(\boldsymbol{x}_i, \boldsymbol{x}_j) : i, j \in [m])$ and the new point is classified the same.

Remark 21. **(Connection to the Regularization)** SVM formulation is equivalent to the following problem:

$$\mathcal{E}_\lambda = \sum_{i=1}^m \max\Big(1 - y_i\left(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle + b\right), 0\Big) + \lambda\|\boldsymbol{w}\|^2$$

where we set $\lambda = 1/(2C)$ and so we have:

$$\begin{aligned}
&\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \left\{ C\sum_{i=1}^m \xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 : y_i\left(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle + b\right) \geq 1 - \xi_i, \xi_i \geq 0 \right\} \\
=&\min_{\boldsymbol{w},b} \left\{ \min_{\boldsymbol{\xi}} \left\{ C\sum_{i=1}^m \xi_i + \frac{1}{2}\|\boldsymbol{w}\|^2 : y_i\left(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle + b\right) \geq 1 - \xi_i, \xi_i \geq 0 \right\}\right\} \\
=&\min_{\boldsymbol{w},b} \left\{ C\sum_{i=1}^m \Big(1 - y_i\left(\langle \boldsymbol{w}, \boldsymbol{\phi}(\boldsymbol{x}_i)\rangle + b\right), 0\Big) + \frac{1}{2}\|\boldsymbol{w}\|^2 \right\} = C\mathcal{E}_{1/(2C)}(\boldsymbol{w}, b)
\end{aligned}$$

Remark 22. **(SVM for Regression)** If we have the regression for the SVM, then we use the following loss:

$$|y - f(\boldsymbol{x})|_\varepsilon = \max(|y - f(\boldsymbol{x})| - \varepsilon, 0)$$

This would gives the following optimization problem:

$$\boxed{\begin{array}{rl}
\min & \dfrac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\displaystyle\sum_{i=1}^m(\xi_i + \xi_i^*) \\
\text{s.t} & \boldsymbol{w}^T\boldsymbol{x}_i + b - y_i \leq \varepsilon + \xi_i \\
& y_i - \boldsymbol{w}^T\boldsymbol{x}_i - b \leq \varepsilon + \xi_i^* \\
& \xi_i, \xi_i^* \geq 0 \text{ for } i \in [m]
\end{array}}$$

Please note that the loss function is scale sensitive as the error below certain. This gives the sparse solution. One can use decompositve to solve all of the KKT problems.

4 Tree Based and Ensemble Model

4.1 Tree Based Method

Definition 4.1. (Tree Method) We are interesting to partition the input space into retangles and fit simple model in each one; for example, we have the function:

$$f(\boldsymbol{x}) = \sum_{p=1}^{P} c_p \mathbb{I}[\boldsymbol{x} \in R_p]$$

Where we hve the following:

- We partition the input space with hyper-retangle $R_1, R_2, \dots, R_p$ where: $\bigcup_{p=1}^{P} R_p = \mathcal{X}$ and $R_a \cap R_b = \emptyset$ if $a \neq b$
- $\{c_p\}_{p=1}^{P}$ is some real parameter with a natural choice to be:

$$c_p = \text{avg}(y_i | \boldsymbol{x}_i \in R_p) = \frac{\sum_{i=1}^{m} y_i \mathbb{I}[\boldsymbol{x}_i \in R_p]}{\sum_{i=1}^{m} \mathbb{I}[\boldsymbol{x}_i \in R_p]}$$

We are interested to solving the following optimization problem:

$$\boxed{\min_{R_1,\dots,R_p} \left\{ \sum_{i=1}^{m} \left(y_i - \sum_{p=1}^{P} \text{avg}(y_i | \boldsymbol{x}_i \in R_p) \mathbb{I}[\boldsymbol{x}_i \in R_p] \right)^2 \right\}}$$

Definition 4.2. (Heuristic Search) It seem to be intractable, so we need heuristic approach. Let's find the way to split the tree. Define a pair of axis parallel half-spaces:

$$R_1(j,s) = \{\boldsymbol{x} | x_j \leq s\} \qquad R_2(j,s) = \{\boldsymbol{x} | x_j > s\}$$

Then we search for optimal values j^* and s^*, which solves the problem:

$$\min_{j,s} \left\{ \min_{c_1} \sum_{\boldsymbol{x}_i \in R_1(j,s)} (y_i - c_1(\boldsymbol{x}_i))^2 + \min_{c_2} \sum_{\boldsymbol{x}_i \in R_2(j,s)} (y_i - c_2(\boldsymbol{x}_i))^2 \right\}$$

We minimise the total squared error by selecting the best split (feature jj and threshold ss). Squared error measures how far the true values y_i are from the predicted value c1 or c2.

The inner minimizer is solved by:

$$c_1^* = \text{avg}(y_i | \boldsymbol{x}_i \in R_1(j,s)) \qquad c_2^* = \text{avg}(y_i | \boldsymbol{x}_i \in R_2(j,s))$$

For each splitting variable j, the search for best split at point s can be don by $\mathcal{O}(m)$ computation. Thus, the problem is solved in $\mathcal{O}(nm)$ computation. The decision tree can be solved by repeatedly splitting the tree branches.

Remark 23. **(Overfitting)** If we keep repeating the heuristic search process, we will overfit the data. There are several ways to fix this:

- The following the split only if it decreases the empirical error more than the threshold. However, this might be the best as we might find split below a bad mode.
- We might consider the maximal depth of split tree is reached. This could leads to an underfitting or overfitting. We need to look at the data to determine the size of tree.

Remark 24. **(Solving Overfitting)** We choose the tree adapting from the data. We grows the large tree $\hat{T}$ (stopping when the maximum number of data is assigned at each node). Now consider the prune the tree with cost complexity pruining i.e looks for subtree $T_\lambda \subseteq \hat{T}$ that minimizes:

$$\boxed{C_\lambda(T) = \sum_{p=1}^{|T|} m_p Q_p(T) + \lambda |T|}$$

where T is the subtree of $\hat{T}$, where we have:

- p runs over leaf nodes of T (a subset of the nodes of $\hat{T}$)
- m_p is the number of data point assigned to node p
- Q_p is the training error given as:

  $$Q_p = \frac{1}{m_p} \sum_{\boldsymbol{x}_i \in R_p} (y_i - c_p)^2$$

  At the first term in C_λ is the training error.

One can show that there is a unique $T_\lambda \subseteq \hat{T}$, with minimize C_λ, while a good value of λ can be found by cross-validation.

Definition 4.3. (Weakest Link Pruning) We successively collapse the internal nodes that produces the smallest per node increase in:

$$\sum_{p=1}^{|T|} m_p Q_p(T)$$

We continue until the root the tree is produce. As now, we have a list of prunned trees. We can search along this list for the one that miminizes the objective C_λ, and one can show that T_λ is in the produced list of subtree, hence the algorithm gives the optimal solution.

Definition 4.4. (Classification Tree) When the output is a categorical variable, we use the same algorithm above with 2 important modification:

- For each region R_n, we define the empirical class probability, as we have:

  $$p_{nk} = \frac{1}{m_p} \sum_{(\boldsymbol{x}_i, y_i) \in R_n} \mathbb{I}[y_i = k]$$

- We classify an input which falls in region n in the class with new probability as we have:

  $$f(\boldsymbol{x}) = \underset{k \in \{1,\dots,K\}}{\arg\max} \sum_{n=1}^{N} p_{nk} \mathbb{I}[\boldsymbol{x} \in R_n]$$

Definition 4.5. (Impurity) We consider the training error $Q_p(T)$ to be called impurity, which in can be one of these values:

- *Misclassification Error*: $1 - p_{pk(n)}$ where $k(n) = \arg\max_{k \in \{1,\dots,k\}} p_{nk}$
- *Gini-Index*: $\sum_k p_{pk}(1 - p_{pk})$
- *Cross-Entropy*: $\sum_k p_{pk} \log(1/p_{pk})$

The cross-entropy or gini-index are used to growing the tree, while the misclassification error are often used to prune the tree.

4.2 Ensemble Methods + Bagging

Theorem 4.1. ***(Chernoff-Bound)*** *Let $X_1, X_2, \dots, X_n$ be independent random variable. Assuning $0 \leq x_i \leq 1$. We denote the $X = \sum_{i=1}^n X_i$ and $\mu = \mathbb{E}[X] = \sum_{i=1}^n \mathbb{E}[X_i]$, then for all $0 \leq k \leq \mu$:*

$$\mathbb{P}(X \leq k) \leq \exp\left(-\frac{(\mu - k)^2}{2\mu}\right)$$

Remark 25. **(Motivation - Wisdom of the Crowd)** A single individual might often wrong but the crowd majority may often be corrected. Suppose each individual in the crowd $h_1, h_2, \dots, h_{2T+1}$ of the size $2T+1$ predicts the outcome correctly with probability $1/2 + \gamma$ independent from each other. We consider the vote of the crowd to be:

$$H_T = \text{sgn}\left(\sum_{t=1}^{2T+1} h_t\right)$$

The probability of H_T being wrong is given as:

$$\mathbb{P}(H_T \text{ is wrong }) = \sum_{i=1}^{T} \binom{2T+1}{i} \left(\frac{1}{2} + \gamma\right)^i \left(\frac{1}{2} - \gamma\right)^{2T+1-i}$$

We simplify the above using a Chernoff bound. We let $X_1, \dots, X_i, \dots, X_n$ be Bernoulli random variable where $X_i = 1$ if voter i is correct and 0 otherwise. Taking $k = T$ and $n = 2T+1$ thus:

$$\mu = (2T+1)\left(\frac{1}{2} + \gamma\right) = T + \frac{1}{2} + 2T\gamma + \gamma$$

Now, we substuite the bound:

$$\begin{aligned} \mathbb{P}(H_T \text{ is wrong }) &\leq \exp\left(-\frac{(\mu - T)^2}{2\mu}\right) \\ &= \exp\left(-\frac{(1/2 + 2T\gamma + \gamma)^2}{2(T + 1/2 + 2T\gamma + \gamma)}\right) \\ &\leq \exp\left(-\frac{4T^2\gamma^2}{5T}\right) = \exp\left(-\frac{4\gamma^2}{5}T\right) \end{aligned}$$

The bound may be too crude but the probability of getting wrong, exponentially decays to zero.

Definition 4.6. (Bagging Algorithm) The idea of bagging algorithm is to reduce the variance of a classifier by having many variances of the classifier and then voting. We have the following algorithm:

- Training data: $S = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)\} \subset \mathbb{R}^d \times \{-1, 1\}$
- Ensemble of size T
- Resample dataset of size M
- Classifier function $h_{\mathcal{S}}(\boldsymbol{x})$

This leads to the following pseudocode:

Algorithm 1 Bagging Algorithm

```
1: for $t = 1, 2, \cdots, T$ do
2:     $S[t] = M$ examples sampled with repalcement from $S$
3: end for
4: Return: We perform the following prediction:
```

$$H(\boldsymbol{x}) = \text{sgm}\left(\sum_{t=1}^{T} h_{S[t]}(\boldsymbol{x})\right)$$

We may set M to be m.

Remark 26. If we set $M = m$, we can find the number of unique example from S are in bag $S(t)$. The probability that a particular example doesn't appear in the bag is $(1-1/m)^m$, and please note that:

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{e} \approx 0.368..$$

so there will be around 63% examples in each dataset $S[t]$.

Definition 4.7. (Random Forest) We observe the wisdom of the crowds argument. We can build a tree using a subset of size k features, which is usually $\sqrt{d}$ or $\log d$.

4.3 Boosting

Remark 27. **(Concept of Boosting)** Some of the problem is easy to find the "rule of thumb" that is usually correct. It is hard to find accurate prediction rule. To boosting algorithm is given by:

- Create a computer program for derriving rough rule of thumb.
- We can shoow a rule of thumb to fit a subset of example.
- Repeat T times.
- Combined the classifier by weighted majority votes.

There are two concerns: How do we choose the subset of examples ? At each round as we want to concentrate on the hardest example. How do we combine the weak learner ? This can be done by weighted majority.

