download (1).png1 Theory Exercises

Exercise 1 Naive Bayes theory

1. a. Logistic Regression is a discriminative model, and thus we estimate $P(y|x;\theta)$. For Naive Bayes, the generative model, we either can estimate $P(x,y;\theta)$ or $P(x|y;\theta)$ and $P(y;\theta)$ which essentially means we generate a sample.

1. b. For Naive Bayes, once we finish learning, we have found $P(y^{(i)};\theta)$ and $P(x_j^{(i)}|y^{(i)};\theta)$ for $j = 1, \ldots n$. We can then compute $P(y|x;\theta)$ by using Bayes' Theorem.

1. c. The fundamental assumption of naive Bayes that makes it attractive for high-dimensional problems is that all our features are conditionally independent given labels y.

1. d. For each n in $x^{(i)}$, we have 2 possiblities, 0 or 1. There are thus $2n$ parameters for that. We also need to consider the info about the labels, which is our prior distribution information about the labels. This is 1 thing, because it is 1. Thus, $2n + 1$ total model parameters for each feature.

2. a. $P(x_{zebra} = 1|y = \text{spam})$ would be estimated as 0 and our $P(x_{zebra} = 1|y = \text{not spam})$ would also be 0. This is because we haven't seen it at all yet, so the number of times we have seen it in either case $P(x|y) = 0$

2. b. No matter what, our probability is 0 because we multiply by 0.

Exercise 2 Linear Regression: Ordinary Least Squares

1. The cost function we are trying to minimize in OLS is $J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$

2. The design matrix of OLS is $\mathbb{X} = \begin{bmatrix} - & \hat{x}^{(1)T} & - \\ & \vdots & \\ - & \hat{x}^{(m)T} & - \end{bmatrix}$

where each $\hat{x}^{(i)}$ vector contains $n+1$ terms like $\begin{bmatrix} 1 & x_{1i} & x_{2i} & \dots & x_{ni} \end{bmatrix}$

3. $J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$= \frac{1}{2} \sum_{i=1}^{m} (\theta^T x^{(i)} - y^{(i)})^2$

$J(\theta) = \frac{1}{2} ||\mathbb{X}\theta - \hat{y}||_2^2$

4. $J(\theta) = \frac{1}{2} ||\mathbb{X}\theta - \hat{y}||_2^2 = \frac{1}{2} (\mathbb{X}\theta - \hat{y})^T I (\mathbb{X}\theta - \hat{y})$

5. Gradient of cost function with respect to θ

$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2} (\mathbb{X}\theta - \hat{y})^T (\mathbb{X}\theta - \hat{y})$

$\nabla_\theta J(\theta) = \frac{1}{2} \nabla_\theta (\theta^T \mathbb{X}^T - \hat{y}^T)(\mathbb{X}\theta - \hat{y})$

$\nabla_\theta J(\theta) = \frac{1}{2} \nabla_\theta (\theta^T \mathbb{X}^T \mathbb{X}\theta - \hat{y}^T \mathbb{X}\theta - \theta^T \mathbb{X}^T \hat{y} + \hat{y}^T \hat{y})$

$\nabla_\theta J(\theta) = \frac{1}{2} (2\mathbb{X}^T \mathbb{X}\theta - \hat{y}^T \mathbb{X} - \mathbb{X}^T \hat{y} + 0)$

$\nabla_\theta J(\theta) = \frac{1}{2} (2\mathbb{X}^T \mathbb{X}\theta - 2\mathbb{X}^T \hat{y})$

$\nabla_\theta J(\theta) = \mathbb{X}^T \mathbb{X}\theta - \mathbb{X}^T \hat{y}$

6.

$0 = \mathbb{X}^T \mathbb{X}\theta - \mathbb{X}^T \hat{y}$

$\mathbb{X}^T \mathbb{X}\theta = \mathbb{X}^T \hat{y}$

$\theta = (\mathbb{X}^T \mathbb{X})^{-1} \mathbb{X}^T \hat{y}$

This is sufficient for OLS because it is convex

$\theta^* = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\hat{y}$

$\mathbb{X}\theta^* = \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\hat{y}$

$\mathbb{X}\theta^* = U(\Sigma V^T\theta)$

Assuming $m >> n$ and $\mathbb{X}$ is full rank then

$= U\Sigma V^T((U\Sigma V^T)^T(U\Sigma V^T))^{-1}(U\Sigma V^T)^T\hat{y}$

$= U\Sigma V^T((V\Sigma^T U^T)(U\Sigma V^T))^{-1}V\Sigma^T U^T\hat{y}$*

$U^TU = I$ and $V^T = V^{-1}$

$= U\Sigma V^T((V\Sigma^T\Sigma V^T))^{-1}V\Sigma^T U^T\hat{y}$

$= U\Sigma V^T V(\Sigma)^T(\Sigma)V^T V\Sigma^T U^T\hat{y}$

$= U\Sigma(\Sigma)^T(\Sigma)\Sigma^T U^T\hat{y}$

$\mathbb{X}\theta^* = UU^T\hat{y}$

2 Coding Exercises

Exercise 3: Soft Margin Linear SVM

1. Datapoints for which cost is 0 are on or outside the margin of the correct side.

2. If cost > 1 then the point is incorrectly classified

3. a. margin = 1/np.linalg.norm(w) or margin $= \dfrac{1}{||w||}$

3. b. $x : w^T + b = 0$ which is basically being plotted in the file like this:

```
w = clf.coef_[0]
b = clf.intercept_[0]
xx2 = (-w[0]/w[1])*xx1 - b/w[1]
```

3. c. The datapoints that are our support vectors are points on the margin, and error points.

