STAT 37710: Machine Learning	Autumn 2022

Homework 1 Solutions

Please do not distribute.

1. Bias-variance decomposition (*15 points*)

Prove the claim in class:

$$\begin{aligned}\mathbb{E}_D\mathbb{E}_{X,Y}[(Y-\widehat{h}_D(X))^2] = &\ \mathbb{E}_X[(\mathbb{E}_D[\widehat{h}_D(X)]-h^\star(X))^2] \\ &+\mathbb{E}_X\mathbb{E}_D[(\widehat{h}_D(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])^2] \\ &+\mathbb{E}_{X,Y}[(Y-h^\star(X))^2].\end{aligned}$$

Solution: *Split* $Y-\widehat{h}_D(X))^2$ *to* $Y-h^\star(X)+h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)]+\mathbb{E}_{D'}[\widehat{h}'_D(X)]-\widehat{h}_D(X)$*, we have*

$$\begin{aligned}\mathbb{E}_D\mathbb{E}_{X,Y}[(Y-\widehat{h}_D(X))^2] = &\ \mathbb{E}_{X,Y}[(\mathbb{E}_D[\widehat{h}_D(X)]-h^\star(X))^2] \\ &+\mathbb{E}_{X,Y}\mathbb{E}_D[(\widehat{h}_D(X)-\mathbb{E}_D[\widehat{h}'_D(X)])^2] \\ &+\mathbb{E}_{X,Y}\mathbb{E}_D[(Y-h^\star(X))^2] \\ &+2\mathbb{E}_{X,Y}\mathbb{E}_D[(Y-h^\star(X))(h^\star(X)-\mathbb{E}_D[\widehat{h}'_D(X)])] \\ &+2\mathbb{E}_{X,Y}\mathbb{E}_D[(Y-h^\star(X))(\widehat{h}_D(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])] \\ &+2\mathbb{E}_{X,Y}\mathbb{E}_D[(\widehat{h}_D(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])(h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])]\end{aligned}$$

The first three terms simplifies to the three terms in the desired equation. Now it suffices to prove the last three terms all equal zero.

$$\begin{aligned}&\mathbb{E}_{X,Y}\mathbb{E}_D\left[(Y-h^\star(X))(h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])\right] \\ &=\mathbb{E}_X\mathbb{E}_{Y|X}\left[\mathbb{E}_D[(Y-h^\star(X))(h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])]\mid X\right] \\ &=\mathbb{E}_X\mathbb{E}_{Y|X}\left[(Y-h^\star(X))(h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])\mid X\right]\end{aligned}$$

When X *is fixed,* $h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)]$ *is constant, so*

$$\begin{aligned}&\mathbb{E}_X\mathbb{E}_{Y|X}\left[(Y-h^\star(X))(h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)])\mid X\right] \\ &=\left(h^\star(X)-\mathbb{E}_{D'}[\widehat{h}'_D(X)]\right)\mathbb{E}_{Y|X}\left[Y-h^\star(X)\mid X\right]\end{aligned}$$

Since $h^\star(X)=\mathbb{E}_{Y|X}\left[Y\mid X\right]$*, the last term is zero. The other two terms can be shown to be zero in a similar manner.*

2. MLE (*15 points*) Let $\mathbf{x}$ be a sample from some distribution with parameter θ. Let $L(\theta|\mathbf{x})$ be the likelihood function for $\theta\in\Theta$. Let $g:\Theta\to\mathbb{R}$ be a function and $\tilde{L}(\xi|\mathbf{x}):=\sup_{\theta:g(\theta)=\xi}L(\theta|\mathbf{x})$ be the induced likelihood function. Suppose $\hat{\theta}$ is the MLE of θ, prove that $g(\hat{\theta})$ is the MLE for $g(\theta)$.

Solution: *Let $\hat{\xi}$ be the MLE for $g(\theta)$. Then*

$$\begin{aligned}\tilde{L}(\hat{\xi}|\mathbf{x}) &= \sup_{\xi} \sup_{\theta: g(\theta)=\xi} L(\theta|\mathbf{x}) \\ &= \sup_{\theta} L(\theta|\mathbf{x}) \\ &= L(\hat{\theta}|\mathbf{x}) \\ &= \sup_{g(\theta)=g(\hat{\theta})} L(\theta|\mathbf{x}) \\ &= \tilde{L}(g(\hat{\theta})|\mathbf{x})\end{aligned}$$

So $g(\hat{\theta})$ is the MLE for $g(\theta)$.

3. Maximum likelihood estimation (*30 points*)

Suppose we have n i.i.d. samples $X_1, \cdots, X_n$ from the following distributions. Find the MLE of the required parameters.

a.(*5 points*) $\mathrm{Unif}(0, \theta)$. Find MLE for θ.

b.(*5 points*) $N(\mu, \sigma^2)$ with both parameters unknown. Find MLE for σ^2.

c.(*10 points*) $N(0, \Sigma)$. Find the MLE for Σ.

d.(*10 points*) $N(0, \Sigma)$. Find the MLE for $\Theta = \Sigma^{-1}$. Please also provide conditions that such the MLE exists.