Definition 4.8. (Notation Used in Boosting) We have the following variables, as we have:

- $D_t(i)$: Weight on example i at time t when $\sum_{i=1}^m D_t(i) = 1$
- α_t: Weight on weak learner t where $\alpha_t \in \mathbb{R}$
- $h_t(\cdot) : \mathbb{R}^d \to \{-1, +1\}$: Weak learner that is generated at time t.
- $f(\cdot)$: Weighted on weak learner. $\sum_{t=1}^T \alpha_t h_t(\boldsymbol{x})$
- $H(\boldsymbol{x}) = \text{sgn}(f(\boldsymbol{x}))$: Final classifier.
- ε_t: Weight error of weak learner $h_t(\cdot)$ at time t:

$$\varepsilon_t = \sum_{i=1}^m D_t(i)\mathbb{I}[h_t(\boldsymbol{x}_i) \neq y_i]$$

- Weak learning will generate the output:

$$D_t(1), \dots, D_t(m), (\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$$

  The weak-learner will output a weaker learner $h_t(\cdot)$ such that $\varepsilon_t < 1/2$

Definition 4.9. (Adaboost Algorithm) We have the following pseudocode for the adaboost this is shown in the pseudocode 2.

Algorithm 2 Adaboost

```
1: Input: Training set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$
2: Initialize: $D_1(1) = \cdots = D_1(m) = 1/m$
3: for $i = 1, 2, \cdots, T$ do
4:     Fit the classifier $h_t : \mathbb{R}^d \to \{-1, 1\}$ using a distribution $D_t$
5:     Choose $\alpha_t \in \mathbb{R}$:
           $$\alpha_t = \frac{1}{2} \log \frac{1 - \varepsilon_t}{\varepsilon_t}$$
6:     Update for each $i \in [m]$, where $Z_t$ is normalization factor:
           $$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$
7: end for
8: Return: Classifier is given as:
           $$H(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$$
```

Typically $\varepsilon_t \leq 0.5$ hence $\alpha_t \geq 0$. Thus f is a linear combination of h_t with weights controlled by training error. The basic intuition for the adaboost assign a larger weight are assigned to hard examples, hence the weak learner will focus on those example.

Theorem 4.2. *Given a training set $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ and assume that each iteration of Adaboost the weak learner reutrns a hypothesis with a weighted error $1/2 - \gamma \geq \varepsilon_t$, then training error of the output hypothesis is at most:*

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{I}[H(\boldsymbol{x}_i) \neq y_i] \leq \exp(-2\gamma^2 T)$$

Proof. Please note that the training error is bounded as:

$$\frac{1}{m} \sum_{i=1}^{m} \mathbb{I}[H(\boldsymbol{x}_i) \neq y_i] \leq \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(\boldsymbol{x}_i))$$

where $f = \sum_t \alpha_t h_t$ so that $H(\boldsymbol{x}) = \operatorname{sgn}(f(\boldsymbol{x}))$. The inequality follows from $H(\boldsymbol{x}_i) \neq y_i$ implies that $\exp(-y_i f(\boldsymbol{x}_i)) \geq 1$. Now consider the definition of D_t where, recursively:

$$D_{T+1}(i) = \frac{1}{m} \frac{\prod_{t=1}^{T} \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{\prod_{t=1}^{T} Z_t}$$

We can expand this equation, where we have:

$$\begin{aligned}
\frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f(\boldsymbol{x}_i)) &= \frac{1}{m} \sum_{i=1}^{m} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}_i)\right) \\
&= \frac{1}{m} \sum_{i=1}^{m} \prod_{t=1}^{T} \exp(-y_i \alpha_t h_t(\boldsymbol{x}_i)) \\
&= \sum_{i=1}^{m} D_{T+1}(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t
\end{aligned}$$

If at each iteration, we choose α_t and h_t by minimizing Z_t, the final training error of H will be reduced most rapidly. Recall that:

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))$$

Using the fact that Z_t is a binary, we have that:

$$\begin{aligned} Z_t &= \exp(\alpha_t) \sum_{i:y_i \neq h_t(\boldsymbol{x}_i)} D_t(i) + \exp(-\alpha_t) \sum_{i:y_i = h_t(\boldsymbol{x}_i)} D_t(i) \\ &= \varepsilon_t \exp(\alpha_t) + (1-\varepsilon_t)\exp(-\alpha_t) \end{aligned}$$

Setting the derivative of Z_t to zero with respected to α_t, which gives us the weight:

$$\alpha_t = \frac{1}{2}\log\frac{1-\varepsilon_t}{\varepsilon_t}$$

Placing α_t to the value Z_t, and we have:

$$\begin{aligned} Z_t &= \varepsilon_t \exp(\alpha_t) + (1-\varepsilon_t)\exp(-\alpha_t) \\ &= 2\sqrt{\varepsilon_t(1-\varepsilon_t)} = \sqrt{1-4\gamma_t^2} \end{aligned}$$

Please note that $\gamma_t = 1/2 - \varepsilon_t$. Hence we have:

$$\frac{1}{m}\sum_{i=1}^m \mathbb{I}[H(\boldsymbol{x}_i) \neq y_i] \le \prod_{t=1}^T Z_t = \prod_{t=1}^T \sqrt{1-4\gamma_t^2} \le \exp\left(-2\sum_{t=1}^T \gamma_t^2\right)$$

The final inequality use the fact that $1 - x \le \exp(x)$. If each weak classifier is slightly better than random guessing, the training drops exponentially fast. □

Remark 28. **(Derivation of Adaboost)** The boosting can be seen as a greedy way to solve problem:

$$\min\left\{\sum_{i=1}^m V\left(y_i, \sum_{i=1}^T \alpha_t h_t(\boldsymbol{x}_i)\right) : \alpha_1, \dots, \alpha_T \in \mathbb{R}^T, h_1, \dots, h_T \in \mathcal{H}^T\right\}$$

where $\mathcal{H}$ is hypothesis class which contains the weaker learner and the loss function is exponential for instance $V(y, \hat{y}) = \exp(-y\hat{y})$. At each iteration, a new basis function is added to the current basis expansion $f^{(t-1)} = \sum_{s=1}^{t-1}\alpha_s h_s$, which we have:

$$(\alpha_t, h_t) = \arg\min_{\alpha_t, h_t} \sum_{i=1}^m V\Big(y_i, f^{(t-1)}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i)\Big)$$

unlike the decision tree, where each iteration in previous basis is re-adjusted. In statistics literature, this kind of model is called stagewise additive model. To derive the adaboost, substute $V(y, \hat{y}) = \exp(-y\hat{y})$ and we consider the followimg optimization problem:

$$\min_{\alpha_t, h_t} \sum_{i=1}^m \exp\left(-y_i\left(f^{(t-1)}(\boldsymbol{x}_i) + \alpha_t h_t(\boldsymbol{x}_i)\right)\right)$$

We define $\mathcal{D}_t(i) = \exp(-y_i f^{(t-1)}(\boldsymbol{x}_i))$ as we have:

$$\min_{\alpha_t, h_t} \sum_{i=1}^m \mathcal{D}_t(i)\exp(-\alpha_t h_t(\boldsymbol{x}_i) y_i)$$

We can see that the This equation can be rewritten as:

$$\begin{aligned} &\min_{\alpha_t, h_t}\left(\exp(\alpha_t)\sum_{i:y_i \neq h_i(\boldsymbol{x}_i)} \mathcal{D}_t(i) + \exp(-\alpha_t)\sum_{i:y_i = h_t(\boldsymbol{x}_i)} \mathcal{D}_t(i)\right) \\ &= \min_{\alpha_t, h_t}\left((e^{\alpha_t} - e^{-\alpha_t})\sum_{i=1}^m \mathcal{D}_t(i)\mathbb{I}[y_i \neq h_t(\boldsymbol{x}_i)] + e^{-\alpha_t}\sum_{i=1}^m \mathcal{D}_t(i)\right) \end{aligned}$$

This is similar to the adaboost, which we have: h_t minimizes the weight misclassification error weight by $\mathcal{D}_t$ that is is propotional to adaboost D_t. Finally, minimization of α_t is the same as adaboost.

Remark 29. **(Classification and Regression)** In the typical setup of classification as we have:

$$\min_{f\in\mathcal{F}} \sum_{i=1}^{m} V(y_i, f(\boldsymbol{x}_i)) + \lambda \text{ complexity}(f)$$

There are some problems with classification as we use the exponential loss. To make the class of function $\mathcal{F}$ both rich and smooth, we have the function f that maps to $\mathbb{R}$ rather than $\{-1, 1\}$ then predict the sign. We have the typical loss function, where we have for $y \in \{-1, +1\}$:

- Misclassification Loss: $V_{\text{mc}}(y, \hat{y}) = \mathbb{I}[y = \text{sgn}(\hat{y})]$. It isn't continuous.
- Hinge Loss: $V_{\text{hinge}}(y, \hat{y}) = \max(0, 1 - y\hat{y})$. It punishes the negative margin but not positive margin, but it isn't differetiable everywhere.
- Square Loss: $V_{\text{sq}}(y, \hat{y}) = (y - \hat{y})^2$. It unnecessary punishes predicting with increasing positive margin.
- Exponential Loss: $V_{\text{exp}}(y, \hat{y}) = \exp(-y\hat{y})$. It punishes negative margine and promote large positve margin.

Thus the exponential loss is choosen.

5 Online Learning

5.1 Introduction

Definition 5.1. (Online Learning with Expert Advice) There exists an online sequence of data:

$$S = (\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m) \in \{0, 1\}^n \times \{0, 1\}$$

The vector $\boldsymbol{x}_t$ is the set of prediction of n experts at time t, which we aim to predict y_t. We would like to find an algorithm that tried to combine the prediction $\boldsymbol{x}_t$ of the n expets to predict $\hat{y}_t$, an estimate of y_t. The loss of mater algorithm A on sequence S:

$$L_A(S) = \sum_{t=1}^{m} |y_t - \hat{y}_t|$$

We want to find an algorithm with a small loss.

Definition 5.2. (Regret) Recall the loss function $L_A(S)$ and we let:

$$L_i(S) = \sum_{t=1}^{m} |y_t - x_{t,i}|$$

being the loss of i-th expert E_i. The aim of our algorithm should that the found of the form, such that for all sequence S:

$$L_A(S) \le a \min_i L_i(S) + b \log(n)$$

where a, b are small constant. This is known as regret as it is the loss of objective related to the best expert.

Definition 5.3. (Halving Algorithm) Suppose that there is an expert are consistent (gives correct answer), we can perofrm the search on this consistent expert, in which we will have the correct prediction:

- If mistake is mad, the number of consistent experts is (at least) halved.
- For any sequence with consistent expert Halving algorithm made less than or equal to $\log_2 n$ mistakes.

5.2 Learning from Expert Algorithm

Definition 5.4. (Weighted Majority) This algorithm for non-consistent expert, which we can perform the prediction with larger scale. We have weight of the wrong expert is multiplied by $\beta \in [0, 1)$.

Theorem 5.1. *The number of mistake of master algorithm M, with $\beta = 1/2$ is given by:*

$$M \leq 2.63 \min_i M_i + 2.63 \ln n$$

where M_i is the number of mistakes of expert E_i

Proof. We have the following quantities:

- $M_{t,i}$ is the number of mistake of the expert i, E_i at the start of trial t.
- $w_{t,i} = \beta^{M_{t,i}}$ weights of E_i at the begin of trial t.
- Please note that $w_{1,i} = 1$, and $W_t = \sum_{i=1}^n w_{t,i}$ is the total weight at trial t.

It is clear that the total weight of the minority is when it is less that $1/2W_t$, but the total weight of the majority is when it is more than $1/2W_t$. There are 2 scenarios, which we have:

- If no mistake, the minority expert weight is multiplied by β as we have (Trivial Bound): $W_{t+1} \leq 1 \cdot W_t$
- If there is a mistake, the majority exper weights are multiplied by β as:
$$\begin{aligned} W_{t+1} &= \text{Minority} + \beta\text{Majority} \\ &\leq \frac{1}{2}W_t + \beta\text{Majority} \\ &\leq \frac{1}{2}W_t + \beta\frac{1}{2}W_t \leq \frac{1+\beta}{2}W_t \end{aligned}$$
The third inequality comes from the fact that the majority is at least $1/2W_t$, making the upperbound tighter.

