# DS-GA 3001.007: Introduction to Machine Learning
# Homework 4

**Due:** Monday November 11 at 11:59PM. Remember to submit a pdf to Gradescope. You must mark the pages for each answer in the pdf.

**Instructions**: Your answers to the questions below, including plots and mathematical work, should be submitted as a single PDF file. It's preferred that you write your answers using software that typesets mathematics (e.g. LaTeX, LyX, or MathJax via iPython), though if you need to you may scan handwritten work. You may find the minted package convenient for including source code in your LaTeX document. If you are using LyX, then the listings package tends to work better.

## 1 Introduction

In this assignment, we'll be working with the data from Homework 1 on emails. However, we will use a different approach. You will be implementing Pegasos. Pegasos is essentially stochastic subgradient descent for the SVM with a particular schedule for the step-size. We have provided you with two files: `spam train.txt`, `spam test.txt`. Each row of the data files corresponds to a single email. The first column gives the label (1=spam, 0=not spam).

## 2 Calculating Subgradients

Recall that a vector $g \in \mathbf{R}^d$ is a **subgradient** of $f : \mathbf{R}^d \to \mathbf{R}$ at $x$ if for all $z$,

$$f(z) \geq f(x) + g^T(z - x).$$

As we noted in lecture, there may be 0, 1, or infinitely many subgradients at any point. The **subdifferential** of $f$ at a point $x$, denoted $\partial f(x)$, is the set of all subgradients of $f$ at $x$.

Just as there is a calculus for gradients, there is a calculus for subgradients[1]. For our purposes, we can usually get by using the definition of subgradient directly. However, in the first problem below we derive a property that will make our life easier for finding a subgradient of the hinge loss and perceptron loss.

1. [Subgradients for pointwise maximum of functions] Suppose $f_1, \ldots, f_m : \mathbf{R}^d \to \mathbf{R}$ are convex functions, and
$$f(x) = \max_{i=1,\ldots,,m} f_i(x).$$

[1] A good reference for subgradients are the course notes on Subgradients by Boyd et al.

Let $k$ be any index for which $f_k(x) = f(x)$, and choose $g \in \partial f_k(x)$. [We are using the fact that a convex function on $\mathbf{R}^d$ has a non-empty subdifferential at all points.] Show that $g \in \partial f(x)$.

2. [Subgradient of hinge loss for linear prediction] Give a subgradient of

$$J(w) = \max\left\{0, 1 - yw^T x\right\}.$$

# 3 Perceptron

We studied the perceptron algorithm in homework 1. Suppose we have a labeled training set $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbf{R}^d \times \{-1, 1\}$. In the perceptron algorithm, we are looking for a hyperplane that perfectly separates the classes. That is, we're looking for $w \in \mathbf{R}^d$ such that

$$y_i w^T x_i > 0 \; \forall i \in \{1, \ldots, n\}.$$

Visually, this would mean that all the $x$'s with label $y = 1$ are on one side of the hyperplane $\left\{x \mid w^T x = 0\right\}$, and all the $x's$ with label $y = -1$ are on the other side. When such a hyperplane exists, we say that the data are **linearly separable**. The perceptron algorithm is given in Algorithm 1.

**Algorithm 1:** Perceptron Algorithm

input: Training set $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbf{R}^d \times \{-1, 1\}$
$w^{(0)} = (0, \ldots, 0) \in \mathbf{R}^d$
$k = 0$ # step number
repeat
  all_correct = TRUE
  for $i = 1, 2, \ldots, n$ # loop through data
    if $(y_i x_i^T w^{(k)} \leq 0)$
      $w^{(k+1)} = w^{(k)} + y_i x_i$
      all_correct = FALSE
    else
      $w^{(k+1)} = w^{(k)}$
    end if
    $k = k + 1$
  end for
until (all_correct == TRUE)
return $w^{(k)}$

There is also something called the **perceptron loss,** given by

$$\ell(\hat{y}, y) = \max\left\{0, -\hat{y}y\right\}.$$

In this problem we will see why this loss function has this name.

1. Show that if $\left\{x \mid w^T x = 0\right\}$ is a separating hyperplane for a training set $\mathcal{D} = ((x_1, y_1), \ldots, (x_n, y_n))$, then the average perceptron loss on $\mathcal{D}$ is 0. Thus any separating hyperplane of $\mathcal{D}$ is an empirical risk minimizer for perceptron loss.

2. Let $\mathcal{H}$ be the linear hypothesis space consisting of functions $x \mapsto w^T x$. Consider running stochastic subgradient descent (SSGD) to minimize the empirical risk with the perceptron loss. We'll use the version of SSGD in which we cycle through the data points in each epoch. Show that if we use a fixed step size 1, we terminate when our training data are separated, and we make the right choice of subgradient, then we are exactly doing the Perceptron algorithm.