4. a. The margin is 1.1221702071161122 if C=1 according to the program

4. b. If C = 1, there are support vectors [16 17] for each class, so 33 total support vectors.

4. C. all plots included with file

```
C=10^-3
Number of support vectors from each class: [49 49]
margin: 6.933741692417118
w: [0.09620674 0.10744453]
b: -0.8368885305253787
```

```
C = 1
Number of support vectors from each class: [16 17]
Margin: 1.1221702071161122
w: [0.48022148 0.75066686]
b: -1.6445252371736272
```

```
C = 10^3
Number of support vectors from each class: [16 17]
margin: 0.9603131228687151
w: [0.55976543 0.87808008]
b: -2.0813791958599066
```

4. d. As C increases, the margin appears to decrease.

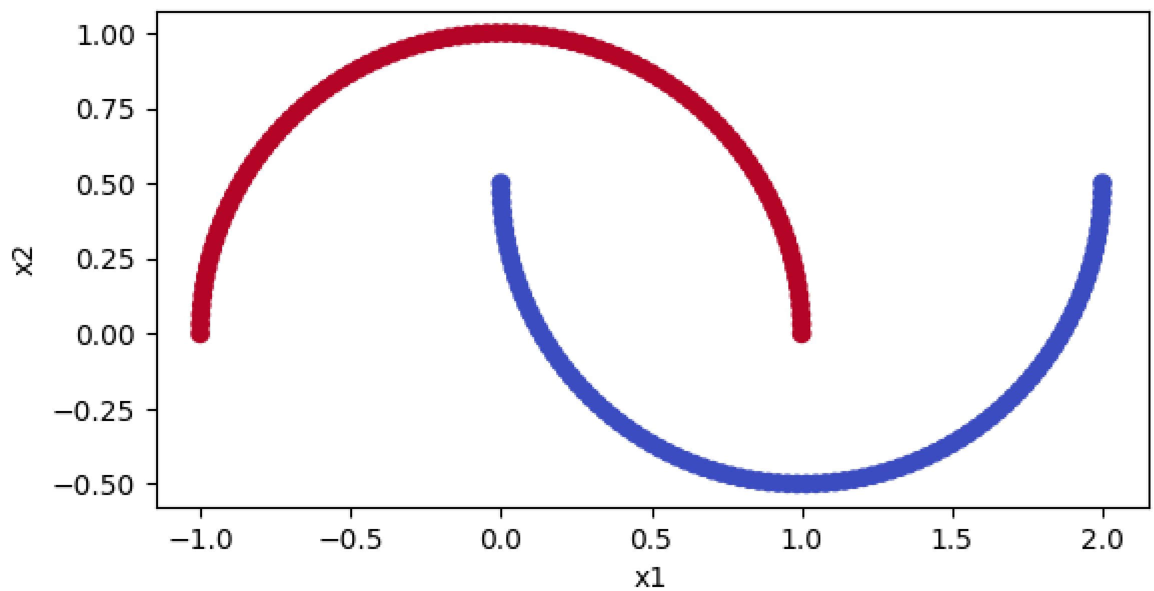
5. a. We should use something somewhere between $C = 10^{-2}$ and $C = 10^4$ these all had reasonably close performance that was the lowest errors. Plot is included in the file.

5. b. No, there is no way to linearly separate the data, so training error can't ever be zero.

Exercise 4. RBF Kernel SVMs

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.model_selection import validation_curve, GridSearchCV
from sklearn import datasets
```

```
# load and plot the famous moons dataset from sklearn
xdat, ydat = datasets.make_moons(n_samples=200, noise=0, random_state=8)
fig0, ax0 = plt.subplots()
ax0.scatter(xdat[:,0], xdat[:,1], c=1-ydat, cmap='coolwarm') # set color to c=1-ydat to reverse colors
ax0.set(xlabel='x1', ylabel='x2')
ax0.set_aspect('equal')
```



1. When the moons dataset has 0 noise, it looks like 2 arcs

2. There are the following 2 hyperparameters for an RBF kernel SVM:C and gamma.

The larger C is, the closer everything is to the decision boundary. Larger gammas made the decision boundary super wiggly and everything was much closer.

3. The optimal value of gamma was 10 as it had the lowest training, and validation error that wasn't going up yet.

4. There can be a gamma where the training error is 0 because we can fit every single datapoint by just wiggling around all of them.

+ Code + Text

5. The best parameters were the ones initially in the program.

```
The best parameters are {'C': 10000.0, 'gamma': 0.1} with a validation accuracy of 0.94
```

Plot has been included with the file.