This gives us:

$$\left(\frac{1+\beta}{2}\right)^M W_1 \geq W_{m+1} = \sum_{j=1}^n W_{m+1,j} = \sum_{j=1}^n \beta^{M_j} \geq \beta^{M_i}$$

Note that M is number of mistakes, while m is number of running time. It is clear that $W_1 = n$, solving for M gives us:

$$M \leq \frac{\ln 1/\beta}{\ln 2/(1+\beta)} M_i + \frac{1}{\ln 2/(1+\beta)} \ln n$$

Setting $\beta = 1/e$ yields the result, completing the proof. □

Remark 30. **(Notion of Regret)** We would like to obtain regret bound for arbitary loss function $L : \mathcal{Y} \times \hat{\mathcal{Y}} \to [0, \infty]$. Making our notation of regre more precise:

$$L_A(S) - \min_{i \in [n]} L_i(S) \leq o(m)$$

where $o(m)$ denotes some function that is sublinear in m that depends on other parameter:

$$\frac{L_A(S) - \min_{i \in [n] L_i(S)}}{m} \leq \frac{o(m)}{m}$$

Note that as $m \to \infty$:

$$\frac{L_A(S)}{m} \leq -\frac{\min_{i \in [n]} L_i(S)}{m}$$

The limit of the mean asympototic loss is bounded by the mean of asympototic loss of the best expert.

- The loss function $L : \{0, 1\} \times [0, 1] \to [0, \infty)$ the entropic loss given by:

$$L(y, \hat{y}) = y \ln \frac{y}{\hat{y}} + (1 - y) \ln \frac{1 - y}{1 - \hat{y}}$$

  We can show that the regret with small constant with $\log(n)$

- Arbitary loss function $L : \mathcal{Y} \times \hat{\mathcal{Y}} \to [0, B]$. The regret will be $\mathcal{O}(\sqrt{m \log n})$

Definition 5.5. (Simplex and Related Entropy) We define the simplex over probability distribution:

$$\Delta_n = \left\{ \boldsymbol{x} : [0, 1]^n : \sum_{i=1}^{n} x_i = 1 \right\}$$

We define the relative entropy $d : \Delta_n \times \Delta_n \to [0, \infty)$ as we have:

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} u_i \ln \frac{u_i}{v_i}$$

we define the entropy loss is given as $L_{\text{en}}(y, \hat{y}) = d((y, 1 - y), (\hat{y}, 1 - \hat{y}))$

Definition 5.6. (Weighted Average) We will consider a projection in $[0, 1]$ rather than $\{0, 1\}$, and we will predict with weighted average. One weight per expert as we have $w_{t,i} = \beta^{L_{i,t}} = \exp(-\eta L_{t,i})$ where $L_{t,i}$ is the cumulative loss of E_i before the trial t, while η is the learning rate. The master algorithm predicts with the weighted average:

$$v_{t,i} = \frac{w_{t,i}}{\sum_{i=1}^{n} w_{t,i}} \qquad \hat{y}_t = \sum_{i=1}^{n} v_{t,i} x_{t,i} = \boldsymbol{v}_t^T \boldsymbol{x}_t$$

where the $x_{t,i}$ is the prediction of E_i at trial t. We start with the initialize weight $\boldsymbol{v}_1 = \boldsymbol{w}_1 = (1/n, \dots, 1/n)$. This gives the pseudocode:

Algorithm 3 Weighted Average

```
1: Input: Input $\boldsymbol{v}_1 = \boldsymbol{w}_1 = (1/n, \dots, 1/n)$ with $L_{\text{WA}} = 0$ and $\boldsymbol{L} = \boldsymbol{0}$
2: for $i = 1, 2, \cdots, m$ do
3:     Receives instance $\boldsymbol{x}_t \in [0, 1]^n$
4:     Predict $\hat{y}_t = \boldsymbol{v}_t^T \boldsymbol{x}_t$
5:     Receives label $y_t \in [0, 1]$
6:     Incur Loss $L_{\text{WA}} = L_{\text{WA}} + L(y_t, \hat{y}_t)$ and $L_i = L_i + L(y_t, x_{t,i})$ for $i \in [n]$
7:     Update Weight, for $i \in [n]$:
           $v_{t+1,i} = \frac{v_{t,i} \exp(-\eta L(y_t, x_{t,i}))}{\sum_{j=1}^{n} v_{t,j} \exp(-\eta L(y_t, x_{t,j}))}$
8: end for
```

Theorem 5.2. *For sequence of examples $S = (\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m) \in [0, 1]^n \times [0, 1]$. The regret of the weighted average WA algorithm is:*

$$L_{\text{WA}}(S) - \min_i L_i(S) \leq 1/\eta \ln(n)$$

with square and entropic loss for $\eta = 1/2$ and $\eta = 1$ respectively.

Proof. We will proof the progress vs regret first, for all $\boldsymbol{u} \in \Delta_n$. Let's start with the assumption that $y_t = 1$

and by the error $L_{\text{en}}(1,x) = -\ln x$, we have

$$\begin{aligned}
d(\boldsymbol{u}, \boldsymbol{v}_t) - d(\boldsymbol{u}, \boldsymbol{v}_{t+1}) &= \sum_{i=1}^n u_i \ln\left(\frac{v_{t+1,i}}{v_{t,i}}\right) \\
&= \sum_{i=1}^n u_i \ln \frac{\frac{v_{t,i}\exp(-L_{\text{en}}(1,x_{t,i}))}{\sum_{j=1}^n v_{t,j}\exp(-L_{\text{en}}(1,x_{t,j}))}}{v_{t,i}} \\
&= \sum_{i=1}^n u_i \ln \frac{\frac{v_{t,i}x_{t,i}}{\sum_{j=1}^n v_{t,j}x_{t,j}}}{v_{t,i}} \\
&= \sum_{i=1}^n u_i \ln \frac{x_{t,i}}{\hat{y}_t} = \left(\sum_{i=1}^n u_i \ln x_{t,i}\right) - \ln(\hat{y}_t) \\
&= L_{\text{en}}(y_t, \hat{y}_t) - \sum_{i=1}^n u_i L_{\text{en}}(y_t, x_{t,i})
\end{aligned}$$

This also works by symmetry with the case $y = 0$, and so the claim:

$$d(\boldsymbol{u}, \boldsymbol{v}_t) - d(\boldsymbol{u}, \boldsymbol{v}_{t+1}) = L_{\text{en}}(y_t, \hat{y}_t) - \sum_{i=1}^n u_i L_{\text{en}}(y_t, x_{t,i})$$

is correct. Consider the telescoping sum, which we have:

$$\sum_{t=1}^m L_{\text{en}}(y_t, \hat{y}_t) - \sum_{t=1}^m\sum_{i=1}^n u_i L_{\text{en}}(y_t, x_{t,i}) = d(\boldsymbol{u}, \boldsymbol{v}_1) - d(\boldsymbol{u}, \boldsymbol{v}_{m+1})$$

Note that for any $\boldsymbol{u} \in \Delta_n$, espescially the unit vector, which is shown to be an upper bound, and we can see that that $d(\boldsymbol{u}, \boldsymbol{v}_1) \le \ln n$ and $-d(\boldsymbol{u}, \boldsymbol{v}_{m+1}) \le 0$, and so we have proven the theorem, as:

$$\begin{aligned}
\sum_{t=1}^m L_{\text{en}}(y_t, \hat{y}_t) - \sum_{t=1}^m\sum_{i=1}^n u_i L_{\text{en}}(y_t, x_{t,i}) &= \sum_{t=1}^m L_{\text{en}}(y_t, \hat{y}_t) - \min_i L_i(S) \\
&\le d(\boldsymbol{u}, \boldsymbol{v}_1) - d(\boldsymbol{u}, \boldsymbol{v}_{m+1}) \le \ln(n) - 0
\end{aligned}$$

Note that we can have a unit vector u_i that is the correct expert. □

Definition 5.7. (Allocation Setting) On each trial, the learner plays an allocation $\boldsymbol{v}_t \in \Delta_t$, the the nature returns the loss vector $\boldsymbol{l}_t$ for example of the loss of expert i is $l_{t,i}$. There are 2 models for the learner:

- We can consider the incure loss directly: $L_A(t) = \boldsymbol{v}_t^T \boldsymbol{l}_t$
- The learner randomly select $\hat{y}_t \in [n]$ according to discrete distribution over $[n]$ with probability $v_{t,i}$ for each action, thus we have:

  $$\mathbb{E}[L_A(t)] = \mathbb{E}[L_{t,\hat{y}}] = \boldsymbol{v}_t^T \boldsymbol{l}_t$$

  The mechanism generating the loss vector $\boldsymbol{l}_t$ must be obvious to the learner's selection $\hat{y}$ until $t+1$

This setting can simulate the setting where we rescue side-information $\boldsymbol{x}_t$ and have a fixed loss function.

Theorem 5.3. ***(Hedge Theorem)*** *For all sequence of loss vector $S = \boldsymbol{l}_1, \dots, \boldsymbol{l}_m \in [0,1]^n$. The regret of the weighted average algorithm with $\eta = \sqrt{2m \ln n}$ is equal to:*

$$\mathbb{E}[L_{\text{WA}}(S)] - \min_i L_i(S) \le \sqrt{2m \ln n}$$

Proof. Given any $\boldsymbol{u} \in \Delta_n$. Letting $Z_t = \sum_{i=1}^n v_{t,i} \exp(-\eta l_{t,i})$, we observe that:

$$\begin{aligned}
d(\boldsymbol{u}, \boldsymbol{v}_t) - d(\boldsymbol{u}, \boldsymbol{v}_{t+1}) &= \sum_{i=1}^n u_i \ln \frac{v_{t+1,i}}{v_{t,i}} \\
&= -\eta \sum_{i=1}^n u_i l_{t,i} - \sum_{i=1}^n u_{t,i} \ln Z_t \\
&= -\eta \boldsymbol{u}^T \boldsymbol{l}_t - \ln \sum_{i=1}^n v_{t,i} \exp(-\eta l_{t,i}) \\
&\ge -\eta \boldsymbol{u}^T \boldsymbol{l}_t - \ln \sum_{i=1}^n v_{t,i} \exp\left(-\eta l_{t,i} + \frac{1}{2}\eta^2 l_{t,i}^2\right) \\
&= -\eta \boldsymbol{u}^T \boldsymbol{l}_t - \ln \left(1 - \eta \boldsymbol{v}_t^T \boldsymbol{l}_t + \frac{1}{2} \sum_{i=1}^n v_{t,i} l_{t,i}^2\right) \\
&\ge \eta(\boldsymbol{v}_t^T \boldsymbol{l}_t - \boldsymbol{u}^T \boldsymbol{l}_t) - \frac{1}{2}\eta^2 \sum_{i=1}^n v_{t,i} l_{t,i}^2
\end{aligned}$$

The first inequality uses $\exp(-x) \ge 1 - x + x^2/2$ and the second inequality uses $\ln(1+x) \le x$. Now, let's consider the telescoping sum:

$$\begin{aligned}
\sum_{t=1}^m (\boldsymbol{v}_t^T \boldsymbol{l}_t - \boldsymbol{u}^T \boldsymbol{l}_t) &\le \frac{1}{\eta}(d(\boldsymbol{u}, \boldsymbol{v}_1) - d(\boldsymbol{u}, \boldsymbol{v}_{m+1})) + \frac{\eta}{2} \sum_{t=1}^m \sum_{i=1}^n v_{t,i} l_{t,i}^2 \\
&\le \frac{\ln n}{\eta} + \frac{\eta}{2} \sum_{t=1}^m \sum_{i=1}^n v_{t,i} l_{t,i}^2
\end{aligned}$$

This holds for all $\boldsymbol{u} \in \Delta_n$, it holds for a unit vector and we have the upper bound by noting that we have. $d(\boldsymbol{u}, \boldsymbol{v}_1) \le \ln n$ and $-d(\boldsymbol{u}, \boldsymbol{v}_{m+1}) \le 0$ and we have:

$$\sum_{t=1}^m \sum_{i=1}^n v_{t,i} l_{t,i}^2 \le m$$

We can set $\eta = \sqrt{2 \ln n / m}$, as we have proven the thoerem, as we can set $\eta = \sqrt{2 \ln n / m}$, which we have prove the thoerem. $\square$

5.3 Online Learning of Linear Classifier

Definition 5.8. (Problem) We have the sequence of data: $S = (\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m)$ and the total loss is given by $L)A(S)$. The regret is defined as:

$$L_A(S) - \inf_{\boldsymbol{u} \in \mathcal{U}} \text{Loss}_{\boldsymbol{u}}(S)$$

where $\mathcal{U}$ is a set of linear threshold function, as we will focus on the case where there exists $\boldsymbol{u} \in \mathcal{U}$ such that $\text{Loss}_{\boldsymbol{u}}(S) = 0$, which is a reliable case.

