10-701 Machine Learning: Assignment 1

Due on Februrary 20, 2014 at 12 noon

Barnabas Poczos, Aarti Singh

Instructions: Failure to follow these directions may result in loss of points.

- Your solutions for this assignment need to be in a pdf format and should be submitted to the blackboard and a webpage (to be specified later) for peer-reviewing.
- For the programming question, your code should be well-documented, so a TA can understand what is happening.
- DO NOT include any identification (your name, andrew id, or email) in both the content and filename of your submission.

MLE, MAP, Concentration (Pengtao)

1. MLE of Uniform Distributions [5 pts]

Given a set of i.i.d samples $X_1, ..., X_n \quad Uniform(0, \theta)$, find the maximum likelihood estimator of θ.

(a) Write down the likelihood function (3 pts)

(b) Find the maximum likelihood estimator (2 pts)

(a) Write down the likelihood function. (3 pts)
Let $X_{max} = \max\{X_1, \ldots, X_n\}$, and let I_A denote the indicator function of event A. The likelihood function $\mathcal{L}$ can be written as

$$\mathcal{L} = P(X_1, \ldots, X_n|\theta) = \prod_{i=1}^{n} p(X_i|\theta) = \prod_{i=1}^{n} \frac{1}{\theta} I_{\{X_i \le \theta\}} = \begin{cases} (\frac{1}{\theta})^n & \text{if } \forall X_i \le \theta \\ 0 & \text{Otherwise} \end{cases}$$

Simply just writing that the likelihood function is $(\frac{1}{\theta})^n$ is not enough!
(b) Find the maximum likelihood estimator. (2 pts)

$$\begin{aligned} \arg\max_\theta \mathcal{L} &= \arg\max_\theta P(X_1, \ldots, X_n|\theta) = \arg\max_\theta \left(\frac{1}{\theta}\right)^n \quad \text{s.t. } \forall X_i \le \theta \\ &= \max\{X_1, \ldots, X_n\} = X_{max} \end{aligned}$$

The optimal value of θ that maximizes the likelihood function is X_{max}.

2. Concentration [5 pts]

The instructors would like to know what percentage of the students like the Introduction to Machine Learning course. Let this unknown—but hopefully very close to 1—quantity be denoted by μ. To estimate μ, the instructors created an anonymous survey which contains this question:

"Do you like the Intro to ML course? Yes or No"

Each student can only answer this question once, and we assume that the distribution of the answers is i.i.d.

(a) What is the MLE estimation of μ? (1 pts)

(b) Let the above estimator be denoted by $\hat{\mu}$. How many students should the instructors ask if they want the estimated value $\hat{\mu}$ to be so close to the unknown μ such that

$$\mathbb{P}(|\hat{\mu} - \mu| > 0.1) < 0.05, \qquad \text{(4pts)}$$

(a) (1 pts) This problem is equivalent to estimating the mean parameter of a Bernoulli distribution from i.i.d. data. Therefore, the MLE estimation is $\hat{\mu} = \frac{n_1}{N}$, where n_1 is the number of students who answered Yes and N is the total number of students.
(b) (4 pts) Let $X_i = 1$ if a student answered yes, and let $X_i = 0$ if the answer was no. [If you did the calculations with +1,-1 pairs, that is fine too.]
According to Hoeffding's equality,

$$\Pr(|\hat{\mu} - \mu| > \epsilon) \le 2\exp\left(-\frac{2N^2\epsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right)$$

$[a_i, b_i]$ is the range of the random variable X_i, therefore in our case $a_i = 0$, $b_i = 1$

$$P(|\hat{\mu} - \mu| > 0.1) < 2e^{-2N\times(0.1)^2} = 0.05 \qquad (1)$$

from which we have, $N = 50\ln 40$
So the instructor needs 185 students.

