# DDA4210 Spring 2024 - Assignment 1

Yohandi

January 31, 2024

**Problem 1.**

Answer each of the following problems with 1-2 short sentences.

(a) What is a hypothesis set?

(b) What is the hypothesis set of a linear model?

(c) What is overfitting?

(d) What are two ways to prevent overfitting?

(e) What are training data and test data, and how are they used differently? Why should you never change your model based on information from test data?

(f) What are the two assumptions we make about how our dataset is sampled?

(g) Consider the machine learning problem of deciding whether or not an email is spam. What could $X$, the input space, be? What could $Y$, the output space, be?

(h) What is the $k$-fold cross-validation procedure?

(a) A hypothesis set is a set of all possible hypotheses. A hypothesis is a model that can be learned by an algorithm to achieve predictions.

(b) The hypothesis set of a linear model that includes all linear function combinations that maps to predictions formed with given features.

(c) Overfitting is a machine learning behavior that captures noise in the training data too accurate (tailored) but very poor in the testing data (losing generalizability). Overfitting is also considered as modelling error.

(d) Reduce the number of features or model complexity. Add regularization terms to the loss function during training.

(e) Training data is a data that is used to fit a model, while test data is a data that evaluates the performance of the model. Changing such model according to the test data may lead to overfitting.

(f) Samples are independently and identically distributed (each is drawn from the same probability distribution independently of each other).

(g) $X$ could be text content, sender, title, or even frequency of certain words. $Y$ should be a binary value (1 or 0) indicating whether such an email is spam or not.

(h) $k$-fold cross-validation procedure is a procedure divides the dataset into $k$ subsets, where the model is trained on $k-1$ subsets and validated on the remaining one. Usually, the average of the results are used to estimate the model's performance.

**Problem 2.**

Derive the bias-variance decomposition for the squared error loss function. That is, show that for a model $f_S$ trained on a dataset $S$ to predict a target $y(x)$ for each $x$,

$$\mathbb{E}_S[E_{\text{out}}(f_S)] = \mathbb{E}_x[\text{Bias}(x) + \text{Var}(x)]$$

given the following definitions:

$$\begin{aligned} F(x) &= \mathbb{E}_S[f_S(x)] \\ E_{\text{out}}(f_S) &= \mathbb{E}_x[(f_S(x) - y(x))^2] \\ \text{Bias}(x) &= (F(x) - y(x))^2 \\ \text{Var}(x) &= \mathbb{E}_S[(f_S(x) - F(x))^2] \end{aligned}$$

$$\begin{aligned} \mathbb{E}_S[E_{\text{out}}(f_S)] &= \mathbb{E}_S[\mathbb{E}_x[(f_S(x) - y(x))^2]] \\ &= \mathbb{E}_{x,S}[(f_S(x) - F(x) + F(x) - y(x))^2] \\ &= \mathbb{E}_{x,S}[(F(x) - f_S(x))^2 + 2(f_S(x) - F(x))(F(x) - y(x)) + (F(x) - y(x))^2] \\ &= \mathbb{E}_{x,S}[(F(x) - f_S(x))^2] + \mathbb{E}_{x,S}[(F(x) - y(x))^2] + \mathbb{E}_{x,S}[2(f_S(x) - F(x))(F(x) - y(x))] \\ &= \mathbb{E}_x[\text{Var}(x)] + \mathbb{E}_{x,S}[\text{Bias}(x)] + 2\mathbb{E}_{x,S}[(f_S(x) - F(x))(F(x) - y(x))] \\ &= \mathbb{E}_x[\text{Var}(x)] + \mathbb{E}_x[\text{Bias}(x)] + 2\mathbb{E}_{x,S}[(f_S(x) - F(x))(F(x) - y(x))] \\ &= \mathbb{E}_x[\text{Var}(x) + \text{Bias}(x)] + 2\mathbb{E}_{x,S}[(f_S(x) - F(x))(F(x) - y(x))] \\ &= \mathbb{E}_x[\text{Var}(x) + \text{Bias}(x)] + 2\mathbb{E}_x[(F(x) - y(x))\mathbb{E}_S[f_S(x) - F(x)]] \\ &= \mathbb{E}_x[\text{Var}(x) + \text{Bias}(x)] + 2\mathbb{E}_x[(F(x) - y(x))(\mathbb{E}_S[f_S(x)] - \mathbb{E}_S[F(x)])] \\ &= \mathbb{E}_x[\text{Var}(x) + \text{Bias}(x)] + 2\mathbb{E}_x[(F(x) - y(x))(F(x) - F(x))] \\ &= \mathbb{E}_x[\text{Var}(x) + \text{Bias}(x)] + 2\mathbb{E}_x[0] \\ &= \mathbb{E}_x[\text{Var}(x) + \text{Bias}(x)] \end{aligned}$$