Definition 5.9. (Linear Threshold) The linear threshold $f_{\boldsymbol{u},b} : \mathbb{R}^n \to \{-1, 1\}$ function is:

$$f_{\boldsymbol{u},b}(\boldsymbol{x}) = \text{sgn}(\boldsymbol{u}^T \boldsymbol{x} + b)$$

The separating by hyperplane. The comparision class of all linear threshold function:

$$\mathcal{U}_{\text{it}} = \{f_{\boldsymbol{u},b} : \boldsymbol{u} \in \mathbb{R}^n, b \in \mathbb{R}\}$$

Remark 31. **(Assumption)** Data is linear separatable by some margin γ. Hence there exists a linear hyperplane with normal vector $\boldsymbol{v}$ such that: $\|\boldsymbol{v}\| = 1$ and for all $(\boldsymbol{x}_t, y_t)$, which we have $y_t \in \{-1, 1\}$, and $\|\boldsymbol{x}_t\| \le R$ and $y_t(\boldsymbol{x}_t^T \boldsymbol{v}) \ge \gamma$

Definition 5.10. (Perceptron Learning Algorithm) We consider the following learning algorithm

Algorithm 4 Perceptron Learning Algorithm

```
1: Initialize: $\boldsymbol{w}_1 = \boldsymbol{0}$ and $M_1 = 0$
2: for $i = 1, 2, \cdots, m$ do
3:     Receives Pattern $\boldsymbol{x}_t \in \mathbb{R}^n$
4:     Predict $\hat{y}_t = \text{sgn}(\boldsymbol{w}_t^T \boldsymbol{x}_t)$
5:     Receives Label $y_t$
6:     if Mistake $y_t \hat{y}_t \le 0$ then
7:         Update $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_t \boldsymbol{x}_t$
8:         $M_{t+1} = M_t + 1$
9:     else
10:        $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$ and $M_{t+1} = M_t$
11:    end if
12: end for
```

Lemma 5.1. *If* $(\boldsymbol{w}_t^T \boldsymbol{x}_t) y_t < 0$ *then* $\|\boldsymbol{w}_{t+1}\|^2 \le \|\boldsymbol{w}_t\|^2 + \|\boldsymbol{x}_t\|^2$

Proof. We have the following inequality:

$$\begin{aligned} \|\boldsymbol{w}_{t+1}\|^2 &= \|\boldsymbol{w}_t + y_t \boldsymbol{x}_t\|^2 \\ &= \|\boldsymbol{w}_t\|^2 + 2y_t(\boldsymbol{w}_t^T \boldsymbol{x}_t) + \|\boldsymbol{x}_t\|^2 \\ &\le \|\boldsymbol{w}_t\|^2 + \|\boldsymbol{x}_t\|^2 \end{aligned}$$

□

Lemma 5.2. $\|\boldsymbol{w}_t\|^2 \le M_t R^2$

Proof. Using induction, as we have:

- Base: $M_1 = 0$ and $\|\boldsymbol{w}_1\|^2 = 0$
- Induction step, when we have a mistake on the trial t as we have:

$$\|\boldsymbol{w}_{t+1}\|^2 \le \|\boldsymbol{w}_t\|^2 + \|\boldsymbol{x}_t\|^2 \le \|\boldsymbol{w}_t\|^2 + R^2 \le (M_t + 1)R^2$$

If there is no mistake, then the outcome is trivial as we have $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t$ and $M_{t+1} = M_t$ □

Lemma 5.3. $\|\boldsymbol{w}_t\|^2 \ge M_t \gamma$

Proof. Observe that $\|\boldsymbol{w}_t\| \ge \boldsymbol{w}_t^T \boldsymbol{v}$ because $\|\boldsymbol{v}\| = 1$ (via Cauchy-Schawarz). The prove of the lower bound $\boldsymbol{w}_t^T \boldsymbol{v}$ using the induction over t:

- Induction hypothesis $\boldsymbol{w}_t^T \boldsymbol{v} \ge M_t \gamma$
- Base $t = 1$, we have $\boldsymbol{w}_1^T \boldsymbol{v} = 0$
- Induction step: Assume for t and prove for $t + 1$, if there is a mistake as we have:

$$\begin{aligned} \boldsymbol{w}_{t+1}^T \boldsymbol{v} &= (\boldsymbol{w}_t + \boldsymbol{x}_t y_t)^T \boldsymbol{v} \\ &= \boldsymbol{w}_t^T \boldsymbol{v} + y_t \boldsymbol{x}_t^T \boldsymbol{v} \\ &\ge M_t \gamma + \gamma = (M_t + 1)\gamma \end{aligned}$$

This works in the case of non-mistake.

□

Theorem 5.4. *For all sequence of examples $S = (\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \mathbb{R}^n \times \{-1, +1\}$. This mistake of the* `PERCEPTRON` *algorithm is bounded by:*

$$M \leq \left(\frac{R}{\gamma}\right)^2$$

with $R = \max_t \|\boldsymbol{x}_t\|$. If there exists a vector $\boldsymbol{v}$ with $\|\boldsymbol{v}\| = 1$ and constant γ such that $(\boldsymbol{v}^T \boldsymbol{x}_t)y_t \geq \gamma$

Proof. We use the bound on the norm of the weight $\|\boldsymbol{w}_t\|$ as we have:

$$(M\gamma)^2 \leq \|\boldsymbol{w}_{t+1}\|^2 \leq MR^2$$

and the inequality follows. □

Remark 32. It is conveinece to express the bound in the form $M \leq R^2 \|\boldsymbol{u}\|^2$ where $\boldsymbol{u} = \boldsymbol{v}/\gamma$ then we have for all $\boldsymbol{u}$ such that $(\boldsymbol{u}^T \boldsymbol{x}_t)y_t \geq 1$.

Remark 33. **(Additional Problem)** Suppose that $\boldsymbol{w}_{m+1}$ doesn't necessary linearly separate S. How can we use the PERCEPTRON to define a vector $\boldsymbol{w}$ and how long that would take ?

Remark 34. **(Gradien Descent)** Recalling the regularization approach to supervised learning as we have:

$$h^* = \operatorname*{arg\,min}_{h \in \mathcal{H}} \sum_{t=1}^{m} L(y_t, h(\boldsymbol{x}_t)) + \lambda \text{ penalty}(h)$$

We consider the soft-margin SVM, which we have the following loss function:

$$\operatorname*{arg\,min}_{\boldsymbol{w} \in \mathbb{R}^n, b \in \mathbb{R}} \sum_{i=1}^{m} h_{\text{hi}}(y_t, \boldsymbol{w}^T \boldsymbol{x}_t + b) + \lambda \|\boldsymbol{w}\|^2$$

where $h_{\text{hi}}(y, \hat{y}) = \max(0, 1 - y\hat{y})$, which we can consider the followimg optimization problem:

$$\boldsymbol{w}_{t+1} = \operatorname*{arg\,min}_{\boldsymbol{w} \in \mathbb{R}^n} L_{\text{hi}}(y_i, \boldsymbol{w}^T \boldsymbol{x}_t) + \lambda \|\boldsymbol{w} - \boldsymbol{w}_t\|^2$$

Solving this problem, gives us:

$$\boldsymbol{w}_{t+1} = \begin{cases} \boldsymbol{w}_t & y_t(\boldsymbol{w}^T \boldsymbol{x}_t) > 1 \\ \boldsymbol{w}_t + \dfrac{y_t \boldsymbol{x}_t}{2\lambda} & y_t(\boldsymbol{w}^T \boldsymbol{x}_t) < 1 \end{cases}$$

Definition 5.11. (Online Gradient Descent) We consider the online gradient descent with hinge loss and $\|\cdot\|_2^2$ penalty, which we have the following pseudocode:

Algorithm 5 Online Gradient Descent

```
1: Initialize: $\boldsymbol{w}_1 = \mathbf{0}$ and $L_{\text{OGD}} = 0$
2: Select $\eta \in (0, \infty)$
3: for $i = 1, 2, \cdots, m$ do
4:     Receives instance $\boldsymbol{x}_t \in \mathbb{R}^n$
5:     Predict $\hat{y}_t = \boldsymbol{w}_t^T \boldsymbol{x}_t$
6:     Receives Lable $y_t \in \{+1, -1\}$
7:     Incur Loss $L_{\text{OGD}} = L_{\text{OGD}} + L_{\text{hi}}(y_t, \hat{y}_t)$
8:     Update weight $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \mathbb{I}[y_t\hat{y}_t < 1]\eta y_t \boldsymbol{x}_t$
9: end for
```

Theorem 5.5. *Given $R = \max_t \|\boldsymbol{x}_t\|$ and $\|\boldsymbol{u}\| \le U$. For the algorithm OGD with $\eta = U/(R\sqrt{m})$ as:*

$$\sum_{t=1}^m L_{hi}(y_t, \hat{y}_t) - L_{hi}(y_t, \boldsymbol{u}^T\boldsymbol{x}_t) \le \sqrt{U^2R^2m}$$

for any vector $\boldsymbol{u}$.

Proof. Using the convexity of the hinge loss (with respected to 2nd argument), which we have:

$$L_{\text{hi}}(y_t, \hat{y}_t) - L_{\text{hi}}(y_t, \boldsymbol{u}^T\boldsymbol{x}) \le (\boldsymbol{w}_t - \boldsymbol{u})^T\boldsymbol{z}_t$$

where $\boldsymbol{z}_t = -y_t\boldsymbol{x}_t[y_t(\boldsymbol{w}_t^T\boldsymbol{x}_t) < 1] \in \partial_{\boldsymbol{w}} h_{\text{hi}}(y_t, \boldsymbol{w}_t^T\boldsymbol{x})$. For the update, we have:

$$\begin{aligned}\|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 &= \|\boldsymbol{w}_t - \eta\boldsymbol{z}_t - \boldsymbol{u}\|^2 \\ &= \|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - 2\eta(\boldsymbol{w}_t - \boldsymbol{u})^T\boldsymbol{z}_t + \eta^2\|\boldsymbol{z}_t\|^2\end{aligned}$$

And so, we have the:

$$(\boldsymbol{w}_t - \boldsymbol{u}_t)^T\boldsymbol{z}_t = \frac{1}{2\eta}\left(\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 + \eta^2\|\boldsymbol{z}_t\|^2\right)$$

and so we have:

$$\begin{aligned}\sum_{t=1}^m (\boldsymbol{w}_t - \boldsymbol{u})^T\boldsymbol{z}_t &= \sum_{t=1}^m \frac{1}{2\eta}\left(\|\boldsymbol{w}_t - \boldsymbol{u}\|^2 - \|\boldsymbol{w}_{t+1} - \boldsymbol{u}\|^2 + \eta^2\|\boldsymbol{z}_t\|^2\right) \\ &\le \frac{1}{2\eta}\left(\|\boldsymbol{u}^2\| + \eta^2\sum_{t=1}^m\|\boldsymbol{z}_t\|^2\right) \\ &= \frac{1}{2\eta}\|\boldsymbol{u}\|^2 + \frac{\eta}{2}\sum_{t=1}^m\|\boldsymbol{x}_t\|^2\mathbb{I}[y_t(\boldsymbol{w}_t^T\boldsymbol{x}_t) < 1] \\ &\le \frac{1}{2\eta}U^2 + \frac{\eta}{2}mR^2 = \sqrt{U^2R^2m}\end{aligned}$$

as we have $\eta = U/(R\sqrt{m})$ and using the result from the convex setting yields the result. □