3. MAP of Multinational Distribution [10 pts]

You have just got a loaded 6-sided dice from your statistician friend. Unfortunately, he does not remember its exact probability distribution $p_1, p_2, ..., p_6$. He remembers, however, that he generated the vector $(p_1, p_2, \ldots, p_6)$ from the following Dirichlet distribution.

$$\mathbb{P}(p_1, p_2, \ldots, p_6) = \frac{\Gamma(\sum_{i=1}^{6} u_i)}{\prod_{i=1}^{6}\Gamma(u_i)} \prod_{i=1}^{6} p_i^{u_i - 1} \delta(\sum_{i=1}^{6} p_i - 1),$$

where he chose $u_i = i$ for all $i = 1, \ldots, 6$. Here Γ denotes the gamma function, and δ is the delta function, i.e. $\delta(a) = 1$ if $a = 0$, and $\delta(a) = 0$ otherwise. To estimate the probabilities $p_1, p_2, \ldots, p_6$, you roll the dice 1000 times and then observe that side i occurred n_i times ($\sum_{i=1}^{6} n_i = 1000$).

(a) Prove that the Dirichlet distribution is conjugate prior for the multinomial distribution.

(b) What is the posterior distribution of the side probabilities, $\mathbb{P}(p_1, p_2, \ldots, p_6 | n_1, n_2, \ldots, n_6)$?

(a) (8 pts)
We solve the problem with the above notation for the 6-sided dice. If you solve it for the more general case, that is also fine.
Our data is $n_1, n_2, \ldots, n_6$, and the distribution we study is multinomial distribution. Therefore, the likelihood function is

$$\mathbb{P}(n_1, n_2, \ldots, n_6 | p_1, p_2, \ldots, p_6) = \prod_{i=1}^{6} p_i^{n_i}$$

To see that the Dirichlet distribution is conjugate prior to the multinomial distribution, we need to calculate the product of the likelihood and the Dirichlet distribution.

$$\begin{aligned}\mathbb{P}(n_1, n_2, \ldots, n_6, p_1, p_2, \ldots, p_6) &= \left(\prod_{i=1}^{6} p_i^{n_i}\right) \frac{\Gamma(\sum_{i=1}^{6} u_i)}{\prod_{i=1}^{6} \Gamma(u_i)} \prod_{i=1}^{6} p_i^{u_i - 1} \delta(\sum_{i=1}^{6} p_i - 1) \\ &= \frac{\Gamma(\sum_{i=1}^{6} u_i)}{\prod_{i=1}^{6} \Gamma(u_i)} \prod_{i=1}^{6} p_i^{n_i + u_i - 1} \delta(\sum_{i=1}^{6} p_i - 1)\end{aligned}$$

Therefore,

$$\begin{aligned}\mathbb{P}(p_1, p_2, \ldots, p_6 | n_1, n_2, \ldots, n_6) &= \frac{\mathbb{P}(p_1, p_2, \ldots, p_6, n_1, n_2, \ldots, n_6)}{\mathbb{P}(n_1, n_2, \ldots, n_6)} \\ &= \frac{1}{\mathbb{P}(n_1, n_2, \ldots, n_6)} \frac{\Gamma(\sum_{i=1}^{6} u_i)}{\prod_{i=1}^{6} \Gamma(u_i)} \prod_{i=1}^{6} p_i^{n_i + u_i - 1} \delta(\sum_{i=1}^{6} p_i - 1) \\ &= \mathit{const} \times \prod_{i=1}^{6} p_i^{n_i + u_i - 1} \delta(\sum_{i=1}^{6} p_i - 1)\end{aligned}$$

where *const* doesn't depend on $p_1, p_2, \ldots, p_6$.
Now we can see that the posterior and the prior are both Dirichlet distributions, thereby, Dirichlet prior is a conjugate prior for the multinomial distribution.

(b) (2 pts)
From the above equation, we have that the posterior is

$$P(\{p_i\}_{i=1}^{6} | \{n_i\}_{i=1}^{6}) = \frac{\Gamma(\sum\limits_{i=1}^{6} n_i + \mu_i)}{\prod\limits_{i=1}^{6} \Gamma(n_i + \mu_i)} \prod_{i=1}^{6} p_i^{n_i + \mu_i - 1} \delta(\sum_{i=1}^{6} p_i - 1) \tag{2}$$

Linear Regression (Dani)

1. Optimal MSE rule [10 pts]

Suppose we knew the joint distribution P_{XY}. The optimal rule $f^* : X \to Y$ which minimizes the MSE (Mean Square Error) is given as:

$$f^* = \arg\min_f \mathbb{E}[(f(X) - Y)^2]$$

Show that $f^*(X) = \mathbb{E}[Y|X]$.