Hence, shown.

**Problem 3.**

Find the closed-form solutions of the following optimization problems ($\mathbf{W} \in \mathbb{R}^{K \times D}, N \gg D > K$):

(a) $\text{minimize}_{\mathbf{W},\mathbf{b}} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{W}\mathbf{x}_i - \mathbf{b}\|^2$

(b) $\text{minimize}_{\mathbf{W},\mathbf{b}} \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{W}\mathbf{x}_i - \mathbf{b}\|^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2$

Let

$$\hat{\mathbf{W}} = \begin{bmatrix} \mathbf{b}^T \\ \mathbf{w}_1^T \\ \vdots \\ \mathbf{w}_D^T \end{bmatrix} \in \mathbb{R}^{(D+1)\times K}, \quad \hat{\mathbf{X}} = \begin{bmatrix} 1 & \mathbf{x}_1^T \\ \vdots & \vdots \\ 1 & \mathbf{x}_N^T \end{bmatrix} \in \mathbb{R}^{N\times(D+1)}, \quad Y = \begin{bmatrix} \mathbf{y}_1^T \\ \vdots \\ \mathbf{y}_N^T \end{bmatrix} \in \mathbb{R}^{N\times K}, \quad \Lambda = \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda \end{bmatrix}$$

(a) Let $J(\hat{\mathbf{W}}) = \sum_{i=1}^{N} \|y_i - Wx_i - b\|^2$, then:

$$
\begin{aligned}
J(\hat{\mathbf{W}}) &= \sum_{i=1}^{N} \|y_i - Wx_i - b\|^2 \\
&= \text{trace}[(Y - \hat{\mathbf{X}}\hat{\mathbf{W}})^T(Y - \hat{\mathbf{X}}\hat{\mathbf{W}})] \\
&= \text{trace}[Y^TY - \hat{\mathbf{W}}^T\hat{\mathbf{X}}^TY - Y^T\hat{\mathbf{X}}\hat{\mathbf{W}} + \hat{\mathbf{W}}^T\hat{\mathbf{X}}^T\hat{\mathbf{X}}\hat{\mathbf{W}}]
\end{aligned}
$$

$$
\frac{\partial J(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}} = -\hat{\mathbf{X}}^TY - \hat{\mathbf{X}}^TY + (\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \hat{\mathbf{X}}^T\hat{\mathbf{X}})\hat{\mathbf{W}}
$$

Setting $\frac{\partial J(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}} = 0$ allows us to obtain $\hat{\mathbf{W}}^* = (\hat{\mathbf{X}}^T\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}^TY$.

This solution is inspired and adapted from my past solution in the DDA3020 course assignment

(b) Let $J(\hat{\mathbf{W}}) = \frac{1}{2}\sum_{i=1}^{N} \|y_i - Wx_i - b\|^2 + \frac{1}{2}\lambda\|\mathbf{W}\|_F^2$, then:

$$
\begin{aligned}
J(\hat{\mathbf{W}}) &= \frac{1}{2}\sum_{i=1}^{N} \|y_i - Wx_i - b\|^2 + \frac{1}{2}\lambda\|\mathbf{W}\|_F^2 \\
&= \frac{1}{2}\text{trace}[(Y - \hat{\mathbf{X}}\hat{\mathbf{W}})^T(Y - \hat{\mathbf{X}}\hat{\mathbf{W}})] + \frac{1}{2}\text{trace}[\hat{\mathbf{W}}^T\Lambda\hat{\mathbf{W}}] \\
&= \frac{1}{2}\text{trace}[Y^TY - \hat{\mathbf{W}}^T\hat{\mathbf{X}}^TY - Y^T\hat{\mathbf{X}}\hat{\mathbf{W}} + \hat{\mathbf{W}}^T\hat{\mathbf{X}}^T\hat{\mathbf{X}}\hat{\mathbf{W}}] + \frac{1}{2}\text{trace}[\hat{\mathbf{W}}^T\Lambda\hat{\mathbf{W}}]
\end{aligned}
$$

