Machine Learning & Data Mining
Homework 1

Caltech CS/CNS/EE 155
Released January 5^{th}, 2016

- Due 2pm, January 12^{th}, via Moodle.
- You are free to collaborate on all of the problems, subject to the collaboration policy stated in the syllabus.
- The first two sections (Basics and Cross Validation) are meant to test your basic understanding of machine learning concepts.
- The next two sections (Perceptron and Gradient Descent) are meant to test your ability to successfully understand and implement basic algorithms.
- If you have trouble with this homework, it may be an indication that you should drop the class.

1 Basics

Give a short 1-2 sentence answer to the following questions.

Question A: What is a hypothesis set? Specifically, what is the hypothesis set of a linear model?

Solution A: *The hypothesis set is a set of functions (i.e. a function space) from which the hypothesis function $h(x)$ is chosen. For a linear model, the hypothesis set is all functions of the form $h(x) = \mathbf{w}^T\mathbf{x}$.*

Question B: What is a overfitting?

Solution B: *Overfitting is when a model has a much lower in sample error than tested out of sample error. This usually happens when the model is fitted too closely to the in sample error, and loses robustness.*

Question C: What is the most common way to prevent overfitting?

Solution C: *The most common way to prevent overfitting is regularization, which adds a penalty for the complexity of the model. Complex models are more likely to overfit.*

Question D: What is the difference between training data and validation data and test data?

Solution D: *Test data is what we use to finally evaluate our training. Training data is the data used to generate models to predict the test data. Some training data is set aside as validation data to proxy as out of sample data points. It is important to note that all validation data is considered training data.*

Question E: What is the fundamental training data sampling assumption that permits generalization?

Solution E: *The fundemental assumption is that the training data is sampled from the same distribution as the target data, and each data point is sampled independently from the rest. Half credit is awarded if the student does not mention that the samples are assumed to be independently sampled.*

Question F: Consider the machine learning problem of trying to decide whether or not an email is spam. What could X, the input space, be? What should Y, the output space, be? Finally, what is the meaning of the target function $f : X \to Y$ in the context of this problem?

Solution F: *A potential input space may simply be the bag of words representation of the emails. The output space is a binary decision of spam or not spam. The target function is a mapping between bag of wards data and whether or not it is considered spam or not.*

2 Bias Variance Tradeoff

In this problem, we will explore Bias Variance tradeoff by looking at bias and variance in two different linear models. You may try this simulation in any language you prefer, although either Matlab or Python are recommended. Note that we are using squared error as our loss measure.

Suppose we want to learn the function f:

$$y = \sin(\pi x)$$

over the domain $x = [-1, 1]$. However, suppose we only have access to two data points (x_1, y_1) and (x_2, y_2) randomly selected from the domain at a time. We call the ith set of data points $\mathcal{D}^{(i)}$. We hypothesize the following two linear hypothesis classes:

$g(x) = b$: the constant between the two points. Note that to get a hypothesis $g^{(i)}(x)$ from pair $\mathcal{D}^{(i)}$ we calculate $b = \frac{y_1+y_2}{2}$

$h(x) = ax + b$: the line that passes through (x_1, y_1) and (x_2, y_2). Remember that the slope of a line passing through two points is given by $\frac{y_2-y_1}{x_2-x_1}$ and that the intercept may be found by solving $y_1 = ax_1 + b$ for b.

Question A: Generate 10,000 samples of pairs of x_1 and x_2 by selecting independently from the uniform distribution from -1 to 1. Calculate their corresponding $y_1 = f(x_1)$ and $y_2 = f(x_2)$ values. We shall refer to the set of all generated pairs as $\mathcal{D}$, and the expectation value $\mathbb{E}_{\mathcal{D}}(x)$ refers to the expected value of x over all pairs in $\mathcal{D}$. Calculate and report $\mathbb{E}_{\mathcal{D}}(y_1)$ and $\mathbb{E}_{\mathcal{D}}(y_2)$ from your generated values. Are these values consistent with what they should be theoretically?

Solution A: *For a complete coded solution, see the iPython notebook posted onto Moodle.*
The expectation values should both be around 0, since sine is odd about the origin.