(10 points)
Notice that it suffices to argue that

$$\mathbb{E}[(f(X) - Y)^2] \geq \mathbb{E}[(\mathbb{E}[Y|X] - Y)^2] \text{ for all } f$$

and hence $f^*(X) = \mathbb{E}[Y|X]$.

$$\begin{aligned}
\mathbb{E}[(f(X) - Y)^2] &= \mathbb{E}[(f(X) - \mathbb{E}[Y|X] + \mathbb{E}[Y|X] - Y)^2] \\
&= \mathbb{E}[(f(X) - \mathbb{E}[Y|X])^2 + (\mathbb{E}[Y|X] - Y)^2 + 2(f(X) - \mathbb{E}[Y|X])(\mathbb{E}(Y|X) - Y)] \\
&= \mathbb{E}[(f(X) - \mathbb{E}[Y|X])^2] + \mathbb{E}[(\mathbb{E}[Y|X] - Y)^2] + 2\mathbb{E}[(f(X) - \mathbb{E}[Y|X])(\mathbb{E}[Y|X] - Y)]
\end{aligned}$$

Now using the fact that $\mathbb{E}_{XY}[\dots] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\dots|X]]$, we have

$$\begin{aligned}
\mathbb{E}_{XY}[(f(X) - \mathbb{E}[Y|X])(\mathbb{E}[Y|X] - Y)] &= \mathbb{E}_X[\mathbb{E}_{Y|X}[(f(X) - \mathbb{E}[Y|X])(\mathbb{E}[Y|X] - Y)|X]] \\
&= \mathbb{E}_X[(f(X) - \mathbb{E}[Y|X])\mathbb{E}_{Y|X}[(\mathbb{E}[Y|X] - Y)|X]] = 0
\end{aligned}$$

where the second last step follows since conditioning on X, $f(X)$ and $\mathbb{E}[Y|X]$ are constant. Therefore,

$$\begin{aligned}
\mathbb{E}[(f(X) - Y)^2] &= \mathbb{E}[(f(X) - \mathbb{E}[Y|X])^2] + \mathbb{E}[(\mathbb{E}[Y|X] - Y)^2] \\
&\geq \mathbb{E}[(\mathbb{E}[Y|X] - Y)^2]
\end{aligned}$$

since the first term being square of a quantity is non-negative.
Note: There are many ways to prove this. Please use your judgement and give partial credit if some arguments are right and others are not.

2. Ridge Regression [10 pts]

In class, we discussed ℓ_2 penalized linear regression:

$$\widehat{\beta} = \arg\min_\beta \sum_{i=1}^n (Y_i - X_i\beta)^2 + \lambda\|\beta\|_2^2$$

where $X_i = [X_i^{(1)} \dots X_i^{(p)}]$.

a) Show that a closed form expression for the ridge estimator is $\widehat{\beta} = (\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I})^{-1}\boldsymbol{A}^\top \boldsymbol{Y}$ where $\boldsymbol{A} = [X_1; \dots; X_n]$ and $\boldsymbol{Y} = [Y_1; ...; Y_n]$.

b) An advantage of ridge regression is that a unique solution always exists since $(\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I})$ is invertible. To be invertible, a matrix needs to be full rank. Argue that $(\boldsymbol{A}^\top \boldsymbol{A} + \lambda \boldsymbol{I})$ is full rank by characterizing its p eigenvalues in terms of the singular values of $\boldsymbol{A}$ and λ.

a) (5 points) As in class, we start by writing the objective function in matrix form:

$$\begin{aligned}\mathcal{L} &= \|\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2 \\ &= (\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{\beta})^\top(\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}\end{aligned}$$

We will now compute the derivative of the objective function with respect to $\boldsymbol{\beta}$ using the following two vector differentiation rules: (1) $\partial\boldsymbol{b}^\top\boldsymbol{x}/\partial\boldsymbol{x} = \boldsymbol{b}$ and (2) $\partial\boldsymbol{x}^\top\boldsymbol{B}\boldsymbol{x}/\partial\boldsymbol{x} = (\boldsymbol{B} + \boldsymbol{B}^\top)\boldsymbol{x}$.

