# CSE 546 Machine Learning, Autumn 2014
# Homework 3

Due: Tuesday, November 18, beginning of class

## 1 Fitting an SVM classifier by hand [50 Points]

(Source: Murphy text, Exercise 14.1) Consider a dataset with 2 points in 1d:

$$(x_1 = 0, y_1 = -1) \text{ and } (x_2 = \sqrt{2}, y_2 = 1).$$

Consider mapping each point to 3d using the feature vector $\phi(x) = [1, \sqrt{2}x, x^2]^T$ (This is equivalent to using a second order polynomial kernel). The max margin classifier has the form

$$\hat{w}, \hat{w_0} = \arg\min ||w||^2 \quad s.t. \tag{1}$$
$$y_1(w^T\phi(x_1) + w_0) \geq 1 \tag{2}$$
$$y_2(w^T\phi(x_2) + w_0) \geq 1 \tag{3}$$

1. *(10 Points)* Write down a vector that is parallel to the optimal vector $\hat{w}$. Hint: Recall from Figure 14.12 (page 500 in the Murphy text) that $\hat{w}$ is perpendicular to the decision boundary between the two points in the 3d feature space.

2. *(10 Points)* What is the value of the margin that is achieved by this $\hat{w}$? Hint: Recall that the margin is the distance from each support vector to the decision boundary. Hint 2: Think about the geometry of the points in feature space, and the vector between them.

3. *(10 Points)* Solve for $\hat{w}$, using the fact the margin is equal to $1/||\hat{w}||$.

4. *(10 Points)* Solve for $\hat{w_0}$ using your value for $\hat{w}$ and Equations 1 to 3. Hint: The points will be on the decision boundary, so the inequalities will be tight. A "tight inequality" is an inequality that is as strict as possible. For this problem, this means that plugging in these points will push the left-hand side of Equations 2 and 3 as close to 1 as possible.

5. *(10 Points)* Write down the form of the discriminant function $f(x) = \hat{w_0} + \hat{w}^T\phi(x)$ as an explicit function of $x$. Plot the 2 points in the dataset, along with $f(x)$ in a 2d plot. You may generate this plot by hand, or using a computational tool like Python.

**Show your work.**

# 2 Programming Question [50 Points]

In this problem, we seek to perform a digit recognition task, where we are given an image of a handwritten digit and wish to predict what number it represents. This is a special case of an important area of language processing known as Optical Character Recognition (OCR). We will be simplifying our goal to that of a binary classification between two relatively hard-to-distinguish numbers (specifically, predicting a 3 versus a 5). To do this, you will implement a linear support vector machine (SVM).

## 2.1 Dataset

The digit images have been taken from a Kaggle competition, `http://www.kaggle.com/c/digit-recognizer/data`. This data was originally from the MNIST database of handwritten digits, but was converted into a easier-to-use file format.

The data have also undergone some preprocessing. They have been filtered to just those datapoints whose labels are 3 or 5, which have then been relabeled to 1 and -1 respectively. Then, we subsampled to create two datasets `validation.csv` and `test.csv` ($N = 1000$ for both). Each row in these files represents an image. The first column is the label of the image, and the remaining columns give the grayscale values for each pixel. We will use `validation.csv` as training data, and `test.csv` as test data.

**Preprocessing** The data provided in the csv file is in the range $[0, 256)$. Normally when you work with a linear model, you want to normalize the data so that the range is closer to $[0, 1]$. There are many ways to perform data normalization, and in this homework we use the following L2 normalization

$$\mathbf{x} \leftarrow \mathbf{x} / \|\mathbf{x}\| \tag{4}$$

Here $\|\mathbf{x}\|$ is the L2 norm of $\mathbf{x}$. This normalization will make each input vector have unit length, and it usually works well in practice. You can use the following code to load and normalize the data:

```
def loaddata(fname):
    data = np.loadtxt(fname,skiprows=1, delimiter=',')
    y = data[:,0]
    X = data[:,1:]
    nm = np.sqrt(np.sum(X * X, axis=1))
    X = X / nm[:,None]
    return y, X
trainY, trainX = loaddata('validation.csv')
testY, testX = loaddata('test.csv')
```

## 2.2 Implement Linear SVM

The objective of a linear SVM can be written as

$$\arg\min_{\mathbf{w}, w_0} \|\mathbf{w}\|_2^2 + C \sum_{j=1}^{N} l(\mathbf{w}, w_0, \mathbf{x}^{(j)}, y^{(j)}) \tag{5}$$

Here $l(\mathbf{w}, w_0, \mathbf{x}^{(j)}, y^{(j)})$ is the *hinge loss* of the $j$-th instance:

$$l(\mathbf{w}, w_0, \mathbf{x}^{(j)}, y^{(j)}) = \max\left(1 - y^{(j)}(\mathbf{w}^T\mathbf{x}^{(j)} + w_0), 0\right)$$

Note that this objective value grows with $N$ and $C$. In this homework, we will use an equivalent formalization, by multiplying the objective with $\frac{1}{NC}$, so that the objective and its gradient have the same scale as $N$ and $C$ changes.

$$\arg\min_{\mathbf{w}, w_0} \frac{1}{NC} \|\mathbf{w}\|_2^2 + \frac{1}{N} \sum_{j=1}^{N} l(\mathbf{w}, w_0, \mathbf{x}^{(j)}, y^{(j)}) \tag{6}$$

We can use stochastic gradient descent to optimize this objective. The update rule for $\mathbf{w}_i$ with the $j$-th instance will be

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} - \eta \, \partial_{\mathbf{w}_i} \tilde{L}^{(j)}(\mathbf{w}^{(t)}, w_0^{(t)}) \tag{7}$$

where $\tilde{L}^{(j)}$ is appromixation of the loss function using only the $j$-th sample

$$\tilde{L}^{(j)}(\mathbf{w}, w_0) = \frac{1}{NC} \|\mathbf{w}\|_2^2 + l(\mathbf{w}, w_0, \mathbf{x}^{(j)}, y^{(j)}) \tag{8}$$

and $\partial_{\mathbf{w}_i} \tilde{L}^{(j)}(\mathbf{w}^{(t)}, w_0^{(t)})$ is the subgradient of the approximate loss over $\mathbf{w}_i$. A similar update rule can be applied to $w_0$.

1. *(4 points)* Write the stochastic gradient descent update rules for both $\mathbf{w}$ and $w_0$ in linear SVMs.

2. Implement SGD for linear SVMs. Initially start all the weights at 0.

3. *(9 points)* Using `validation.csv` as your training set, run 50 passes over the dataset (i.e. $50 \times 1000$ SGD updates), using $\eta = 0.03$ and $C = 1$. Plot the loss in Eq. (6) after each pass.

4. *(9 points)* Using the $\hat{\mathbf{w}}$, $\hat{w_0}$ learned after 50 passes, report:

   (a) The prediction error on `test.csv` (test error)

   (b) The prediction error on `validation.csv` (training error)

   (c) $\|\mathbf{w}\|$

5. *(18 points)* Change $C$ to 100 and repeat what you did in the previous two questions, using the same values for $\eta$ and the number of passes.

6. *(10 points)* Compare the results you get using $C = 100$ vs. $C = 1$ on the test/training error and $\|\mathbf{w}\|$. Explain the difference using the bias-variance tradeoff. (Hint: Which setting has higher variance?)