Homework #3

18-461/661: Intro to ML for Engineers
Prof. Gauri Joshi and Prof. Carlee Joe-Wong
Due: Friday Feb 21st, 2020 at 8:59PM PT / 11:59PM ET

Please remember to show your work for all problems and to write down the names of any students that you collaborate with. The full collaboration and grading policies are available on the course website: `https://www.andrew.cmu.edu/course/18-661/`.

Your solutions should be uploaded to Gradescope (`https://www.gradescope.com/`) in PDF format by the deadline. We will not accept hardcopies. **If you choose to hand-write your solutions, please make sure your handwriting on the uploaded copies is legible.** Gradescope will ask you to identify which page(s) contain your solutions to which problems, so make sure you leave enough time to finish this before the deadline. We will give you a 30-minute grace period to upload your solutions in case of technical problems.

1 Kernel SVM *[20 points]*

In this question, we consider the kernel version of SVMs, which has the following primal formulation:

$$\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad y^{(i)}(\mathbf{w}^T\phi(\mathbf{x})^{(i)} + b) \geq 1 - \xi_i, \forall i = 1, ..., n$$

$$\xi_i \geq 0, \forall i = 1, ..., n$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^p, i = 1, 2, ..., n$ is the original training data associated with the label $y^{(i)} \in \{-1, 1\}$, $C > 0$ is a constant, and the function $\phi : \mathbb{R}^p \to \mathbb{R}^d$ maps the original data $\mathbf{x}^{(i)}$ to a new feature vector $\phi(\mathbf{x}^{(i)})$ in $\mathbb{R}^d$. We optimize over the model weights $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, as well as the slack variables $\xi_i \in \mathbb{R},\ i = 1, 2, ..., n$.

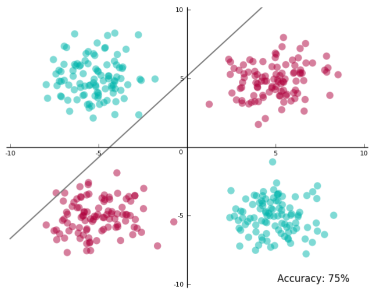
a. *[5 points]* Recall that the dual formulation of the SVM problem can be written as

$$\max_{\alpha} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \phi(\mathbf{x}_m)^T\phi(\mathbf{x}_n) \quad \text{s.t.} \quad \sum_n \alpha_n y_n = 0,\ 0 \leq \alpha_n \leq C$$

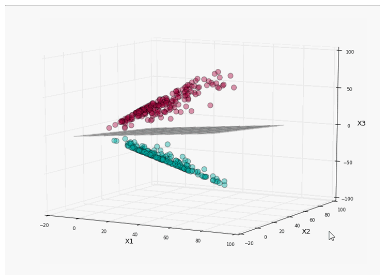
where α_n denotes the dual variables. Suppose we have obtained the solution of the dual problem, denoted as $\alpha_i^\star$ for $i = 1, 2, ..., n$. In lecture, we saw that when we do not define the transformation ϕ, the optimal $\mathbf{w}^\star$ and $b^\star$ can be written in terms of the optimal dual variables $\alpha_i^\star$. Given the SVM formulation with the transformation ϕ as above, find $\mathbf{w}^\star$ and $b^\star$ in terms of the optimal dual variables $\alpha_i^\star$.

b. *[4 points]* How does the dual formulation change when we use kernel functions, i.e. $k(\mathbf{u}, \mathbf{v})$, defined over a pair of inputs $(\mathbf{u}, \mathbf{v})$? The kernel function $k(\mathbf{u}, \mathbf{v})$ measures similarity between inputs $(\mathbf{u}, \mathbf{v})$.

c. Consider a test data point $\mathbf{z} \in \mathbb{R}^p$. We wish to use the kernel SVM classifier derived above to make a classification decision for $\mathbf{z}$.



(a) Original dataset.



(b) Transformed dataset.

Figure 1: Example of original and transformed data for Q1c.

(1) *[3 points]* Under what conditions would we classify $\mathbf{z}$ as having the label 1?