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= -2\boldsymbol{A}^\top(\boldsymbol{Y} - \boldsymbol{A}\boldsymbol{\beta}) + 2\lambda\boldsymbol{\beta} \\ &= -2\boldsymbol{A}^\top\boldsymbol{Y} + 2\boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{\beta} + 2\lambda\boldsymbol{\beta}\end{aligned}$$

Since the derivative is zero at the minimizer $\hat{\boldsymbol{\beta}}$, we get

$$\begin{aligned}\boldsymbol{A}^\top\boldsymbol{Y} &= (\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I})\hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{\beta}} &= (\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I})^{-1}\boldsymbol{A}^\top\boldsymbol{Y}\end{aligned}$$

b) (5 points) $\boldsymbol{A}^\top\boldsymbol{A}$ is a real symmetric matrix with real eigenvalues. It is also a positive semidefinite matrix, so all eigenvalues are nonnegative. (Equivalently, you can say that the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ are the square of the singular values of $\boldsymbol{A}$, and hence are nonnegative.) Let us denote these real nonnegative eigenvalues by $\{\nu_i\}_{i=1}^p$ (Notice that some of these can be zero).
Now notice that any eigenvector $\boldsymbol{v}_i$ of $\boldsymbol{A}^\top\boldsymbol{A}$ corresponding to eigenvalue ν_i is also an eigenvector of $\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I}$ with eigenvalue $\nu_i + \lambda$ since

$$(\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I})\boldsymbol{v}_i = \boldsymbol{A}^\top\boldsymbol{A}\boldsymbol{v}_i + \lambda\boldsymbol{v}_i = (\nu_i + \lambda)\boldsymbol{v}_i.$$

Since $\lambda > 0$, all eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A} + \lambda\boldsymbol{I}$ are positive, so it is a full rank matrix and invertible.
Note: If you do not show that the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ are nonnegative, you only lose 2 points. If you do not argue why adding a positive constant λ times identity matrix to $\boldsymbol{A}^\top\boldsymbol{A}$ makes the eigenvalues equal to $\nu_i + \lambda$, you lose 1 point.

Logistic Regression (Prashant)

1. Overfitting and Regularized Logistic Regression [10 pts]

a) Plot the sigmoid function $1/(1+e^{-wX})$ vs. $X \in \mathbb{R}$ for increasing weight $w \in \{1, 5, 100\}$. A qualitative sketch is enough. Use these plots to argue why a solution with large weights can cause logistic regression to overfit.

b) To prevent overfitting, we want the weights to be small. To achieve this, instead of maximum conditional likelihood estimation M(C)LE for logistic regression:

$$\max_{w_0,\ldots,w_d} \prod_{i=1}^{n} P(Y_i|X_i, w_0, \ldots, w_d),$$

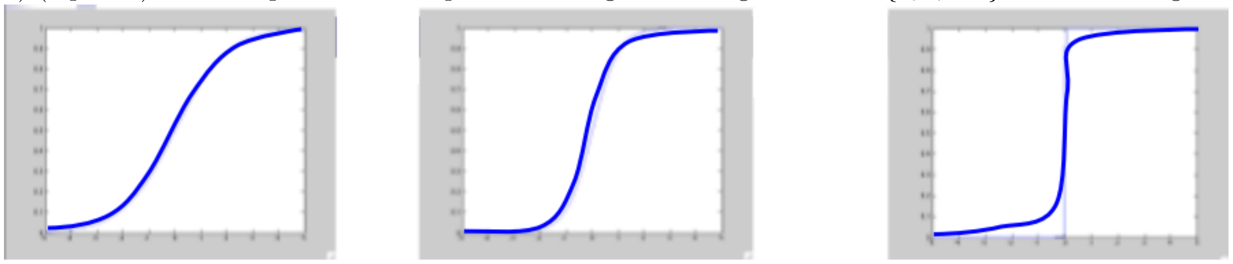
we can consider maximum conditional a posterior M(C)AP estimation:

$$\max_{w_0,\ldots,w_d} \prod_{i=1}^{n} P(Y_i|X_i, w_0, \ldots, w_d) P(w_0, \ldots, w_d)$$

where $P(w_0, \ldots, w_d)$ is a prior on the weights.

Assuming a standard Gaussian prior $\mathcal{N}(0, \boldsymbol{I})$ for the weight vector, derive the gradient ascent update rules for the weights.

a) (3 points) This is a picture of how plots of the sigmoid change with $w \in \{1, 5, 100\}$ from left to right.



As we can see, the curve gets steeper as w gets larger. The steeper curve means that the model will be nearly completely sure of the class (almost 0 probability or almost 1 probability). With large weights, small changes in input can lead to a large change in probability of class, leading to easy flipping of the predicted output. This is the intuition behind why it overfits.
Note: Award credit if the answer has the required curves and presents some reasonable intuition for why large w leads to overfitting.

b) (7 points) The derivation for the second part of the question proceeds as follows. We start by writing out the log conditional posterior (akin to log conditional likelihood for the unregularized case). Here $\boldsymbol{w} = [w_0, \ldots, w_d]^\top$.

$$L(\boldsymbol{w}) = \log(p(\boldsymbol{w}) \prod_{j=1}^{n} P(y^j | x^j, \boldsymbol{w}))$$

$$p(\boldsymbol{w}) = \prod_{i=0}^{d} \frac{1}{\sqrt{2\pi}} \exp(-\frac{w_i^2}{2})$$

Therefore, the M(C)AP estimate is

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} L(\boldsymbol{w}) = \arg\max_{\boldsymbol{w}} \left[\sum_{j=1}^{n} \log(P(y^j | x^j, \boldsymbol{w})) - \sum_i \frac{w_i^2}{2} \right]$$

The gradient ascent update rule is:

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left. \frac{\partial L(\boldsymbol{w})}{\partial w_i} \right|_t$$

The gradient of the log conditional posterior is

$$\frac{\partial L(\boldsymbol{w})}{\partial w_i} = \frac{\partial}{\partial w_i} \log \ p(\boldsymbol{w}) + \frac{\partial}{\partial w_i} \log(\prod_{j=1}^{n} P(y^j | x^j, \boldsymbol{w}))$$

The second term is same as derived in class for unregularized case. First term leads to an extra factor of

$$\frac{\partial}{\partial w_i} \log p(w) = -w_i$$

Therefore, the final update rule is

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta(-w_i^{(t)} + \sum_j x_i^j (y^j - P(Y = 1 | x^j, w^{(t)})))$$

Note: Award 3 points for the correct log conditional posterior expression or correct M(C)AP estimate expression, 2 points for the correct gradient and 2 points for the correct update step.

2. Multi-class Logistic Regression [10 pts]

One way to extend logistic regression to multi-class (say K class labels) setting is to consider (K-1) sets of weight vectors and define

$$P(Y = y_k|X) \propto \exp(w_{k0} + \sum_{i=1}^{d} w_{ki}X_i) \text{ for } k = 1, \ldots, K-1$$

a) What model does this imply for $P(Y = y_K|X)$?

b) What would be the classification rule in this case?

c) Draw a set of training data with three labels and the decision boundary resulting from a multi-class logistic regression. (The boundary does not need to be quantitatively correct but should qualitatively depict how a typical boundary from multi-class logistic regression would look like.)