$$
\frac{\partial J(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}} = -\hat{\mathbf{X}}^TY + \hat{\mathbf{X}}^T\hat{\mathbf{X}}\hat{\mathbf{W}} + \Lambda\hat{\mathbf{W}}
$$

Setting $\frac{\partial J(\hat{\mathbf{W}})}{\partial \hat{\mathbf{W}}} = 0$ allows us to obtain $\hat{\mathbf{W}}^* = (\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \Lambda)^{-1}\hat{\mathbf{X}}^TY$.

This solution is inspired and adapted from my past solution in the DDA3020 course assignment

**Problem 4.**

Consider the following problem

$$
\underset{\mathbf{W}}{\text{minimize}} \frac{1}{2}\|\mathbf{W}\Phi - \mathbf{Y}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2
$$

where $\|\cdot\|_F$ denotes the Frobenius norm; $\mathbf{Y} \in \mathbb{R}^{K\times N}, \Phi = \left[\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)\right], \mathbf{x}_i \in \mathbb{R}^D, i = 1, 2, \ldots, N$, and $\phi$ is the feature map induced by kernel function $k(\cdot,\cdot)$. Prove that for any $\mathbf{x} \in \mathbb{R}^D$, we can make prediction as

$$
\mathbf{y} = \mathbf{W}\phi(\mathbf{x}) = \mathbf{Y}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x}),
$$

where $\mathbf{K} = \Phi^\top\Phi$ and $\mathbf{k}(\mathbf{x}) = \left[k(\mathbf{x}_1, \mathbf{x}), k(\mathbf{x}_2, \mathbf{x}), \ldots, k(\mathbf{x}_N, \mathbf{x})\right]^\top$

Let $J(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\Phi - \mathbf{Y}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2$ and $\Lambda = \lambda\mathbf{I} = \begin{bmatrix} \lambda & 0 & \dots & 0 \\ 0 & \lambda & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda \end{bmatrix}$, then:

$$\begin{aligned} J(\mathbf{W}) &= \frac{1}{2}\|\mathbf{W}\Phi - \mathbf{Y}\|_F^2 + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 \\ &= \frac{1}{2}\text{trace}((\mathbf{W}\Phi - \mathbf{Y})^T(\mathbf{W}\Phi - \mathbf{Y})) + \frac{1}{2}\text{trace}(\mathbf{W}^T\Lambda\mathbf{W}) \\ &= \frac{1}{2}\text{trace}(\Phi^T\mathbf{W}^T\mathbf{W}\Phi - \mathbf{Y}^T\mathbf{W}\Phi - \Phi^T\mathbf{W}^T\mathbf{Y} + \mathbf{Y}^T\mathbf{Y}) + \frac{1}{2}\text{trace}(\mathbf{W}^T\Lambda\mathbf{W}) \end{aligned}$$

$$\frac{\partial J(W)}{\partial W} = \mathbf{W}\Phi^T\Phi - \mathbf{Y}\Phi^T + \Lambda\mathbf{W}$$

Setting $\frac{\partial J(W)}{\partial W} = 0$ allows us to obtain $\mathbf{W}^* = \mathbf{Y}\Phi^T(\Phi\Phi^T + \Lambda)^{-1}$

Let $y^*$ be the prediction that we should make, then:

$$\begin{aligned} y^* &= \mathbf{W}^*\phi(\mathbf{x}) \\ &= \mathbf{Y}\Phi^T(\Phi^T\Phi + \Lambda)^{-1}\phi(\mathbf{x}) \\ &= \mathbf{Y}\Phi^T(\mathbf{K} + \Lambda)^{-1}\phi(\mathbf{x}) \end{aligned}$$