3. Suppose the perceptron algorithm returns $w$. Show that $w$ is a linear combination of the input points. That is, we can write $w = \sum_{i=1}^{n} \alpha_i x_i$ for some $\alpha_1, \ldots, \alpha_n \in \mathbf{R}$. The $x_i$ for which $\alpha_i \neq 0$ are called support vectors. Give a characterization of points that are support vectors and not support vectors.

# 4 Support Vector Machine

## 4.1 Practice with Margins

Consider a (hard margin) support vector machine and the following training data from two classes:

$$+1 : (2,2)\ (4,4)\ (4,0)$$

$$-1 : (0,0)\ (2,0)\ (0,2)$$

1a. Plot these six training points, and construct by inspection the weight vector for the optimal hyperplane. In your solution, specify the hyperplane in terms of $w$ and $b$ such that $w_1 x_1 + w_2 x_2 + b = 0$. Calculate what the margin is (i.e., $2\gamma$, where $\gamma$ is the distance from the hyperplane to its closest data point), showing all of your work.

1b. What are the support vectors? Explain why.

2. Show that, irrespective of the dimension of the data space, a data set consisting of just two data points (call them $x_1$ and $x_2$), one from each class, is sufficient to determine the maximum-margin hyperplane. Fully explain your answer, including giving an explicit formula for the solution to the hard margin SVM (i.e., $w$) as a function of $x_1$ and $x_2$.

## 4.2 Pegasos

In this question you will implement the Pegasos algorithm to optimize the SVM objective using stochastic subgradient descent. Here we use the objective

$$\frac{\lambda}{2}||w||^2 + \frac{1}{m}\sum_{i=1}^{m} \max\left\{0, 1 - y_i(w \cdot x_i)\right\}$$

We use the identical setting as in the problem set 1. Split the data in spam train.txt into a training and validate set, putting the last 1000 emails into the validation set. Transform all of the data into

feature vectors. Build a vocabulary list using only the 4000 e-mail training set by finding all words that occur across the training set. Ignore all words that appear in fewer than X = 30 e-mails of the 4000 e-mail training set. For each email, transform it into a feature vector $x$ where the ith entry, $x_i$ , is 1 if the $i$th word in the vocabulary occurs in the email, and 0 otherwise.

Initialize: Choose $\boldsymbol{w}_1 = 0, t = 0$.
1. For iter $= 1, 2, \cdots, 20$
2. For $j = 1, 2, \cdots, |\text{data}|$
3. $t = t + 1; \quad \eta_t = \frac{1}{t\lambda}$
5. If $y_j(\boldsymbol{w}_t \cdot \boldsymbol{x}_j) < 1$
6. $\boldsymbol{w}_{t+1} = (1 - \eta_t\lambda)\boldsymbol{w}_t + \eta_t y_j \boldsymbol{x}_j$
7. Else
8. $\boldsymbol{w}_{t+1} = (1 - \eta_t\lambda)\boldsymbol{w}_t$
8. Output: $\boldsymbol{w}_{t+1}$

Note: To keep your algorithm simple, we will not use an offset term $b$ when optimizing the SVM primal objective using Pegasos.

1. Implement the function pegasos_svm_train(data, w). The function should return $w$, the final classification vector. For simplicity, the stopping criterion is set so that the total number of passes over the training data is 20. After each pass through the data, evaluate the SVM objective

$$f(w_t) = \frac{\lambda}{2}||w_t||^2 + \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i(w_t \cdot x_i)\}$$

   and store its value ($m = |\,\text{data}\,|$). Plot $f(w_t)$ as a function of iteration (i.e. for $t = |\,\text{data}\,|, \ldots, 20|\,\text{data}\,|$), and submit the plot for $\lambda = 2^{-5}$

2. Implement the function pegasos_svm_test(data, w).

3. Run your learning algorithm for various values of the regularization constant, $\lambda = 2^{-9}, 2^{-8}, \ldots, 2^{1}$. Plot the average training error, average hinge loss of the training samples $\frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y_i(w_t \cdot x_i)\}$, and average validation error as a function of $\log_2 \lambda$.

4i You should expect that the hinge loss upper-bounds the training error. What is the minimum of your validation error?

4ii For the classifier that has the smallest validate error: What is the test error?

4iii How many training samples are support vectors? How did you find them?

4iv Compare your test error with your result from problem set 1.

Note: Using a smaller value of X such as 0 (i.e., not filtering the vocabulary) would give even better results (the SVM's regularization prevents overfitting). If you try this, we recommend you use sparse matrices (e.g., with Python's `scipy.sparse`) as the feature vectors will be large but sparse and this will improve efficiency.