a) (3 points)
Since all probabilities must sum to 1, we should have

$$P(Y = y_K|X) = 1 - \sum_{k=1}^{K-1} P(Y = y_k|X).$$

Also, note that introducing another set of weights for this class will be redundant, just as in binary classification. We can define

$$P(Y = y_K|X) = \frac{1}{1 + \sum_{k=1}^{K-1} \exp(w_{k_0} + \sum_{i=1}^{d} w_{k_i}X_i)}$$

and for $k = 1, \ldots, K-1$

$$P(Y = y_k|X) = \frac{\exp(w_{k_0} + \sum_{i=1}^{d} w_{k_i}X_i)}{1 + \sum_{k=1}^{K-1} \exp(w_{k_0} + \sum_{i=1}^{d} w_{k_i}X_i)}$$

Note: Other solutions are possible too. As long as all probabilities sum to 1 and each $P(Y = y_k|X) \propto \exp(w_{k_0} + \sum_{i=1}^{d} w_{k_i}X_i)$, you can award 2 points. Award 1 more point (i.e. full 3 points) if the students realize that they don't need an extra set of weights.

b) (3 points) The classification rule simply picks the label with highest probability:

$$y = y_{k^*} \text{ where } k^* = \arg\max_{k\in\{1,\ldots,K\}} P(Y = y_k|X)$$

c) (4 points)
The decision boundary between each pair of classes is linear and hence the overall decision boundary is piece-wise linear. Equivalently, since $\arg\max_i \exp(a_i) = \arg\max_i a_i$ and max of linear functions is piece-wise linear, the overall decision boundary is piece-wise linear.
Note: Award 4 points to an image with a piecewise-linear decision boundary.

Naive Bayes Classifier (Pulkit)

1. Naive Bayes Classification Implementation [25 pts]

In this question, you will write a Naive Bayes classifier and verify its performance on a news-group data-set. As you learned in class, Naive Bayes is a simple classification algorithm that makes an assumption about conditional independence of features, but it works quite well in practice. You will implement the Naive Bayes algorithm (Multinomial Model) to classify a news corpus into 20 different categories.

Handout - `http://www.cs.cmu.edu/~aarti/Class/10701_Spring14/assignments/homework1.tar`

You have been provided with the following data files in the handout:

- train.data - Contains bag-of-words data for each training document. Each row of the file represents the number of occurrences of a particular term in some document. The format of each row is (docId, termId, Count).
- train.label - Contains a label for each document in the training data.
- test.data - Contains bag-of-words data for each testing document. The format of this file is the same as that of the train.data file.
- test.label - Contains a label for each document in the testing data.

For this assignment, you need to write code to complete the following functions:

- logPrior(trainLabels) - Computes the log prior of the training data-set. (5 pts)
- logLikelihood(trainData, trainLabels) - Computes the log likelihood of the training data-set. (7 pts)
- naiveBayesClassify(trainData, trainLabels, testData) - Classifies the data using the Naive Bayes algorithm. (13 pts)

Implementation Notes

1. We compute the log probabilities to prevent numerical underflow when calculating multiplicative probabilities. You may refer to this article on how to perform addition and multiplication in log space.
2. You may encounter words during classification that you haven't during training. This may be for a particular class or over all. Your code should deal with that. Hint: Laplace Smoothing
3. Be memory efficient and please do not create a document-term-matrix in your code. That would require upwards of 600MB of memory.

Due to popular demand, we are allowing the solution to be coded in 3 languages: Octave, Julia, and Python.

Octave is an industry standard in numerical computing. Unlike MATLAB, it is an open-source language and has similar capabilities and syntax.

Julia is a popular new open-source language developed for numerical and scientific computing was well as beginning effective for general programming purposes. This is the first time this language is being supported in a CMU course.

Python is an extremely flexible language and is popular in industry and the data science community. Powerful python libraries would not be available to you.

For Octave and Julia, a blank function interface has been provided for you (in the handout). However, for Python, you will need to perform the I/O for the data files and ensure the results are written to the correct output files.

Challenge Question

This question is not graded, but it is highly recommended that you try it. In the above question, we are using all the terms from the vocabulary to make a prediction. This would lead to a lot of noisy features. Although it seems counter-intuitive, classifiers built from a smaller vocabulary perform better because they generalize better over unseen data. Noisy features that are not well-represented often skew the perceived distribution of words, leading to classification errors. Therefore, the classification can be improved by selecting a subset of extremely effective words.
Write a program to select a subset of the words from the vocabulary provided to you and then use this subset to run your naive bayes classification again. Verify changes in accuracy. TF-IDF and Information Theory are good places to start looking.