Observe that, by definition, $\phi(\mathbf{x}_i)^\top\phi(\mathbf{x}) = k(\mathbf{x}_i, \mathbf{x}), \forall i \in \{1, \dots, N\}$. Then, the expression of $\Phi^T\phi(\mathbf{x})$ has each of its element to be the inner product of $\phi(\mathbf{x}_i)$ with $\phi(\mathbf{x})$, which is $k(\mathbf{x}_i, \mathbf{x})$, $\forall i \in \{1, \dots, N\}$. Then, $\Phi^T\phi(\mathbf{x}) = [k(\mathbf{x}_1, \mathbf{x}), \dots, k(\mathbf{x}_N, \mathbf{x})] = k(\mathbf{x})$. We substitute $\Phi^\top\phi(\mathbf{x})$ accordingly and obtain $y^*$ as the following:

$$y^* = \mathbf{Y}(\mathbf{K} + \lambda\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$$

Hence, proved.

**Problem 5.**

Compute the space and time complexities (in the form of big $O$, consider only the training stage) of the following algorithms:

(a) Ridge regression (Question 3(b)) with the closed-form solution

(b) PCA ($N$ data points of $D$-dimension, choose $d$ principal components)

(c) Neural network with architecture $D - H_1 - H_2 - K$ on a mini-batch of size $B$ (consider only the forward process and neglect the computational costs of activation functions)

(a) To compute $\hat{\mathbf{W}}^* = (\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \Lambda)^{-1}\hat{\mathbf{X}}^T Y$, the ridge regression with the closed-form solution algorithm works as follows:

i. Computing $\hat{\mathbf{X}}^T\hat{\mathbf{X}}$ and add each element of the result with each element of $\Lambda$, which requires $\mathcal{O}(D^2N)$ time complexity. Then, storing the result in a $(D+1) \times (D+1)$ matrix requiring $\mathcal{O}(D^2)$ space complexity.

ii. Inverting $\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \Lambda$, which requires $\mathcal{O}(D^3)$ time complexity.

iii. Computing $\hat{\mathbf{X}}^T Y$, which requires $\mathcal{O}(DKN)$ time complexity. Then, storing the result in a $(D+1) \times K$ matrix requiring $\mathcal{O}(DK)$ space complexity.

iv. Multiplying $(\hat{\mathbf{X}}^T\hat{\mathbf{X}} + \Lambda)^{-1}$ with $\hat{\mathbf{X}}^T Y$, which requires $\mathcal{O}(D^2K)$ time complexity. Then, storing the result in a $(D+1) \times K$ matrix requiring $\mathcal{O}(DK)$ space complexity.

In overall, the algorithm requires $\mathcal{O}(D^3 + D^2N + DKN)$ time complexity and $\mathcal{O}(D^2 + DK)$ space complexity.

(b) To get the $d$ principal components from the $N$ data points of $D$-dimension, the PCA works as follows:

i. Computing mean of the data ($\mu$) and subtracting each data with the mean ($D'_i = D_i - \mu, \forall i \in \{1, \ldots, D\}$), which requires $\mathcal{O}(DN)$. The subtracted data is then stored in a $D \times N$ matrix requiring $\mathcal{O}(DN)$ space complexity.

ii. Computing the Covariance Matrix ($\frac{1}{n}D'D'^T$), which requires $\mathcal{O}(D^2N)$ time complexity. Then, storing the result in a $D \times D$ matrix requiring $\mathcal{O}(D^2)$ space complexity.

iii. Performing SVD for $\frac{1}{n}D'D'^T$ to get eigenvalues and eigenvectors, which requires $\mathcal{O}(D^3)$ time complexity. Then, storing the result in a $D \times D$ matrix requiring $\mathcal{O}(D^2)$ space complexity.

iv. Sorting and taking the top $d$ eigenvectors, which requires $\mathcal{O}(D \log D + dN)$ time complexity. Then, storing the result in a $d \times D$ matrix requiring $\mathcal{O}(dD)$ space complexity.

v. Projecting the data, which requires $\mathcal{O}(dDN)$ time complexity. Then, storing the result in a $d \times N$ matrix requiring $\mathcal{O}(dN)$ space complexity.

In overall, the algorithm requires $\mathcal{O}(D^3 + D^2N)$ time complexity (note that $d \leq D$) and $\mathcal{O}(D^2 + DN)$ space complexity.