Remark 35. **(Perceptron Bound)** The perceptron bound can be arrived by using the analysis of the OGD above as we have:

- If we consider the hinge, we have:

$$\sum_{t=1}^m \mathbb{I}[y_t \ne \text{sgn}(\hat{y}_t)] - L_{\text{hi}}(y_t, \boldsymbol{u}^T\boldsymbol{x}_t) \ge \sqrt{U^2R^2m}$$

- Assuming that there is a linear classifier $\boldsymbol{u}$ such that $y_t(\boldsymbol{u}^T\boldsymbol{x}_t) \ge 1$ for all $t = 1, \dots, m$ as we have:

$$\sum_{t=1}^m \mathbb{I}[y_t \ne \text{sgn}(\hat{y}_t)] \ge \sqrt{U^2R^2m}$$

- Make OGD conservative that we only update when $y_t\hat{y}_t \le 0$ instead of $y_t\hat{y}_t \le 1$ as we have the trial when mistake is made.
- With respect to the bound, we can ignore the trial, which the mistake is made, so we can take the value $m = M := \sum_{t=1}^m \mathbb{I}[y_t \ne \text{sgn}(\hat{y}_t)]$, which implies that:

$$M \le \sqrt{U^2R^2M} \implies M \le U^2R^2$$

- We can set $\eta = 1$ as its number doesn't matter at all. Recall the update rule for the perceptron, when mistake is made, with learning rate η: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta y_t \boldsymbol{x}_t$, as we have:
$$\boldsymbol{w} = \sum_{m:\text{mistake}} \eta y_t \boldsymbol{x}_t$$
If $\eta > 0$, then we have $\eta \sum_{m:\text{mistake}} y_t \boldsymbol{x}_t$. Note that the prediction made by perceptron is based on the sign of the dot product, and so η doesn't take on any effect.

5.4 Disjunction Learning

Definition 5.12. (Boolean Function) The boolean function f may be represented as a map $f : \{0,1\}^n \to \{0,1\}$ where, we have the following:

- $x_1 \wedge x_2 = x_1 x_2$
- $x_1 \vee x_2 = \text{sign}(x_1 + x_2)$
- $\bar{x} = 1 - x$

Furthermore, we have the following addiitonal definiiton for the boolean function:

- Single variable is called a literal.
- Term or Conjuction is an iterated "and" applied.
- Clause or Disjunction is an iterated "or" applied.
- Monotone disjunction or conjuction implies no negated literal.

Remark 36. **(Naive Weighted-Majority)** The goal is to predict as well as k-literal (monotone) disjunction (over n variables). We can consider the use of weighted majority as each experts are disjunction of various variable and size. So, we have $\binom{n}{k}$ total expert and weights. This gives us the following bound:
$$\text{Mistake} \leq 2.63M + 2.63k \ln \frac{nl}{k}$$
where M is the mistakes of best disjunction, while we use the inequality
$$\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$$
It is clear that the time and space are exponent in k (run time). We need better algorithm.

Corollary 5.1. *With the feature map $\phi(\boldsymbol{x}) = (\boldsymbol{x}, 1)$, we use the perceptron to learn monotone disjunction:*
$$M \leq (4k+1)(n+1)$$
when k is the number of literal out of the n possible literal. And, so there exists a generic lower bound for rotation invariance algorithm (SVM and perceptron) where $M = \Omega(n)$

Proof. We use the following perceptron bound:
$$M \leq R^2 \left\|\boldsymbol{u}\right\|^2$$
for all $\boldsymbol{u}$ such that $(\boldsymbol{u}^T \boldsymbol{x}_t) y_t \geq 1$. With $\boldsymbol{x} \in \{0,1\}^n$, the the feature map $\boldsymbol{\phi}(\boldsymbol{x}) = (\boldsymbol{x}, 1)$, claim the following that $\boldsymbol{u}^* \in \mathbb{R}^{n+1}$ separate with margin of 1, where:
$$u_i^* = \begin{cases} 2 & i \text{ is a literal} \\ 0 & i \text{ isn't a literal} \\ -1 & i \text{ is biase weight} \end{cases}$$
Such that, we have the following calculation:

- $(\boldsymbol{u}^*)^T\boldsymbol{\phi}(\boldsymbol{x}) \geq 1$ as we have positive example $y_t = 1$
- $(\boldsymbol{u}^*)^T\boldsymbol{\phi}(\boldsymbol{x}) = -1$ as we have negative example $y_t = -1$

Note that for some $\boldsymbol{x} \in \{0,1\}^n$, then $\|\boldsymbol{\phi}(\boldsymbol{x})\|^2 \leq n+1$ and we have $\|\boldsymbol{u}^*\|^2 = 4k+1$, thus we have $M \leq (4k+1)(n+1)$ as required. ☐

Definition 5.13. (Winnow Algorithm) We define the winnow algorithm to be

Algorithm 6 Winnow Algorithm

```
1: Input: $(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_m, y_m) \in \{0,1\}^n \times \{0,1\}$
2: Initialize: $\boldsymbol{w}_1 = \mathbf{1}$
3: Select $\eta \in (0, \infty)$
4: for $i = 1, 2, \cdots, m$ do
5:     Receives instances $\boldsymbol{x}_t \in \{0,1\}^n$
6:     Predict the value:
           $$\hat{y}_t = \begin{cases} 0 & \boldsymbol{w}_t^T\boldsymbol{x}_t < n \\ 1 & \boldsymbol{w}_t^T\boldsymbol{x}_t \geq n \end{cases}$$
7:     Receives the label $y_t \in \{0,1\}$
8:     if Mistake $\hat{y}_t \neq y_t$ then
9:         Update the value:
               $$w_{t+1,i} = w_{t,i}2^{(y_t - \hat{y}_t)x_{t,i}} \qquad \text{for } i \in [n]$$
10:    end if
11: end for
```

Theorem 5.6. *The mistake of winnow is bounded by:*

$$M \leq 3k(\log n + 1) + 2$$

Proof. Let's consider 2 scenarios as we consider the bound on mistake:

- On a mistake, at least one element weight is doubled and the relevent weight never decreases.
- Once it the weight $w_{t,i} \geq n$, it will no longer change (it will saturated to n) and so the mistake is:
$$M_p \leq k(\log n + 1)$$
where M_p is the bound on the positive example $y_t = 1$

Let $W_t = \sum_{i=1}^n w_{t,i}$. We can see that $W_1 = n$, and so:

- On the positive mistake ($y_t = 1$) we have $W_{t+1} \leq W_t + n$ (as we can only double it)
- On the negative mistake ($y_t = 0$) we have $W_{t+1} \leq W_t - n/2$ as we can only half the number of weights.

Consider the progression of weights, we can see that:

$$0 \leq W_{m+1} \leq W_1 + M_pn - M_fn/2 = n + M_pn - M_fn/2$$

Thus, we have $M_f \leq 2k(\log n + 1) + 2$, where M_p is the bound on the positive example $y_t = 0$. Combining them and we have:

$$M \leq M_p + M_f \leq 2 + 3k(\log n + 1)$$

☐

Remark 37. There are several observation that we have to make:

- WINNOW is an improvement over PERCEPTRON in terms of dimension m in the mistake bound.
- The bound for linear threshold learning for the WINNOW is incompatible as the algorithm prefer sparse hypothesis.

Theorem 5.7. *Given m, let t drawn uniformly at random from $\{1, \dots, m\}$. Let S be set of t examples sampled from p. Let $(\boldsymbol{x}', y')$ be addiitonal example sample from P, then:*

$$\mathbb{P}(A_S(\boldsymbol{x}') \neq y') \leq \frac{B}{m}$$

with respected to be drawing of t, S and $(\boldsymbol{x}', y)$, where the mistake bound for A is B.

Proof. There are no more than B trials with mistake, therefore, since t is drawn uniformly from $\{1, \dots, m\}$ there is no more than B/m probability of hitting trial with a mistake. □

Definition 5.14. (Disjunctive Normal Form) DNF is a disjunction of terms, for example:

$$x_1x_4x_7 \vee x_1\bar{x}_2 \vee x_2x_5$$

All boolean function may be represented as DNF.

Remark 38. DNF corresponds to simple boolean network with a signle layer as such they may learn by a neural network with single hidden layer.

Definition 5.15. (ANOVA Kernel) We consider the feature map to be:

$$\boldsymbol{x} = \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix} \qquad \Phi(\boldsymbol{x}) = \begin{matrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \\ x_1x_2 \\ \vdots \\ x_1x_2\dots x_n \end{matrix}$$

THere are 2^n features. The k-terms DNF in input space is k-literal in feature space:

$$\boldsymbol{\Phi}(\boldsymbol{x})\boldsymbol{\Phi}(\boldsymbol{y}) = \prod_{i=1}^{n}(1 + x_iy_i) = k_{\text{anova}}(\boldsymbol{x}, \boldsymbol{y})$$

Please note that it also represent a disjunction normal form.

Remark 39. **(Perceptron for K-term DNF)** The weight of the perceptron, we have the weight to be:

$$\boldsymbol{w}_t = \sum_{q \in \text{mistakes}} \alpha_q \boldsymbol{\Phi}(\boldsymbol{x}_q)$$

And performing a predicting gives us:

$$\boldsymbol{w}_t^T \boldsymbol{\Phi}(\boldsymbol{x}_t) = \left(\sum_{q \in \text{mistakes}} \alpha_q \boldsymbol{\Phi}(\boldsymbol{x}_q) \right) \boldsymbol{\Phi}(\boldsymbol{x}_t) = \sum_{q \in \text{mistakes}} \alpha_q \boldsymbol{k}(\boldsymbol{x}_q, \boldsymbol{x}_t)$$

The prediction time complexity is $\mathcal{O}(n \cdot \#\text{mistakes}) \leq \mathcal{O}(nm)$. The mistake bound is $\mathcal{O}(k2^n)$

Remark 40. The winnow weight is given as:

$$w_{t,i} = \exp\left(-\eta \sum_{q\in\text{mistake}} \alpha_q [\Phi(\boldsymbol{x}_q)]_i\right)$$

The have the log of weights that is linear combination of the past examples. We have the mistake bound to be $\mathcal{O}(k \ln 2^n) = \mathcal{O}(kn)$ with the prediction time to be $\Omega(2^n \#\text{mistake})$, as there is no obvious fas way to compute $\boldsymbol{w}_t^T \boldsymbol{\Phi}(\boldsymbol{x}_t)$ for now.

6 Online Learning 2: Bandits

Definition 6.1. (Partial Feedback Protocal) We cosnider the following setting:

Algorithm 7 Partial Feedback Control

```
1: for $i = 1, 2, \cdots, T$ do
2:     Predict $\hat{y}_t \in [n]$
3:     Observe loss of prediction $l_{t,\hat{y}_t} \in [0, 1]$
4: end for
```

We have the following goal:

$$\sum_{t=1}^{m} l_{t,\hat{y_t}} - \min_{i\in[n]} \sum_{t=1}^{m} l_{t,i} \le o(m)$$

This is the same as the regret. Please note that we didn't get to see all loss function that is induced by the prediction.

Definition 6.2. (Unbiased Estimation) An estimator $\hat{\theta}$ estimate a parameter θ of a distribution from a sample is unbiased if we can show that $\mathbb{E}[\hat{\theta}] = \theta$.

