# Homework #3

ECE 661: Introduction to Machine Learning
Prof. Carlee Joe-Wong and Prof. Virginia Smith
**Due: Thursday October 4th, 2018 at 8:30AM PT / 11:30AM ET**

September 30, 2018

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://18661.github.io/.

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 the uploaded copies are 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 Bias-Variance Tradeoff

**[25 points]** Consider a dataset with $n$ data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$, drawn from the following linear model:

$$y = \mathbf{x}^\top \boldsymbol{\beta}^\star + \varepsilon,$$

where $\varepsilon \sim N(0, \sigma^2)$ is Gaussian noise and the star sign $(^\star)$ is used to differentiate the true parameter from the estimators that will be introduced later. Consider the $L_2$-regularized linear regression model:

$$\widehat{\boldsymbol{\beta}}_\lambda = \text{argmin}_{\boldsymbol{\beta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\},$$

where $\lambda \geq 0$ is the regularization parameter. Let $X \in \mathbb{R}^{n\times p}$ denote the matrix obtained by stacking $\mathbf{x}_i^\top$ in each row. Recall that the resulting closed form solution for $\widehat{\boldsymbol{\beta}}_\lambda$ can be written as follows:

$$\widehat{\boldsymbol{\beta}}_\lambda = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y} = (X^\top X + \lambda I)^{-1} X^\top (X\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon}).$$

Note that properties of an affine transformation of a Gaussian random variable will be useful throughout this problem.

a. Calculate the bias term $\mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda] - \mathbf{x}^\top \boldsymbol{\beta}^\star$ as a function of $\lambda$ and a *fixed* test point $\mathbf{x}$.

b. Calculate the variance term $\mathbb{E}\left[\left(\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda - \mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda]\right)^2\right]$ as a function of $\lambda$ and a *fixed* test point $\mathbf{x}$.

c. Use the results from parts (b) and (c) and the bias–variance tradeoff that we saw in class to analyze the impact of $\lambda$ in the squared error. Specifically, which term dominates when $\lambda$ is small or large?

## 2 SVMs: Hinge loss and mistake bounds

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

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

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

a. Show that the function $\ell((\mathbf{x}, y), \mathbf{w}) = \max\left[0, 1 - y\mathbf{w}^T\mathbf{x}\right]$ is convex (as a function of $\mathbf{w}$).

b. 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*.

## 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 for 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, in the same PDF file.**

### 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 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 $j$th feature for the $i$th 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

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

```
Initialize feature vector bg_feature = [0,0,...,0]
for token in text.tokenize() do
    if token in dict then
        token_idx = getIndex(dict, token)
        bg_feature[token_idx]++
    else
        continue
    end if
end for
return bg_feature
```

provided with the predefined dictionary (`dic.dat`), and you should only consider words that appear in that dictionary and ignore all others.[1] 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:[2]

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). 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)}

## 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. $\lambda$ is the regularization coefficient and $w_0$ 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 $\sigma(b + \mathbf{w}^T\mathbf{x})$ using small constant value, $1e - 16$. You can use the following logic in your code:
$tmp = \sigma(b + \mathbf{w}^T\mathbf{x})$

[1] 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.
[2] http://en.wikipedia.org/wiki/Bag-of-words_model also provides a simple example.

if $tmp < 1e-16$ then
  $tmp = 1e-16$

## 3.4 Batch Gradient Descent

**(Q3.4a)** 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)** 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 steps ($T = [1, ..., 50]$) 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 $\eta_i$ (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)** 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 steps ($T = [1, ..., 50]$) 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 $\lambda_i$ (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 $\eta = 0.01$) | | | | | | |