Solution: *(a)* $\max_{1 \le i \le n} X_i$.

(b) $\frac{1}{n}\sum_{i=1}^n (X_i - \bar{X})^2$ *where* $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$.

(c) $\frac{1}{n}\sum_{i=1}^n X_i X_i^\top$

(d) $(\frac{1}{n}\sum_{i=1}^n X_i X_i^\top)^{-1}$ *given* $\frac{1}{n}\sum_{i=1}^n X_i X_i^\top$ *is invertible.*

4. Convex optimization (*10 points*)

Suppose that $f : \mathbb{R}^d \mapsto \mathbb{R}$ is a convex and differentiable function. Show that x is a minimizer of f if and only if $\nabla f(x) = 0$.

Solution: *Suppose $\nabla f(x) = 0$, then for any $y \in \mathbb{R}^d$, $f(y) \ge f(x) + \nabla f(x)^\top (y - x) = f(x)$ because of convexity.*

Suppose x is a minimizer, take any unit vector v. From the definition of gradient we know

$$\lim_{t \to 0} \frac{f(x+tv) - f(x)}{t} = \nabla(x)^\top v$$

Since the limit exists and $\frac{f(x+tv)-f(x)}{t} > 0$ for $t > 0$, $\frac{f(x+tv)-f(x)}{t} < 0$ for $t < 0$, the limit has to be zero. $\nabla(x)^\top v = 0$ for all unit vector v implies $\nabla f(x) = 0$.

5. Programming assignment: Empirical risk minimization (*30 points*) Given a sample data set

D and loss function $\ell(y, \hat{y})$, the *empirical risk* (or empirical loss) of a hypothesis h is defined as the sample mean of the loss on $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$:

$$\hat{R}_D(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i))$$

where $h(x_i)$ is the predicted label for x_i. In *empirical risk minimization*, we use the empirical risk $\hat{R}_D(h)$ as an estimator of the minimal true risk, a.k.a. the *Bayes risk*, defined as

$$R^* := \min_h \mathbb{E}_{X,Y} \ell(Y, h(X)).$$

Given hypothesis class $\mathcal{H}$ (e.g. a collection of predictors), denote the empirical risk estimator as

$$\hat{h}_D := \mathrm{argmin}_{h \in \mathcal{H}} \hat{R}_D(h).$$

The expected error of the empirical risk estimator $\hat{h}_D$ is

$$\mathbb{E}_D \hat{R}_D(\hat{h}_D) - R^* = \underbrace{\mathbb{E}_D \hat{R}_D(\hat{h}_D) - \min_{h \in \mathcal{H}} R(h)}_{\text{estimation error}} + \underbrace{\min_{h \in \mathcal{H}} R(h) - R^*}_{\text{approximation error}}$$

Hereby, the *estimation error* is due to error caused by n training samples instead of full knowledge of the joint distribution of X, Y, and the *approximation error* is due to restricting our attention to model class $\mathcal{H}$.

In this question, we will investigate the trade-off between *estimation error* and *approximation error* given different collections of predictors.

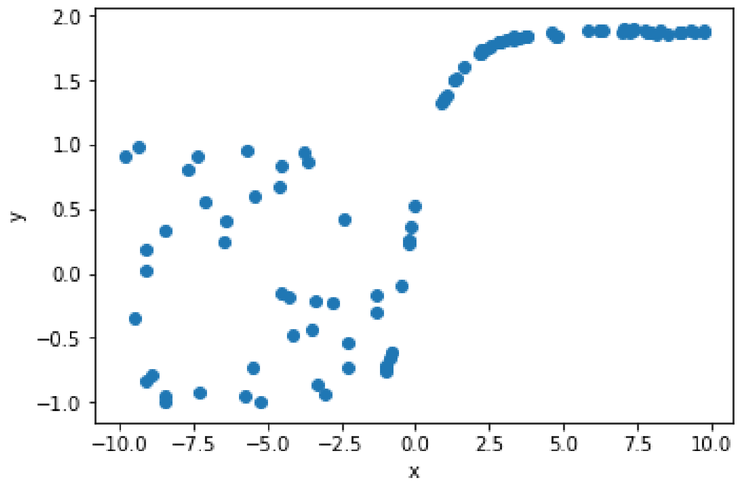
Use the following code to generate a dataset:

```python
#python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

def data_generator(n_samples):
    x = np.random.uniform(-10, 10, n_samples)
    y = np.cos(0.5 + np.exp(-x)) + 1/(1 + np.exp(-x))
    noise = np.random.normal(0, 0.01, n_samples)
    y += noise
    return x, y

complete_X, complete_Y = data_generator(5000)
train_X, train_Y = complete_X[:100], complete_Y[:100]
large_X, large_Y = complete_X[100:], complete_Y[100:]

loss_func = mean_squared_error
```

Use `train_X` and `train_Y` as training samples for ERM. `large_X` and `large_Y` are for the approximation of true data distribution of $\mathbf{X}$ and Y, in order to estimate true risk. A plot of a small portion of the dataset.