Example 6.1. *Suppose $X_1, \dots, X_n$ are iid random variable for a distribution with mean μ, then:*

$$\hat{\theta} = \frac{1}{n}(X_1 + \cdots + X_n)$$

is an unbiased estimate of μ

Example 6.2. *Suppose X is a random variable with the discrete unifrom distribution over $\{1, \dots, n\}$. Suppose n is unknown and we wish to estimate it.*

- *The estimate $\hat{\theta}_1 = X$ is the maximum likelihood estimator, since $\mathcal{L}(\theta, X = x) = 1/\theta$ is maximized when $\theta = x$. Then we have:*

$$\mathbb{E}[\hat{\theta}_1; \theta = n] = \sum_{x=1}^{n} \frac{x}{n} = \frac{n+1}{2}$$

- *And so, $\hat{\theta}_2 = 2x - 1$ is unbiased estimator, which is:*

$$\mathbb{E}[\hat{\theta}_2; \theta = n] = \sum_{x=1}^{n} \frac{1}{n}(2x-1) = 2\sum_{x=1}^{n} \frac{1}{n}(2x-1) = 2\sum_{x=1}^{n} \frac{1}{n}x - \sum_{x=1}^{n} \frac{1}{n} = n$$

Remark 41. **(Assumption and Estimation)** Suppose, we have a distribution D_i over $[0, 1]$ for each $i \in [n]$ arms. For each arm i, we use iid sample $l_{t,i}$ for D_i. Suppose, we play i on trials $S_{t,i} \subseteq [t]$, then:

$$\hat{\mu}_{t,i} = \sum_{t\in S_i} \frac{l_{t,i}}{|S_i|}$$

This is unbiased estimator of μ_i. Now, we can consider the usage as we have:

- We can use a concentration inequality that allows us to quantitatively estimate the likelihood to estimate differently for the parameter.
- Using the observation, the algorithm UCB balances exploration and exploitation to obtain good regret bounds for this method.
- Suppose tha tthe underlying D_i is changing over time (being $D_{t,i}$):

$$\mu_{t,i} = \frac{\sum_{j=1}^{t} \mathbb{E}[l_{j,t}]}{t}$$

  where $S_i = [t]$. However, if we only have $S_{t,i} = [t]$, then we have no information about the other arms.
- We need to have simultaneous unbiased estimate for all arms S

Definition 6.3. (Importance Weighting) We have the following series of observation:

- Suppose X is a random variable over $\mathbb{R}$ with a mean μ. By definition, $\mathbb{E}[X] = \mu$ and $\hat{\theta}_1 = X$ is an unbiased estimator of the mean.
- Consider the biased coin Z_p with outcome 1 with probability p. Suppse, we have the estimator $\hat{\theta}_0$ setting to equal to X/p if $Z_p = 1$.
- Its expectation is equal to:

$$\mathbb{E}[\hat{\theta}_0] = \mathbb{P}(Z_p = 1)(X/p) + 0\mathbb{P}(Z_p = 0) = (p)(X/p) + (1-p)(0) = X$$

  This is unbiased.

Definition 6.4. (Hallucinated Loss Vector) We generalize this to obtain an unbiased estimator of l_t in the bandit setting. Given $\boldsymbol{v}_t \in \Delta_n$ by the relation tha t$\hat{y}_t \sim \boldsymbol{v}_t$. The unbiased estimator l_t^n or $\boldsymbol{k}_t$ with respected to $\boldsymbol{v}_t$ is given as:

$$\left(l_{t,i}^h = \frac{l_{t,i}}{v_{t,i}} \mathbb{I}[i = \hat{y}_t] \right)_{i \in [n]}$$

Remark 42. **(Expectation of Hallucinated Loss Vector)** Observed that l_t^h is unbiased for all $i \in [n]$ since we have:

$$\mathbb{E}_{\hat{y}_t, \boldsymbol{v}_t}[l_{t,i}^h] = \sum_{j=1}^{n} v_{t,j} \frac{l_{t,i}}{v_{t,i}} \mathbb{I}[i = j] = l_{t,i}$$

We have unbiased estimator for all arms by only observing the single arm. We can apply the hedge to l_t^h requires bounded loss vector. We can use more careful analysis of the hedge.

Definition 6.5. (EXP3) Exponential-Weight algorithm for Exploration and Exploitation is given by:

Algorithm 8 EXP3

```
1: Initialize: $\eta \in (0, \infty)$
2: Set $\boldsymbol{v}_1 = (1/n, \ldots, 1/n)$
3: for $i = 1, 2, \cdots, T$ do
4:     Sample $\hat{y}_t \sim \boldsymbol{v}_t$
5:     Observe Loss $l_{t,\hat{y}} \in [0, 1]$
6:     Construct Hallucinated Loss vector:
           $l_t^h = \left( l_{t,i}^h = \frac{l_{t,i}}{v_{t,i}} \mathbb{I}[i = \hat{y}_t] \right)_{i \in [n]}$
7:     Perform the update, for $i \in [n]$ and $Z_t = \sum_{i=1}^{n} v_{t,i} \exp(-\eta l_{t,i}^h)$:
           $v_{t+1,i} = v_i \exp(-\eta l_{t,i}^h) / Z_t$
8: end for
```

Lemma 6.1. *For any sequence of loss vector $\boldsymbol{l}_1, \dots, \boldsymbol{l}_m \in [0,1]^n$, we have the following loss bound:*

$$\sum_{t=1}^{m} \boldsymbol{v}_t^T \boldsymbol{l}_t^h - \sum_{t=1}^{m} \boldsymbol{u}^T \boldsymbol{l}_t^h \leq \frac{\ln n}{\eta} + \frac{\eta}{2} \sum_{t=1}^{m} \sum_{i=1}^{n} v_{t,i} (l_{t,i}^h)^2$$

For all $\boldsymbol{u} \in \Delta_n$

Proof. The lemma follows from the fact that EXP3 is just Hedge with $\boldsymbol{l}_t$ weighted to be $\boldsymbol{l}_t^h$ and the Hedge inequality is proven before. □

Remark 43. We can show the property of EXP3, where we consider that: we need to perform and so we may replace hallucination losses $\boldsymbol{l}_t^h$ with time loss $\boldsymbol{l}$:

- We can model some of the randomness as we use the adversarial loss $\boldsymbol{l}_1, \dots, \boldsymbol{l}_m$.
- We have to bound the term $\sum_{t=1}^m \sum_{i=1}^n v_{t,i}(l_{t,i}^h)^2$ and tune η

Definition 6.6. (Deterministic Adversarial Model) We will to set $\boldsymbol{l}_1, \dots, \boldsymbol{l}_m$ before running the algorithm. The adversary is assumed to be complete given the prior knowledge, and:

- The limitation of near omniscient adversary is that it is non-adaptive.
- It many simulate the stochastic model by repeatedly sample the $\mathcal{D}_1, \dots, \mathcal{D}_m$ in advance.

Theorem 6.1. *For any sequence of loss vector $S = l_1, \dots, l_m \in [0,1]^n$, the regret for EXP3 with $\eta = \sqrt{2 \ln n / mn}$ is:*

$$\mathbb{E}[L_A(S)] - \min_i L_i \leq \sqrt{2mn \ln n}$$

where $L_A(S) = \sum_{t=1}^m l_{t,\hat{y}_y}$ and $L_i = \sum_{t=1}^m l_{t,i}$

Proof. Observe that the only source of randomness are the sample $\hat{y}_t \sim \boldsymbol{v}_t$. As previously argue, note that $\mathbb{E}[l_{t,i}^h] = l_{t,i}$, and we have:

$$\mathbb{E}[\boldsymbol{v}_t^T \boldsymbol{l}_t^h] = \sum_{i=1}^n \mathbb{E}[v_{t,i} l_{t,i}^h] = \sum_{i=1}^n v_{t,i} \mathbb{E}[l_{t,i}^h] = \sum_{i=1}^n v_{t,i} l_{t,i} = \mathbb{E}[l_{t,\hat{y}_t}]$$

Similarly, we have:

$$\mathbb{E}[(l_{t,i}^h)^2] = \sum_{j=1}^n v_{t,j} \left(\frac{l_{t,i}}{v_{t,i}} \right)^2 \mathbb{I}[i=j]^2 = v_{t,i} \left(\frac{l_{t,i}}{v_{t,i}} \right)^2 = \frac{l_{t,i}^2}{v_{t,i}}$$

This implies that:

$$\mathbb{E}\left[\sum_{i=1}^n v_{t,i} (l_{t,i}^h)^2 \right] = \sum_{i=1}^n v_{t,i} \frac{l_{t,i}^2}{v_{t,i}} = \sum_{i=1}^n l_{t,i}^2 \leq n$$

Taking the expectation over the Hedge terms, and we have for $\boldsymbol{u} \in \Delta_n$:

$$\mathbb{E}\left[\sum_{t=1}^m \boldsymbol{v}_t^T \boldsymbol{l}_t^h - \sum_{t=1}^m \boldsymbol{u}^T \boldsymbol{l}_t^h \right] \leq \mathbb{E}\left[\frac{\ln n}{\eta} + \frac{\eta}{2} \sum_{t=1}^m \sum_{i=1}^n v_{t,i} (l_{t,i}^h)^2 \right]$$

And, so we have using the fact that: $\mathbb{E}[l_{t,i}^h] = l_{t,i}$, and the previous result with $\boldsymbol{u}$ being a coordinate vector, we have:

$$\mathbb{E}\left[\sum_{t=1}^m \boldsymbol{v}_t^T \boldsymbol{l}_t^h \right] - \min_i \mathbb{E}\left[\sum_{t=1}^m \boldsymbol{l}_{t,i}^h \right] \leq \frac{\ln n}{\eta} + \frac{\eta}{2} \mathbb{E}\left[\sum_{t=1}^m \sum_{i=1}^n v_{t,i} (l_{t,i}^h)^2 \right]$$

And, so we have:

$$\mathbb{E}[L_A(S)] - \min_i L_i(S) \leq \ln \frac{n}{\eta} + \frac{\eta}{2} mn$$

Substuite the $\eta = \sqrt{2 \ln n / mn}$ to prove this theorem. □

7 Learning Theory

7.1 Introduction

Definition 7.1. (Distribution over Subset) If $\mathcal{D}$ is a distribution over $\mathcal{Z}$ then if $A \subseteq \mathcal{Z}$ then $\mathcal{D}(A)$ denotes the probability that if z is drawn from $\mathcal{D}$ that $z \in A$

Definition 7.2. (Expected Error) Data is sampled iid from a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} = \{0, 1\}$. The expected error of function $h : \mathcal{X} \to \mathcal{Y}$ is:

$$L_{\mathcal{D}}(h) = \mathcal{D}(\{x, y\} : h(x) \neq y) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] = \int \mathbb{I}[h(x) \neq y] \ \mathrm{d}p(x, y)$$

where we denote $L_{\mathcal{D}}(h) = \mathcal{E}(h)$

Definition 7.3. (Empirical Error) The empirical error of h given the dataset $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\}$ is denoted as:

$$L_S(h) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[h(x_i) \neq y_i]$$

Or, we denote it as $\mathcal{E}_{\text{emp}}(S, h)$.

Theorem 7.1. ***(Hoeffding's Inequality)*** *Let $Z_1, Z_2, \ldots, Z_m$ be iid bernoulli random variable, when for all i, we have $\mathbb{P}(Z_i = 1) = p$ and let $\bar{Z} = 1/m\sum_{i=1}^{m} Z_i$, then for any $\varepsilon > 0$ as we have:*

$$\mathbb{P}(\bar{Z} > p + \varepsilon) \leq \exp(-2m\varepsilon^2) \qquad \mathbb{P}(\bar{Z} < p - \varepsilon) \leq \exp(-2m\varepsilon^2)$$

Theorem 7.2. *Select a function h then for any $\delta \in (0, 1)$ with probability $1 - \delta$ over the random sample V of size m from $\mathcal{D}$, we have:*

$$L_{\mathcal{D}}(h) \leq L_V(h) + \sqrt{\frac{\ln(1/\delta)}{2m}}$$

The generalization error of a function h may be bounded by the empirical error. We may select a predictor h on any set S, as we may bound it on the validation on separate set of data V.

Proof. Given a predictor h, we have the differences to be:

$$L_{\mathcal{D}}(h) - L_V(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x) \neq y] - \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}[h(x_i) \neq y_i]$$

we can define $Z_i = \mathbb{I}[h(x_i) \neq y_i]$. We can see that $Z_1, \ldots, Z_m$ are statistical independent. Then for all $\mathbb{P}[Z_i] = L_{\mathcal{D}}(h) = \mathbb{P}[h(x) \neq y]$. We apply the Hoeffding inequality, gives us:

$$\mathbb{P}[L_{\mathcal{D}}(h) - L_V(h) \geq \varepsilon] \leq \exp(2 - \varepsilon^2)$$

setting $\delta = \exp(-2\varepsilon^2 m)$, and solving this gives us the theorem. $\square$

Remark 44. If we use the upper and lower bound m, the Hoeffding inequality would gives us:

$$|L_{\mathcal{D}}(h) - L_V(h)| \leq \sqrt{\frac{\ln(2/\delta)}{2m}}$$

This is a nice result, but there are some drawnbacks to this bound:

- The validation bound gives a way to estimate of the confidence interval for the generalization error. The data V can't be used for training.
- Having small number of data, can we choose a model based on the expected error directly, without the training data ?