(2) *[4 points]* Suppose that the mapping $\phi(\mathbf{x})$ is given by the following for $\mathbf{x} = [x_1, \cdots, x_p]^T$:

$$\phi(\mathbf{x}) = \Big(\underbrace{x_p^2, x_{p-1}^2, \cdots, x_1^2}_{\text{length } p}, \underbrace{\sqrt{2}x_p x_{p-1}, \cdots, \sqrt{2}x_p x_1}_{\text{length } p-1}, \underbrace{\sqrt{2}x_{p-1}x_{p-2}, \cdots, \sqrt{2}x_{p-1}x_1}_{\text{length } p-2}, \cdots, \sqrt{2}x_2 x_1\Big)^T.$$

Note that the variable under the brace denotes the number of terms in the brace. For example, if $p = 2$, then

$$\phi(\mathbf{x}) = \left(x_2^2, x_1^2, \sqrt{2}x_2x_1\right)^T,$$

which transforms the 2-dimensional dataset shown in Figure 1a (not linearly separable) to the linearly separable 3-dimensional dataset in Figure 1b.

What is the time complexity of making the classification decision (i.e., how does the number of operations required to compute the condition above grow with p)? Write your answer using big-O notation.[1]

(3) *[4 points]* Define the kernel function $K(\mathbf{u}, \mathbf{v}) = \phi(\mathbf{u})^T\phi(\mathbf{v})$ where $\phi(\cdot)$ has the same definition as above. What is the time complexity of making the classification decision if we directly compute $K(\cdot, \cdot)$ without computing $\phi(\mathbf{u})$ and $\phi(\mathbf{v})$? How does this compare to the complexity when we directly compute ϕ as above?

Hint: When $p = 2$, it is easy to see that $K(\mathbf{u}, \mathbf{v}) = u_1^2v_1^2 + 2u_1u_2v_1v_2 + u_2^2v_2^2 = (u_1v_1 + u_2v_2)^2 = (\mathbf{u}^T\mathbf{v})^2$. Can you generalize this expression for $p > 2$?

2 Double Counting the Evidence *[20 points]*

Consider the two-class classification problem where the class label $y \in \{0; 1\}$ and each training example X has 2 binary attributes $X_1, X_2 \in \{0; 1\}$. Let the class prior be $\Pr(Y = 1) = 0.5$ and also let

$$\Pr(X_1 = 1 \mid Y = 1) = 0.8, \quad \Pr(X_1 = 0 \mid Y = 0) = 0.7,$$
$$\Pr(X_2 = 1 \mid Y = 1) = 0.5, \quad \Pr(X_2 = 0 \mid Y = 0) = 0.9.$$

So, attribute X_1 provides slightly stronger evidence about the class label than X_2.

[1] If you are unfamiliar with big-O notation, this blog post has a good tutorial: `https://rob-bell.net/2009/06/a-beginners-guide-to-big-o-notation/`.

a. *[6 points]* Assume X_1 and X_2 are truly independent given Y. Write down the Naive Bayes decision rule given $X_1 = x_1$ and $X_2 = x_2$. Fill out the following table of predictions, $f(X_1, X_2) \in \{0, 1\}$, based on the Naive Bayes decision rule for each of the 4 settings for X_1, X_2. Please show your calculations.

X_1	X_2	$f(X_1, X_2)$
0	0	
0	1	
1	0	
1	1	

b. *[6 points]* For the Naive Bayes decision function $f(X_1, X_2)$, the error rate is:

$$\sum_{X_1, X_2, Y} \mathbb{I}(Y \neq f(X_1; X_2)) \Pr(X_1, X_2, Y),$$

where $\mathbb{I}(Y \neq f(X_1, X_2)) = 1$ if $Y \neq f(X_1; X_2)$ and 0 otherwise. For this question, we will assume that the true data distribution is exactly the same as the Naive Bayes distribution, so we can write $\Pr(X_1, X_2|Y)$ as $\Pr(Y)\Pr(X_1 \mid Y)\Pr(X_2 \mid Y)$. Show the error rate of the Naive Bayes Classifier is 0.235.

c. Now, suppose that we create a new attribute X_3, which is an exact copy of X_2. So, for every training example, attributes X_2 and X_3 have the same value, $X_2 = X_3$.

(a) *[2 points]* What is the error rate of Naive Bayes now, using X_1, X_2, and X_3? The predicted Y should be computed using the assumption of conditional independence, and the error rate should be computed using the true probabilities.

(b) *[2 points]* Why does Naive Bayes perform worse with the addition of X_3?
Hint: Does the key assumption of Naive Bayes still hold?