Procedure follows **www.yohandi.me/blog/principal-component-analysis/**

(c) For one mini-batch, the computation requirements for each layer in the neural network are as follows:

i. Input Layer $\rightarrow$ Hidden Layer 1 requires $\mathcal{O}(BDH_1)$ time complexity and $\mathcal{O}(DH_1)$ space complexity.

ii. Hidden Layer 1 $\rightarrow$ Hidden Layer 2 requires $\mathcal{O}(BH_1H_2)$ time complexity and $\mathcal{O}(H_1H_2)$ space complexity.

iii. Hidden Layer 2 $\rightarrow$ Output Layer requires $\mathcal{O}(BH_2K)$ time complexity and $\mathcal{O}(H_2K)$ space complexity.

Moreover, the intermediate results take additional $\mathcal{O}(BD + BH_1 + BH_2 + BK)$ space complexity. In overall, assuming the computational costs of activation functions are neglected, the algorithm requires $\mathcal{O}(BDH_1 + BH_1H_2 + BH_2K)$ time complexity and $\mathcal{O}(BD + BH_1 + BH_2 + BK + DH_1 + H_1H_2 + H_2K)$ space complexity

**Problem 6.**

Prove the convergence of the generic gradient boosting algorithm (AnyBoost). Specifically, suppose in the algorithm of AnyBoost (page 14 of Lecture 02), the gradient of the objective function $\mathcal{L}$ is $L$-Lipschitz continuous, i.e., there exists $L > 0$ such that

$$\|\nabla\mathcal{L}(H) - \nabla\mathcal{L}(H')\| \leq L\|H - H'\|$$

holds for any $H$ and $H'$. Suppose in the algorithm, $\alpha$ is computed as

$$\alpha_{t+1} = -\frac{\langle \nabla \mathcal{L}(H_t), h_{t+1} \rangle}{L\|h_{t+1}\|^2}$$

Then the ensemble model is updated as $H_{t+1} = H_t + \alpha_{t+1} h_{t+1}$. Prove that the algorithm either terminates at round T with $\langle \nabla \mathcal{L}(H_t), h_{t+1} \rangle$ or $\mathcal{L}(H_t)$ converges to a finite value, in which case

$$\lim_{t\to\infty} \langle \nabla \mathcal{L}(H_t), h_{t+1} \rangle = 0$$

Using the fact that:

If $\mathcal{L} : \mathcal{H} \to R$ and $\|\nabla \mathcal{L}(F) - \nabla \mathcal{L}(G)\| \leq L\|F - G\|$ holds for any $F$ and $G$ in $\mathcal{H}$, then $\mathcal{L}(F + wG) - \mathcal{L}(F) \leq w\langle \mathcal{L}(F), G \rangle + \frac{Lw^2}{2}\|G\|^2$ holds for any $w > 0$

, we can derive $\mathcal{L}(H_{t+1})$ as follows:

$$\begin{aligned}
\mathcal{L}(H_{t+1}) &= \mathcal{L}(H_t + \alpha_{t+1} h_{t+1}) \\
&\leq \mathcal{L}(H_t) + \alpha_{t+1}\langle \mathcal{L}(H_t), h_{t+1} \rangle + \alpha_{t+1}^2 \frac{L}{2}\|h_{t+1}\|^2 \\
&= \mathcal{L}(H_t) - \alpha_{t+1}^2 \frac{L}{2}\|h_{t+1}\|^2 \text{ (implying } \mathcal{L}(H_t) > \mathcal{L}(H_{t+1})) \\
&= \mathcal{L}(H_t) - \frac{\langle \mathcal{L}(H_t), h_{t+1} \rangle^2}{2L\|h_{t+1}\|^2}
\end{aligned}$$

The above derivation implies that:

- $\mathcal{L}(H_t)$ monotonically decreases as $t$ increases. As $\mathcal{L}(H_t) > 0$, $\mathcal{L}(H_t)$ converges to a finite value;
- equivalently, $\lim_{t\to\infty} \langle \mathcal{L}(H_t), h_{t+1} \rangle^2 \leq \lim_{t\to\infty} 2L\|h_{t+1}\|(\mathcal{L}(H_t) - \mathcal{L}(H_{t+1}))$ converges to 0.