- The bound is about the predictor, while we need to analyze the prediction done by the machine learning algorithm.

Definition 7.4. (General Statistical Consider) Statistical model begin with an assumption that the data is generated by the underlying distribution $\mathcal{D}$ not known to the learner. Assuming that we are given a training set that is generated iid from distribution $\mathcal{D}$:

$$S = \{(x_1, y_1), \dots, (x_m, y_m)\}$$

Definition 7.5. (Empirical Risk Mimization) Assuming we have a learning algorithm A that chooses a hypothesis function $A_{\mathcal{H}}(S)$ from a hypothesis space $\mathcal{H}$ in response to the training set S. We study the ERM:

$$\text{ERM}_{\mathcal{H}}(S) = \underset{h\in\mathcal{H}}{\arg\min}\ L_S(h)$$

There are many possible empirical minimizer as we assume ERM to be an arbitrary one.

Remark 45. The traditional statistic h_S concentrated on analysing:

$$\lim_{m\to\infty} \mathbb{E}_{S_m}[L_{\mathcal{D}}(A(S_m))]$$

where S_m denotes a training set of size m. For finite sample, the generalization $L_D(A(S_m))$ has a distribution depending on the algorithm and function class and sample size:

- *Traditional Statistic*: concentrated on the mean of this distribution but this quantity is misleading for example in the case of low fold cross-validation.
- *Statistical Learning Theory*: analyze the tail of the distribution finding and the bound that holds in high probability.

Definition 7.6. (Reliability Assumption) Assume that there exists a function f^* so that for all $x \in \mathcal{X}$, we have $f^*(x) = y$ there exists a classifier that has zero error. We can now take $\mathcal{D}$ to be only a distribution over $\mathcal{X}$ only. We consider the following loss:

$$L_{\mathcal{D},f^*}(h) = \mathbb{P}[h(x) \neq f^*(x)]$$

We can find the algorithm A so that $h = A(S)$ such that $L_{\mathcal{D},f^*}(h) = 0$ is small.

Remark 46. **(Reason for Approximation)** We can't hope the find the function h such that $L_{\mathcal{D},f^*}(h) = 0$ Let's consider the $\varepsilon \in (0,1)$ that takes $\mathcal{X} = \{x_1, x_2\}$ where $\mathcal{D}(\{x_1\}) = 1-\varepsilon$ and $\mathcal{D}(\{x_2\}) = \varepsilon$:

- The probability to not see x_2 at all among m iid example is $(1-\varepsilon)^m \approx \exp(-\varepsilon m)$
- If $\varepsilon \ll 1/m$, we are unlikely to see x_2 at all. then we don't know its label.

So, we are only happy to see $L_{\mathcal{D},f^*}(h) \le \varepsilon$ when ε is user defined.

Remark 47. **(Reason for Probability)** The input is randomly generated (there is a small chance that we will see the same sample over and over again). No algorithm can generate $L_{\mathcal{D},f^*}(h) \le \varepsilon$ for sure, and so we allow the algorithm to fail with some probability $\delta \in (0,1)$ that is user defined.

Definition 7.7. (PAC Learning) The learner doesn't know $\mathcal{D}$ and f^*. It receives parameter ε and δ. Learner can aske for training data S contrary for $m(\varepsilon,\delta)$ examples. The learner should output a hypothesis h such that with at least probability $1-\delta$, it holds that $L_{\mathcal{D},f^*}(h) \le \varepsilon$

Theorem 7.3. ***(No Free Lunch)***

- *Suppose $|\mathcal{X}| = \infty$. For any fixed $C \subset \mathcal{X}$ take $\mathcal{D}$ to be uniform m distribution over C:*
- *If the number of training example is $m \le |C|/2$, the learner has no knowledge of at least half of elements in C.*

Fix $\delta \in (0,1)$ and $\varepsilon < 1/2$. For any learner A and training set of size m, there exists $\mathcal{D}$ and f^* such that with probability δ over the generation of a training data S of m examples, it holds that

$$L_{\mathcal{D},f^*}(A(S)) \geq \varepsilon$$

Proof. Consider for contradiction, assuming that the class is learnable, consider $\varepsilon > 1/8$ and $\delta \leq 1/7$. With the definition of PAC learnable $m(\varepsilon, \delta) = m$:

- For the consistent case, with probability greater than $1 - \delta$, when A is applied to sample S of size m, generated iid $\mathcal{D}$, we have
$$L_{\mathcal{D},\{^*}(A(S)) \leq \varepsilon$$
- However, using the NFL thoerem above, since $|\mathcal{X}| > 2m$, for every learning algorithm, there exists a $\mathcal{D}$ such that with probability greater than $1/7 > \delta$, and $L_{\mathcal{D},\{^*}(A(S)) > 1/8 > \varepsilon$

This is a contradiction. □

7.2 PAC of Finite Hypothesis Class

Lemma 7.1. *For any 2 sets A and B, and a distribution $\mathcal{D}$ we can show that:*

$$\mathcal{D}(A \cup B) \leq \mathcal{D}(A) + \mathcal{D}(B)$$

Theorem 7.4. *Fix ε, δ. If we have $m \geq \log(|\mathcal{H}|/\delta)/\varepsilon$, then for every $\mathcal{D}, f^*$ with probability of at least $1-\delta$ (with respected to randomly sample training set S of size m), we now have:*

$$L_{\mathcal{D},f^*}(ERM_{\mathcal{H}}(S)) \leq \varepsilon$$

This mean that we have $L_{\mathcal{D},f^*}(ERM_{\mathcal{H}}(S)) \leq (\log|\mathcal{H}| + \log(1/\delta))/m$. The generalization error decrease linear in the number of samples and increase in logarithm in the size of hypothesis class.

Proof. Consider $S|_x = (x_1, \dots, x_m)$ be instances of training set. We will show that:

$$\mathcal{D}^m(\{S|_x : L_{\mathcal{D},f^*}(\text{ERM}_{\mathcal{H}}(S)) > \varepsilon\}) \leq \delta$$

Let $\mathcal{H}_B$ be a set of bound hypothesis as we have $\mathcal{H}_B = \{h \in \mathcal{H} : L_{\mathcal{D},f^*}(h) > \varepsilon\}$ and let M be the set of misleading samples: $\{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\}$ Observe that:

$$\{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\} \subseteq M = \bigcup_{h \in \mathcal{H}_B} \{S|_x : L_S(h) = 0\}$$

Applying the union bound as we have the following union bound:

$$\begin{aligned}
&\mathcal{D}^m(\{S|_x : \exists h \in \mathcal{H}_B, L_S(h) = 0\}) \\
&\leq \sum_{h \in \mathcal{H}_B} \mathcal{D}^m(\{S|_x : L_S(h) = 0\}) \\
&\leq |\mathcal{H}_B| \max_{h \in \mathcal{H}_B} \mathcal{D}^m(\{S|_x : L_S(h) = 0\}) \\
&< |\mathcal{H}_B|(1-\varepsilon)^m \leq |\mathcal{H}| \exp(-\varepsilon m)
\end{aligned}$$

Observe that $\mathcal{D}^m(\{S|_x : L_S(h) = 0\}) = (1 - L_{\mathcal{D},f^*}(h))^m$ if $h \in \mathcal{H}_B$, then $L_{\mathcal{D},f^*}(h) \geq \varepsilon$. This leads to the third inequality, while the last inequality, we have: $1 - \varepsilon \leq \exp(-\varepsilon)$ and $|\mathcal{H}_B| \leq |\mathcal{H}|$. Setting the rhs to $\leq \delta$ and we get the required inequality. □

Definition 7.8. (PAC-Lernability) A hypothesis class $\mathcal{H}$ is PAC-learnable if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and has the property of that for every ε and $\delta \in (0,1)$ and every distribution $\mathcal{D}$ over $\mathcal{X}$ and for every labeling function $f^* : \mathcal{X} \to \{0,1\}$.

- Using the training algorithm $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$ iid examples generated by $\mathcal{D}$ and labeled by f^*
- The algorithm returns a hypothesis h such that with probability of at least $1-\delta$, the loss is $L_{\mathcal{D},f^*}(h) \leq \varepsilon$.
- We call $m_{\mathcal{H}}$ is the sample complexity of the training hypothesis $\mathcal{H}$

Remark 48. We are now interested in the infinite hypothesis space. What is the sample complexity of a given class ? Is there a generic algorithm that achieves the optimal sample complexity ?

Remark 49. **(VC-Dimension: Motivation)** Suppose, we have the training set: $S = (x_1, y_1), \dots, (x_m, y_m)$. We try to explain the label using a hypothesis from $\mathcal{H}$. We may get difference labels:

$$(x_1, y_1'), \cdots, (x_m, y_m')$$

We can try to explain the label using a hypothesis from $\mathcal{H}$. If this works for us, no matter the labels are then, no free-lunch thoerem apply, as now we can't learn from $m/2$ example.

Definition 7.9. (VC-Dimension) Let $C = \{x_1, \dots, x_{|C|}\} \subset \mathcal{X}$. Let $\mathcal{H}_C$ is the restriction of $\mathcal{H}$ to C, then we have:

$$\mathcal{H}_C = \{h_C : h \in \mathcal{H}\} \qquad \text{where} \qquad h_C : C \to \{-1, 1\}$$

is such that $h_C(x_i) = h(x_i)$. For every $x_i \in C$, we can represent each h_C as the vector:

$$\mathcal{H}_C = \left\{(h(x_1), \dots, h(x_{|C|})) \in \{-1, 1\}^{|C|}\right\}$$

and so we have $|\mathcal{H}_C| \leq 2^{|C|}$. We say that $\mathcal{H}$ shatters C if $|\mathcal{H}_C| = 2^{|C|}$ where we have:

$$\text{VCDim}(\mathcal{H}) = \sup\{|C| : \mathcal{H} \text{ shatters } C\}$$

VC dimension is the maximum size of a set C such that $\mathcal{H}$ gives no prior knowledge with respected to C.

Remark 50. To show that the VC dimension $\text{VCDim}(\mathcal{H}) = d$, we have to show that:

- There exists a set C of size d which is shattered by H
- Every set C of size $d+1$ isn't shattered by $\mathcal{H}$

Proposition 7.1. ***(VC-Dimension of Intervals)*** *Interval where we have $\mathcal{H} = \mathbb{R}$ and*

$$\mathcal{H} = \{h_{a,b} : a < b \in \mathbb{R}\}$$

where $h_{a,b}(x) = 1$ iff $x \in [a, b]$. Its VC-Dimension is 2.