Use mean squared error as loss function in this problem (where $\ell(y_i, h(x_i)) = (y_i - h(x_i))^2$), unless noted otherwise.

a.(*10 points*) Let's first define $\mathcal{H}_k$ to be a collection of all possible polynomial functions of degree k. Implement the ERM process to select the predictor $\hat{h} \in \mathcal{H}_k$ with the lowest empirical risk. (Hint: `polyfit` function in numpy could be useful.)



b.(*10 points*) Experiment with the ERM you built with k of $\mathcal{H}_k$ ranging from 0 to 30. Report empirical loss and plot a graph of empirical risk v.s. k.

c.(*10 points*) We will further explore the approximation vs. estimation trade-off. First, use the noise-free distribution in data_generator to estimate R^* with the **complete dataset**. i.e., the complete dataset has noisy (x, y) pairs and we know that without noise

$$y^* := \mathbb{E}[Y \mid \mathbf{X} = x] = \cos(0.5 + e^{-x}) + \frac{1}{1 + e^{-x}}. \tag{1}$$

(Note that in real applications, you normally do not have access to the true distribution of $\mathbf{X}$ and Y.) Now use `large_X`, `large_Y` to estimate the Bayes risk R^*, the risk of the ERM for each k, the estimation error, and the approximation error. Plot graphs of these errors v.s. k. Experiment with k range from 0 to 25. (Note: these different errors might not be in the same scale. You can plot one graph for each error v.s. k)

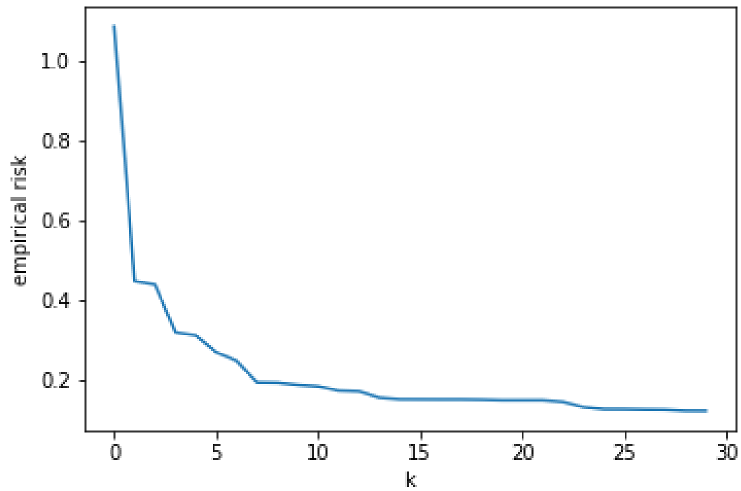
(a)

```
#python
def evaluate(clf, test_x, test_y, loss_func):
    y_hat = clf(test_x)
    loss = loss_func(test_y, y_hat)
    return loss

def erm(k, train_x, train_y, loss_func):
    clf = np.polyfit(train_x, train_y, k)
    clf = np.poly1d(clf)
    empirical_loss =
        evaluate(clf, train_x, train_y, loss_func)
    return clf, empirical_loss
```

(b)

```
#python
empirical_loss_list = []
loss_func = mean_squared_error
for k in range(30):
```

```
    _, emp_loss = erm(k, train_X, train_Y, loss_func)
    empirical_loss_list += emp_loss,

plt.clf()
x = range(30)
plt.plot(x, empirical_loss_list)
plt.show()
```



(c)

```
#python
def compute_r_star(x, y, loss_func):
    y_hat = np.cos(0.5 + np.exp(-x)) + 1/(1 + np.exp(-x))
    risk = loss_func(y, y_hat)
    return risk

r_star = compute_r_star(complete_X, complete_Y, loss_func)

emp_risk_list = []
estimation_list = []
approximation_list = []
real_risk_list = []
erm_risk_list = []
max_k = 25

for k in range(max_k):
    clf, empirical_risk = erm(k, train_X, train_Y, loss_func)
    emp_risk_list += empirical_risk,

    erm_risk = evaluate(clf, large_X, large_Y, loss_func)
    erm_risk_list += erm_risk,

    clf, real_risk = erm(k, large_X, large_Y, loss_func)
```

```
    estimation_error = erm_risk - real_risk
    approximation_error = real_risk - r_star

    estimation_list += estimation_error,
    approximation_list += approximation_error,
    real_risk_list += real_risk,
```