Support Vector Machines (Jit)

1. SVM Matching [15 points]

Figure 1 (at the end of this problem) plots SVM decision boundries resulting from using different kernels and/or different slack penalties. In Figure 1, there are two classes of training data, with labels $y_i \in \{-1, 1\}$, represented by circles and squares respectively. The SOLID circles and squares represent the Support Vectors. Determine which plot in Figure 1 was generated by each of the following optimization problems. Explain your reasoning for each choice.

1.
$$\min \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{n}\xi_i$$

   s.t. $\forall i = 1, \cdots, n$:
   $\xi_i \geq 0$
   $(\mathbf{w}\cdot\mathbf{x_i} + b)y_i - (1-\xi_i) \geq 0$
   and $C = 0.1$.

2.
$$\min \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{i=1}^{n}\xi_i \tag{3}$$

   s.t. $\forall i = 1, \cdots, n$:
   $\xi_i \geq 0$
   $(\mathbf{w}\cdot\mathbf{x_i} + b)y_i - (1-\xi_i) \geq 0$
   and $C = 1$.

3.
$$\max \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) \tag{4}$$

   s.t. $\sum_{i=1}^{n}\alpha_i y_i = 0$;
   $\alpha_i \geq 0, \forall i = 1, \cdots, n$;
   where $\mathbf{K}(\mathbf{u}, \mathbf{v}) = \mathbf{u}\cdot\mathbf{v} + (\mathbf{u}\cdot\mathbf{v})^2$.

4.
$$\max \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) \tag{5}$$

   s.t. $\sum_{i=1}^{n}\alpha_i y_i = 0$;
   $\alpha_i \geq 0, \forall i = 1, \cdots, n$;
   where $\mathbf{K}(\mathbf{u}, \mathbf{v}) = \exp(-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2})$.

5.
$$\max \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{K}(\mathbf{x_i}, \mathbf{x_j}) \tag{6}$$

   s.t. $\sum_{i=1}^{n}\alpha_i y_i = 0$;
   $\alpha_i \geq 0, \forall i = 1, \cdots, n$;
   where $\mathbf{K}(\mathbf{u}, \mathbf{v}) = \exp(- \parallel \mathbf{u}-\mathbf{v} \parallel^2)$.

One point for correctly identifying each plot, and 2 points for each explanation.

1.) **Figure C**: (1 point)
Of the five figures, only B and C correspond to a linear decision boundary *(1 point).*
The smaller error penalty C means that the decision boundary will have a larger margin, as seen by the distance from the support vectors to the decision boundary, and the slack caused by points near the boundary are essentially ignored. *(1 point)*

2.)**Figure B**: (1 point)
Of the five figures, only B and C correspond to a linear decision boundary *(1 point).*
From the distance of the support vectors from the decision boundary, we can infer that margin in this figure is smaller. Because the primal optimizations reduces the width of the margin as the penalty increase, we can conclude this is the linear kernel with the larger error penalty C. *(1 point)*

3.) **Figure D**: (1 point)
The kernel for this problem is the sum of a linear kernel and quadratic kernel, so the resulting decision boundary should be quadratic in shape.Only figure D has a quadratic decision boundary. *(2 points)*

4.) **Figure A**: (1 point)
In the gaussian RBF kernel, $k(\mathbf{u}, \mathbf{v}) = \exp(-\gamma \parallel \mathbf{u} - \mathbf{v} \parallel^2)$, a smaller γ results a decision boundary with a smoother curvature *(1 point).* Of the remaining two figures, figure A has a smoother curvature. *(1 point; Identifying that plots A and E are Gaussian RBFs)*

5.) **Figure E**: (1 point)
Because $\gamma = 1$ in this problem, the larger γ results in the gaussian RBF kernel to have a more distingushied curvature (a larger gamma means the gaussian distribution has a smaller variance). *(1 point)*
Of figure A and figure E, figue E has a more distinguished curvature. *(1 point)*