Proposition 7.2. ***(Axis Aligned Rectangle)*** *We have $\mathcal{X} = \mathbb{R}^2$ as we have the hypothesis set to be:*

$$\mathcal{H} = \left\{h_{(a_1,a_2,b_1,b_2)} : a_1 < a_2 \text{ and } b_1 < b_2\right\}$$

where we have $h_{(a_1,a_2,b_1,b_2)}(\boldsymbol{x}_1, \boldsymbol{x}_2) = 1$ iff $\boldsymbol{x}_1 \in [a_1, a_2]$ and $\boldsymbol{x}_2 \in [b_1, b_2]$. We can show that $\text{VCDim}(\mathcal{H}) = 4$

Proof. We can find 4 points that can be shattered by H, and so $\text{VCDim}(\mathcal{H}) \geq 4$. For any poitn $C \subseteq \mathbb{R}^2$ with 5 points with label $(1, 1, 1, 1, 0)$ where 0 is the point in the middle, we can't obtain any axis aligned rectangle, thus it can't be shattered C. Therefore, $\text{VCDim}(\mathcal{H}) = 5$ □

Proposition 7.3. ***(Finite Class)*** *The VC-Dimension of the finite $\mathcal{H}$ is at most $\log_2(|\mathcal{H}|)$ as there can arbitrary gaps between $\text{VCDim}(\mathcal{H})$ and $\log_2(|\mathcal{H}|)$*

Proof. Let $\mathcal{H}$ be a finite class, for any set C that can be shattered, we have $2^{|C|} = |\mathcal{H}_C| \leq |\mathcal{H}|$, thus the upperbound of the VC dimension is $\log_2 |\mathcal{H}|$ □

Theorem 7.5. ***(Radon)*** *Any set $\mathcal{X}$ of $d+2$ data point $\mathbb{R}^d$ can be partion into 2 sets $\mathcal{X}_1$ and $\mathcal{X}_2$ such that the convex hull of $\mathcal{X}_1$ and $\mathcal{X}_2$ intersect.*

Proof. Let $\mathcal{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{d+2}\} \subset \mathbb{R}^d$ with the following linear equation:

$$\sum_{i=1}^{d+2} \alpha_i \boldsymbol{x}_i = 0 \qquad \sum_{i=1}^{d+2} \alpha_i = 0$$

The number of unknown $d+2$ is larger than the number of equations $d+1$. This implies that the system admits non-zero solution $\beta_1, \dots, \beta_{\alpha+2}$ since $\sum_{i=1}^{d+2} \beta_i = 0$ both:

$$\mathcal{J}_1 = \{i \in [d+2] : \beta_j > 0\} \qquad \mathcal{J}_2 = \{i \in [d+2] : \beta_j \le 0\}$$

This means that $\mathcal{X}_1 = \{x_i : i \in \mathcal{J}_1\}$ and $\mathcal{X}_2 = \{x_i : i \in \mathcal{J}_2\}$ form a partition. The last equation gives us:

$$\sum_{i \in \mathcal{J}_1} \beta_i = -\sum_{i \in \mathcal{J}_2} \beta_j$$

Let $\beta = \sum_{i \in \mathcal{J}_1} \beta_i$, then the first equation implies that:

$$\sum_{i \in \mathcal{J}_1} \frac{\beta_i}{\beta} \boldsymbol{x}_i = -\sum_{i \in \mathcal{J}_2} \frac{\beta_i}{\beta} \boldsymbol{x}_i$$

Please note that: $\sum_{i \in \mathcal{J}_1} \beta_i/\beta = -\sum_{i \in \mathcal{J}_2} \beta_i/\beta = 1$ and $\beta_i/\beta \ge 0$ for $i \in \mathcal{J}_1$ and $-\beta_j/\beta \ge 0$ for $i \in \mathcal{J}_2$. By the definition of the convex hull, this implies that $\sum_{i \in \mathcal{J}_1} \beta_i/\beta \boldsymbol{x}_i$ being both to convex hull $\mathcal{X}_1$ and $\mathcal{X}_2$ $\square$

Proposition 7.4. ***(Hyperplane)*** *We have* $\mathcal{X} = \mathbb{R}^n$ *and the hypothesis class to be:*

$$\mathcal{H} = \{y \mapsto \text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) : \boldsymbol{w} \in \mathbb{R}^n\}$$

Then, we have $\text{VCDim}(\mathcal{H}) = n + 1$

Proof. Starting with the lower bound, setting $\boldsymbol{x}_0$ to be the origin and setting $\boldsymbol{x}_i$ for $i \in [d]$ as the whose i coordinate to be 1 and all the others are 0.

- Let $y_0, y_1, \dots, y_d \in \{-1, 1\}$ be an arbitrary set of label.
- Let $\boldsymbol{w}$ be the vector whose i-th coordinate is y_i.

The classifier defined by the hyperplane of equation $\boldsymbol{w}^T\boldsymbol{x} + y_0/2 = 0$, shatters $\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_d$ we can see that for any $i \in \{0, \dots, d\}$ as we have:

$$\text{sgn}\left(\boldsymbol{w}^T\boldsymbol{x}_i + \frac{y_0}{2}\right) = \text{sgn}\left(y_i + \frac{y_0}{2}\right) = y_i$$

For the upperbound, let $\mathcal{X}$ be set of $d+2$ points. By Radon's theorem, it can be partition into 2 sets $\mathcal{X}_1$ and $\mathcal{X}_2$ such that the convex hull intersects. When the set of points $\mathcal{X}_1$ and $\mathcal{X}_2$ are separated by hyperplane, the convex hull also separated. However, it is a contradiction and so the VC dimension is proven. $\square$

Definition 7.10. (Inner Production Space) The space is the bounded sequence summable square:

$$l_2 = \left\{\boldsymbol{x} \in \mathbb{R}^\infty : \sum_{i=1}^\infty x_i^2 < \infty\right\}$$

with the inner product to be $\{\boldsymbol{x}, \boldsymbol{x}'\} = \sum_{i=1}^\infty x_i x_i'$

Definition 7.11. (Large Margin Halfspaces) Given $\mathcal{X} \subset l_2$ and $\Lambda \in (0, \infty)$, which we define:

$$\mathcal{H}_{\mathcal{X},\Lambda} = \{\boldsymbol{x} \mapsto \text{sgn}(\langle \boldsymbol{w}, \boldsymbol{x} \rangle) : \boldsymbol{x} \in \mathcal{X}, \boldsymbol{w} \in l_2, \|\boldsymbol{w}\| \le \Lambda, \langle \boldsymbol{w}, \boldsymbol{x} \rangle \ge 1\}$$

Observe that $1/\|\boldsymbol{w}\|$ is the margin.

Theorem 7.6. *We can show that for large margin halfspace:*

$$\text{VCDim}(\mathcal{H}_{\mathcal{X},\Lambda}) \le \Lambda^2 \max \|\boldsymbol{x}\|^2_{\boldsymbol{x}\in\mathcal{X}}$$

Theorem 7.7. ***(Fundamental Theorem of Statistical Learning)*** *Let $\mathcal{H}$ be a hypothesis class of binary classifier. Then there are absolute constant C_1 and C_2 such that the sample complexity is given by:*

$$C_1\frac{\text{VCDim}(\mathcal{H}) + \log(1/\delta)}{\varepsilon} \le m_{\mathcal{H}}(\varepsilon,\delta) \le C_2\frac{\text{VCDim}(\mathcal{H})\log(1/\varepsilon) + \log(1/\delta)}{\varepsilon}$$

This sample complexity is achieved by ERM learning rule.

7.3 Agnostic PAC-Learning

Remark 51. **(Motivation for Agnostic PAC)** Assuming that there exists f^* may be too strong, so we relaxed the notation, so we use the assumption that the joint distribution $\mathcal{D}$ be a distribution over $\mathcal{X}\times\mathcal{Y}$ as now we are going to use:

$$L_{\mathcal{D}}(h) = \mathbb{P}_{(x,y)\sim\mathcal{D}}[h(x)\neq y] := \mathcal{D}(\{(x,y) : h(x)\neq y\})$$

We will redefine the approximately correct notion.

Definition 7.12. (General Agnostic PAC) A hypothesis class $\mathcal{H}$ is agnostic PAC learnable if there exists a function $m_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm A with the following properties: for every $\delta,\varepsilon\in(0,1)$ and $m > m_{\mathcal{H}}(\varepsilon,\delta)$:

$$\mathcal{D}^m\left(\left\{S\in(\mathcal{X}\times\mathcal{Y})^m : L_{\mathcal{D}}(A(S)) \le \min_{h\in\mathcal{H}} L_D(h) + \varepsilon\right\}\right) \ge 1-\delta$$

Definition 7.13. ($\boldsymbol{\varepsilon}$-Representation Sample) A training set S is called ε-representative if for all $h\in\mathcal{H}$ as we have:

$$|L_S(h) - L_{\mathcal{D}}(S)| \le \varepsilon$$

Lemma 7.2. *Assume that a training set S is $\varepsilon/2$-representative, then the average output of* $\text{ERM}_{\mathcal{H}}(S)$ *namely $h_S \in \arg\min_{h\in\mathcal{H}} L_S(h)$ satsifies:*

$$L_{\mathcal{D}}(h_S) \le \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon$$

Proof. For every $h\in\mathcal{H}$ as we have:

$$L_{\mathcal{D}}(h_S) \le L_S(h_S) + \frac{\varepsilon}{2} \le L_S(h) + \frac{\varepsilon}{2} \le L_{\mathcal{D}}(h) + \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = L_{\mathcal{D}}(h) + \varepsilon$$

□

Definition 7.14. (Uniform Convergence) Let $\mathcal{H}$ has the uniform convergence if there exists a function $m^{\text{UC}}_{\mathcal{H}} : (0,1)^2 \to \mathbb{N}$ such that for every $\varepsilon,\delta\in(0,1)$ and every distribution $\mathcal{D}$, and we have:

$$\mathcal{D}^m\left\{\Big(S\in Z^m : S \text{ is } \varepsilon-\text{representable}\Big)\right\} \ge 1-\delta$$

where Z is the domain and $m \ge m^{\text{UC}}_{\mathcal{H}} : (0,1)^2$

Corollary 7.1. *From the definition of uniform convergence, we can show that:*

- *If $\mathcal{H}$ has uniform convergence property with a function $m^{UC}_{\mathcal{H}}$ then $\mathcal{H}$ is agnostic PAC learnable with sample complexity of*

  $$m_{\mathcal{H}}(\varepsilon,\delta) \le m^{UC}_{\mathcal{H}}(\varepsilon/2,\delta)$$

  This follows from the lemma above.

- *We can show that* $\text{ERM}_{\mathcal{H}}$ *is successful against PAC learner for* $\mathcal{H}$.

Theorem 7.8. *Assume $\mathcal{H}$ is finite, then $\mathcal{H}$ is agnostic PAC learnable using $ERM_{\mathcal{H}}$ algorithm with:*

$$m_{\mathcal{H}}(\varepsilon, \delta) = \left\lceil \frac{2\log(2\left|\mathcal{H}\right|/\delta)}{\varepsilon^2} \right\rceil$$

Comparing the reliable case generalization, the error will decrease in $\sqrt{m}$ values as oppose to linear.

Proof. It suffices to show that $\mathcal{H}$ has the uniform convergence property with:

$$m_{\mathcal{H}}^{\text{UC}}(\varepsilon, \delta) \leq \left\lceil \frac{2\log(2\left|\mathcal{H}\right|/\delta)}{\varepsilon^2} \right\rceil$$

To show that the uniform convergence, we need to show that:

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) \leq \delta$$

Using the union bound, we can see that:

$$\begin{aligned}\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon\}) \leq \delta = \mathcal{D}^m &\left(\bigcup_{h\in\mathcal{H}} \{S : |L_S(h) - L_{\mathcal{D}}(h)|\} \geq \varepsilon\right) \leq \delta \\ &\leq \sum_{h\in\mathcal{H}} \mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| \geq \varepsilon\}) \leq \delta \\ &\leq 2\left|\mathcal{H}\right| \exp(-2m\varepsilon^2)\end{aligned}$$

The last inequality is shown by Hoeffding inequality, setting the correc m, to finish the proof. □

Remark 52. **(Error Decomposition)** Let $h_S = \text{ERM}_{\mathcal{H}}(S)$, we can decompose the risk as:

$$L_{\mathcal{D}}(h_S) = \mathcal{E}_{\text{app}} + \mathcal{E}_{\text{est}}$$

We have the following error:

- *Approximation Error*: $\mathcal{E}_{\text{app}} = \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$. How much risk do we need to restrict $\mathcal{H}$? This doesn't depend on S, while it decreases with the complexity of $\mathcal{H}$ increases.
- *Estimation Error*: $\mathcal{E}_{\text{est}} = L_{\mathcal{D}}(h_S) - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$. It is the result of L_S being estimator of $L_{\mathcal{D}}$, while it decreases with size S but increase with complexity of $\mathcal{H}$.

This is bias and complexity: choosing $\mathcal{H}' \supset \mathcal{H}$ leads to decreases in $\mathcal{E}_{\text{app}}$ while $\mathcal{E}_{\text{est}}$ increases.