Those implications propose that $\lim_{t\to\infty} \langle \mathcal{L}(H_t), h_{t+1} \rangle = 0$

**Problem 7.**

Stochastic gradient descent (SGD) is an important optimization tool in machine learning, used everywhere from logistic regression to training neural networks. In this problem, you will be asked to first implement SGD for linear regression using the squared loss function. Then, you will analyze how several parameters affect the learning process.

Linear regression learns a model of the form:

$$f(x_1, x_2, \cdots, x_d) = \left( \sum_{i=1}^{d} w_i x_i \right) + b$$

(a) We can make our algebra and coding simpler by writing $f(x_1, x_2, \cdots, x_d) = \mathbf{w}^\top \mathbf{x}$ for vectors $\mathbf{w}$ and $\mathbf{x}$. But at first glance, this formulation seems to be missing the bias term $b$ from the equation above. How should we define $\mathbf{x}$ and $\mathbf{w}$ such that the model includes the bias term?

Linear regression learns a model by minimizing the squared loss function $L$, which is the sum across all training data $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ of the squared difference between actual and predicted output

values:

$$L(f) = \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

(b) SGD uses the gradient of the loss function to make incremental adjustments to the weight vector $\mathbf{w}$. Derive the gradient of the squared loss function with respect to $\mathbf{w}$ for linear regression.

(a) We should define $\mathbf{w}$ as $\mathbf{x}$ as $[1, x_1, \ldots, x_d]^T$ and $[b, w_1, \ldots, w_d]^T$ so that the model includes the bias term.

(b) The derivation is as follows:

$$\begin{aligned} L(f) &= \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \\ &= \sum_{i=1}^{N} y_i^2 - 2y_i \mathbf{w}^T \mathbf{x}_i + (\mathbf{w}^T \mathbf{x}_i)^2 \end{aligned}$$

$$\nabla_w L(f) = \sum_{i=1}^{N} 2\mathbf{w}^T \mathbf{x}_i^2 - 2y_i \mathbf{x}_i$$

**Problem 8.**

True or False? If False, then explain shortly.

(a) The inequality $G(\mathcal{F}, n) \leq n^2$ holds for any model class $\mathcal{F}$.

(b) The VC dimension of an axis-aligned rectangle in a 2D space is 4.

(c) The VC dimension of a circle in a 2D space is 4.

(d) The VC dimension of 1-nearest neighbor classifier in $d$-dimensional space is $d + 1$.

(e) Let $d$ be the VC dimension of $\mathcal{F}$. Then the inequality $G(\mathcal{F}, n) \leq \left(\frac{en}{d}\right)^d$ always holds.

(a) False. Consider linear function model in $\mathbb{R}^d$ that implies $\text{VC}(\mathcal{F}) = d + 1$ (according to page 21 of Lecture 03). Then, $G(\mathcal{F}, n) = G(\mathcal{F}, d + 1) = 2^{d+1} > (d + 1)^2, \forall d > 3$.

(b) True.

(c) False. If at least 3 of the 4 points are collinear then it is simply impossible due to alternating labelling case. Else, if there is a point inside the other 3 points, then it is also impossible in a case where the 3 points are labeled inside the circle. Otherwise, we can always form a convex tetragon. Due to the convexity of the tetragon, if a circle includes both diagonal positive points that lie in a not shorter diagonal line, it must also include at least a negative point lying on the other diagonal. Hence, VC dimension of a circle in a 2D space must be lower than 4.

(d) False. Consider 4 points in a 2-dimensional space, say $A, B, C, D$, are positioned as follows:

- $A$ at $(0, 0)$ is labeled $+$.
- $B$ at $(0, 1)$ is labeled $-$.
- $C$ at $(1, 0)$ is labeled $+$.
- $D$ at $(1, 1)$ is labeled $-$.

Then, $\mathcal{F}$ shatters $4 > (2 + 1)$ points in a 2-dimensional space.

(e) False. It requires $n \geq d$.