(c) *[2 points]* Now consider a logistic regression model $\mathcal{M}$, with weight vector $\mathbf{w} = [w_1, w_2]$ that is used to predict Y using X_1 and X_2 and another model $\mathcal{M}'$, with weight vector $\mathbf{w}' = [w_1', w_2', w_3']$ that is used to predict Y using X_1, X_2, and X_3. What is the relation between $\mathbf{w}$ and $\mathbf{w}'$ after training both models?

(d) *[2 points]* Will the trained logistic regression model $\mathcal{M}'$ suffer from the same problem as the Naive Bayes model in part (a)? Explain why or why not.

3 Gradient Descent for Logistic Regression *[50 points]*

In this problem, you will implement (unregularized/regularized) logistic regression for binary classification problems using gradient descent. Note that **you may not use any libraries/packages that implement gradient descent - you must implement it yourself.** *Cross validation is NOT needed* for this programming assignment. *For all plots you make, you MUST provide title, x-label, y-label and legend (if there is more than one curve).* **Note that you need to submit your source code as part of the homework PDF that you submit.**

3.1 EmailSpam Data

This dataset contains 941 instances and each of them is labeled as either a spam or ham (not spam) email. Each data instance is the text which is the content of the email (subject and body), and your goal is to

Algorithm 1 Pseudocode for generating bag-of-word features from text

```
1: Initialize feature vector bg_feature = [0,0,...,0]
2: for token in text.tokenize() do
3:   if token in dict then
4:     token_idx = getIndex(dict, token)
5:     bg_feature[token_idx]++
6:   else
7:     continue
8:   end if
9: end for
10: return bg_feature
```

classify each email as either a spam or a ham. We have already divided this dataset into training and test datasets, and each of these datasets is stored in two distinct folders (one corresponding to ham and the other corresponding the spam). In other words, to build your model, you need to iterate through all the text files within `/train/spam` and `/train/ham`, and to test your model you need to iterate through `/test/spam` and `/test/ham`.

3.2 Feature Representation

An essential part of machine learning is to build the feature representation for raw input data, which is often unstructured. Once we construct the features, each data instance $\mathbf{x}_i$ can be represented as:

$$\mathbf{x}_i = (x_{i1}, ..., x_{id})$$

where x_{ij} denotes the jth feature for the ith instance.

Since each data instance is text, you need to find a way of converting it into a feature vector. In this problem, you will use the "Bag-of-Words" representation. More specifically, you will convert the text into a feature vector in which each entry of the vector is the count of words that occur in that text. You will be provided with the predefined dictionary (`dic.dat`), and you should only consider words that appear in that dictionary and ignore all others.[2] Above is the pseudocode for generating bag-of-word features from text. For tokenization, please tokenize the string only using **whitespace and these three delimiters: '.,?'**. See below for an example:[3]

Email: *hey, i have a better offer for you, offer. better than all other spam filters. Do you like accepting offer?*

Pre-defined Dictionary: *[add, better, email, filters, hey, offer,like, spam,special]*

Bag-of-words feature vector: $[0, 2, 0, 1, 1, 3, 1, 1, 0]$

(Q3.2). *[8 points]* **After converting all training data into bag-of-words feature vectors, what are the 3 words that occur most frequently?** Report the results using this format:

{(word1: # of occurrences), (word2: # of occurrences), (word3: # of occurrences)}

[2]In practice, you need to define the dictionary based on the specific requirements of your application. This process usually involves removing "stop words", "stemming", etc. In this homework, you don't need to consider these issues.

[3]http://en.wikipedia.org/wiki/Bag-of-words_model also provides a simple example.

3.3 Implementation

The **regularized cross-entropy** function can be written as:

$$\varepsilon(\mathbf{w}, b) = -\sum_{i=1}^{n}\{y_i \log \sigma(b + \mathbf{w}^\top \mathbf{x}_i) + (1 - y_i)\log[1 - \sigma(b + \mathbf{w}^\top \mathbf{x}_i)]\} + \lambda\|\mathbf{w}\|_2^2\,,$$

where $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$, and $\sigma(\cdot)$ is the sigmoid function. λ is the regularization coefficient and b is the bias parameter. Note that we don't regularize the bias term b. Please use the following configuration when implementing batch gradient descent below.