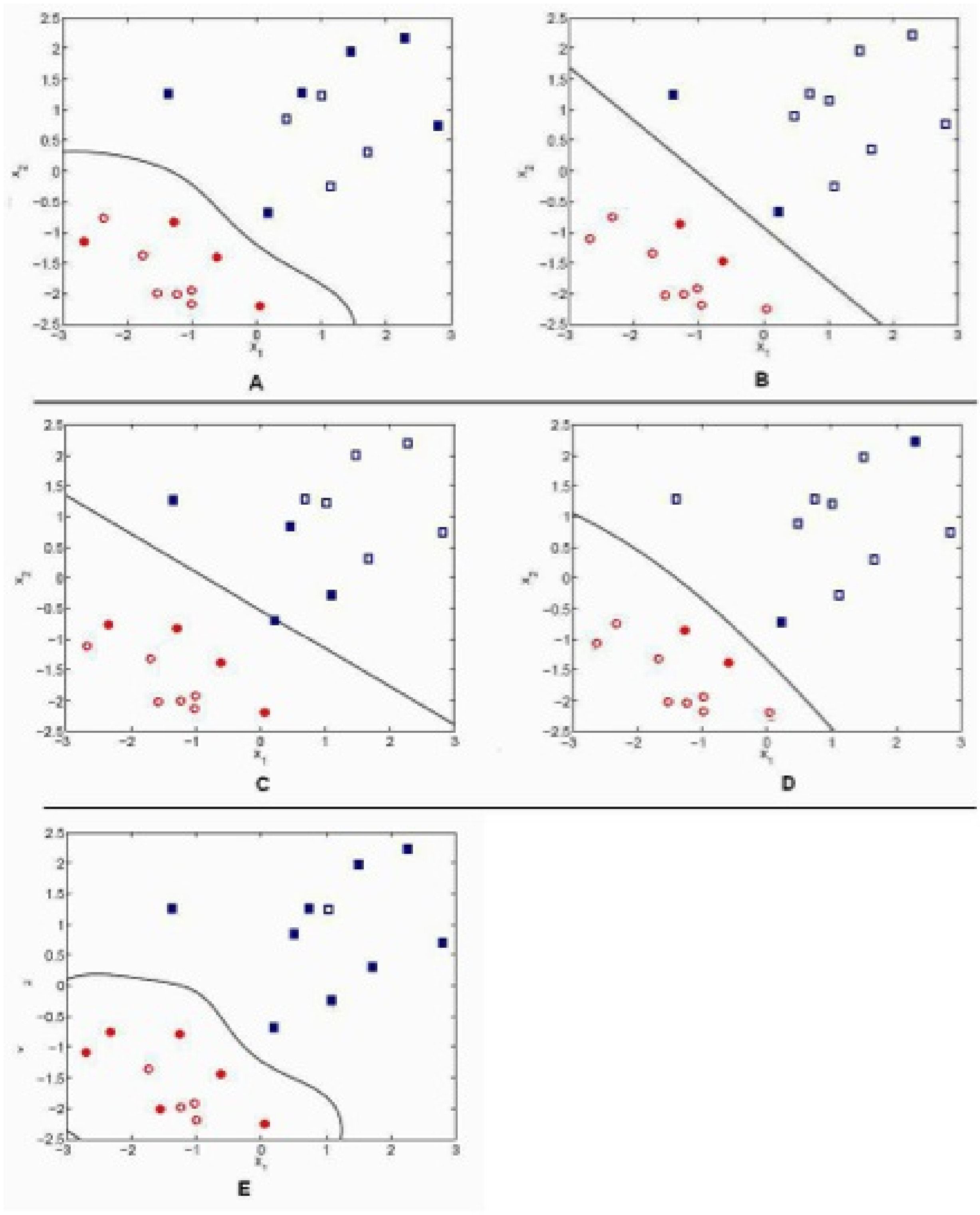


Figure 1: Induced Decision Boundaries