**Problem 9.**

In LASSO, the model class is defined as $\mathcal{F} = \{\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle : \|\mathbf{w}\|_1 \leq \alpha\}$. Suppose $\mathbf{x} \in \mathbb{R}^d$, $y \in \{-1, +1\}$, the training data are $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $\max_{1 \leq i \leq n} \|\mathbf{x}_i\|_\infty \leq \beta$, where $\|\cdot\|_\infty$ denotes the largest absolute element of a vector.

(a) Find an upper bound of the empirical Rademacher complexity

$$\mathcal{R}_S(\mathcal{F}) = \mathbb{E}_\sigma \left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \sigma_i f(\mathbf{x}_i) \right]$$

, where $\sigma_i$ are the Rademacher variables.

(b) Suppose the loss function is the absolute loss. Use the inequality (highlighted in blue) on page 30 and Lemma 5 on page 35 (i.e., $\mathcal{R}(\ell \circ \mathcal{F}) \leq \eta \mathcal{R}(\mathcal{F})$ of Lecture 03 to derive a generalization error bound for LASSO.

(a) We will use the following Lemma to find an upper bound for the empirical Rademacher complexity $\mathcal{R}_S(\mathcal{F})$:

Let $A \subseteq \mathbb{R}^n$ be a finite set of points with $r = \max_{x \in A} \|x\|$ and denote $x = (x_1, x_2, \ldots, x_n)$. Then

$$\mathbb{E}_\sigma \left[ \max_{x \in A} \sum_{i=1}^n x_i \sigma_i \right] \leq r \sqrt{2 \log |A|},$$

where $|A|$ denotes the cardinality of set $A$ and $\sigma_i$ are the Rademacher variables.

$$
\begin{aligned}
\mathcal{R}_{\mathcal{S}}(\mathcal{F}) &= \mathbb{E}_{\sigma}\left[\sup_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i f(\mathbf{x}_i)\right] \\
&= \mathbb{E}_{\sigma}\left[\sup_{\|\mathbf{w}\|_1\le\alpha}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\langle\mathbf{w},\mathbf{x}_i\rangle\right] \\
&= \mathbb{E}_{\sigma}\left[\sup_{\|\mathbf{w}\|_1\le\alpha}\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}\sigma_i w_j x_{ij}\right] \\
&= \frac{1}{n}\mathbb{E}_{\sigma}\left[\sup_{\|\mathbf{w}\|_1\le\alpha}\sum_{j=1}^{d}w_j\left(\sum_{i=1}^{n}\sigma_i x_{ij}\right)\right] \\
&\le \frac{1}{n}\mathbb{E}_{\sigma}\left[\|\mathbf{w}\|_1\max_{j=1,\dots,d}\left|\sum_{i=1}^{n}\sigma_i x_{ij}\right|\right] \quad \text{(by Hölder's inequality)} \\
&\le \frac{\alpha}{n}\mathbb{E}_{\sigma}\left[\max_{j=1,\dots,d}\left|\sum_{i=1}^{n}\sigma_i x_{ij}\right|\right] \\
&\le \frac{\alpha\beta}{n}\mathbb{E}_{\sigma}\left[\max_{j=1,\dots,d}\left|\sum_{i=1}^{n}\sigma_i\right|\right] \\
&\le \frac{\alpha\beta}{n}\sqrt{2\log d},
\end{aligned}
$$

(b) Using the derivation in part (a), we have:

$$
\begin{aligned}
\mathcal{R}(\ell\circ\mathcal{F}) &\le \mathcal{R}(\mathcal{F}) \\
&= \frac{\alpha\beta}{n}\sqrt{2\log d}.
\end{aligned}
$$

By applying the generalization bound for linear models with the absolute loss, we get the following inequality with probability at least $1-\delta$:

$$
\begin{aligned}
\sup_{f\in\mathcal{F}}\left|\mathbb{E}[\ell(f(\mathbf{x}),y)] - \frac{1}{n}\sum_{i=1}^{n}\ell(f(\mathbf{x}_i),y_i)\right| &\le 2\mathcal{R}(\ell\circ\mathcal{F}) + 3B\sqrt{\frac{\log(2/\delta)}{2n}} \\
&\le 2\cdot\frac{\alpha\beta}{n}\sqrt{2\log d} + 3\beta\sqrt{\frac{\log(2/\delta)}{2n}}
\end{aligned}
$$