Stopping Criteria. Run for 50 iterations.
Step size. Use a fixed step size (i.e., set the step size to a constant number).
Initialization. Initialize the weight $\mathbf{w}$ to 0, and b to 0.1.
Extreme Condition. It is possible that when $\sigma(b + \mathbf{w}^T\mathbf{x})$ approaches 0, $\log(\sigma(b + \mathbf{w}^T\mathbf{x}))$ goes to -infinity. In order to prevent such case, please bound the value $\sigma(b + \mathbf{w}^T\mathbf{x})$ to be larger than the small constant value, $1e - 16$. You can use the following logic in your code:

```
$tmp = \sigma(b + \mathbf{w}^T\mathbf{x})$
if $tmp < 1e - 16$ then
    $tmp = 1e - 16$
```

Similarly, you may need to bound the value of $1 - \sigma(b + \mathbf{w}^T\mathbf{x})$ when $\sigma(b + \mathbf{w}^T\mathbf{x})$ is close to 1.

3.4 Batch Gradient Descent

(Q3.4a) *[8 points]* Write down the gradient descent updates for $\mathbf{w}$ and b, for **both unregularized logistic regression and regularized logistic regression**. In particular, at iteration t using data points $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, where $\mathbf{x}_i = [x_{i1}, ..., x_{id}]$, and $y_i \in \{0, 1\}$ is the label, how do we compute $\mathbf{w}^{t+1}$ and b^{t+1} from $\mathbf{w}^t$ and b^t?

(Q3.4b) *[14 points]* For step sizes $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5\}$ and **without regularization**, implement batch gradient descent (using the whole training data for the gradient calculation).

- Plot the cross-entropy function value with respect to the number of iterations ($T = [1, ..., 50]$) for the training data for each step size.
- Report the L_2 norm of the vector $\mathbf{w}$ after 50 iterations for each step size η_i (fill in the table below).

Step size	$\eta = 0.001$	$\eta = 0.01$	$\eta = 0.05$	$\eta = 0.1$	$\eta = 0.5$
L_2 norm (without regularization)					

(Q3.4c) *[20 points]* For step sizes $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5\}$ and with regularization coefficients $\lambda = \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$, do the following:

- Setting $\lambda = 0.1$, plot the cross-entropy function value with respect to the number of iterations ($T = [1, ..., 50]$) for the training data for each step size.
- Setting $\eta = 0.01$, report the L_2 norm of the vector $\mathbf{w}$ after 50 iterations for each regularization coefficient λ_i (fill in the table below).

Regularization parameter	$\lambda = 0$	$\lambda = 0.1$	$\lambda = 0.2$	$\lambda = 0.3$	$\lambda = 0.4$	$\lambda = 0.5$
L_2 norm (with $\eta = 0.01$)						

- Plot the cross-entropy function value with respect to $\lambda = 0, 0.1, \ldots, 0.5$ for the *test* data when $\eta = 0.05$.

4 SVMs: Hinge loss and mistake bounds *[10 points]*

Suppose we build a predictive model $\hat{y} = f_{\mathbf{w}}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x})$, where $\text{sgn}(z) = 1$ if z is positive and -1 otherwise. Here we are dealing with binary predictions where each label is in $\{-1, 1\}$. The classification loss counts the number of mistakes (i.e., points where $y \neq \text{sgn}(\mathbf{w}^T\mathbf{x})$) that we make with $\mathbf{w}$ on a dataset. The SVM loss function can be viewed as a relaxation to the classification loss. The *hinge loss* on a pair $(\mathbf{x}, y)$ is defined as:

$$\ell((\mathbf{x}, y), \mathbf{w}) = \max\left[0, 1 - y(\mathbf{w}^T\mathbf{x})\right], \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^D$, $\mathbf{w} \in \mathbb{R}^D$, and $y \in \{-1, 1\}$. The SVM attempts to minimize:

$$\frac{1}{N}\sum_{i=1}^{N} \ell((\mathbf{x}_i, y_i), \mathbf{w}) + \lambda\|\mathbf{w}\|^2, \tag{2}$$

where $\lambda \geq 0$ is the regularization parameter.

Suppose that for some $\mathbf{w}$ we have a correct prediction of y_i with $\mathbf{x}_i$, i.e. $y_i = \text{sgn}(\mathbf{w}^T\mathbf{x})$. What range of values can the hinge loss, $\ell((\mathbf{x}, y), \mathbf{w})$, take on this correctly classified example? Points that are classified correctly and which have non-zero hinge loss are referred to as *margin mistakes*.