Question B: Calculate the parameters for the hypotheses $g^{(i)}(x)$ and $h^{(i)}(x)$ from each pair $\mathcal{D}^{(i)}$. Use these hypothesis functions to calculate the E_{out} for each data set. Report the expected E_{out} over $\mathcal{D}$ for the two hypotheses:

$$\mathbb{E}_{\mathcal{D}^{(i)}\sim\mathcal{D}}\left[E_{\text{out}}(g^{(i)})\right] \quad \text{and} \quad \mathbb{E}_{\mathcal{D}^{(i)}\sim\mathcal{D}}\left[E_{\text{out}}(h^{(i)})\right]. \tag{1}$$

Note that:

$$\mathbb{E}_{D^{(i)}\sim D}\left[\cdot\right] = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}^{(i)}}\left[\cdot\right],$$

$$E_{\text{out}}\left[v\right] = \mathbb{E}_x\left[\left(v(x) - f(x)\right)^2\right]$$

for some hypothesis function v

Hint: *An expectation value $\mathbb{E}_x$ means the average value over the entire domain. This may be found analytically or numerically. For a continuous function $f : [a, b] \to \mathbb{R}$, we can calculate $\mathbb{E}_x[f(x)]$ over the domain $[a, b]$ with*

$$\frac{1}{(b-a)}\int_a^b f(x)dx$$

You may also calculate this numerically with a Riemann sum or a Monte Carlo simulation.

Solution B: *$E_{out}(g)$ is about 0.75. $E_{out}(h)$ should be about 1.9*

Question C: Use these parameters to find the average functions $\mathbb{E}_{\mathcal{D}}[g_i(x)] = \bar{g}(x)$ and $\mathbb{E}_{\mathcal{D}}[h_i(x)] = \bar{h}(x)$. Note that we can estimate the average function for any x by taking the average over all final hypothesis, i.e.

$$\bar{g}(x) = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}} g^{(i)}(x)$$

What functions do you get for $\bar{g}$ and $\bar{h}$?

Solution C:

$$\bar{g} \approx 0$$
$$\bar{h} \approx .79x + 0$$

Question D: Now we shall calculate the expected bias and variance for each hypothesis class. The bias measures how much the average function that we would learn using different data sets $\mathcal{D}$ deviates from the target function that generated these sets. Thus, expected bias can be calculated by

$$\mathbb{E}_x\left[\text{Bias}(x)\right] = \mathbb{E}_x\left[(\bar{g}(x) - f(x))^2\right]$$

The variance measure the amount of variance in the final hypotheses depending on what data set you choose. Thus, variance can be written as

$$\mathbb{E}_x\left[\text{Var}(x)\right] = \mathbb{E}_x\left[\mathbb{E}_{\mathcal{D}}\left[(g^{(i)}(x) - \bar{g}(x))^2\right]\right]$$

Calculate the estimated bias and variance for hypothesis classes g and h. Show that the estimated E_{out} is equal to the sum of the bias and variance.

Solution D: *For hypothesis class g:*

$$\text{Bias}(g) = 0.5, \text{Var}(g) = 0.25, \text{Bias} + \text{Var} = E_{out} = 0.75$$

For hypothesis class h:

$$\text{Bias}(h) = 0.21, \text{Var}(h) = 1.68, \text{Bias} + \text{Var} = E_{out} = 1.89$$

Question E: (extra credit) Show the derivation of the bias variance decomposition. I.e., show that

$$\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(i)})\right] = \mathbb{E}_x[\text{Bias}(x) + \text{Var}(x)]$$

Hint: *Use the property*

$$\mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_x[f(\mathbf{x})]\right] = \mathbb{E}_x\left[\mathbb{E}_{\mathcal{D}}[f(\mathbf{x})]\right]$$

Solution E:

$$\begin{aligned}
\mathbb{E}_x\left[\text{Bias}(x) + \text{Var}(x)\right] &= \mathbb{E}_x\left[(\bar{g}(x) - f(x))^2 + \mathbb{E}_{\mathcal{D}}\left[(g^{(i)}(x) - \bar{g}(x))^2\right]\right] \\
&= \mathbb{E}_x\left[\mathbb{E}_{\mathcal{D}}\left[\bar{g}(x)^2 - 2g^{(i)}(x)f(x) + f(x)^2\right] + \mathbb{E}_{\mathcal{D}}\left[g^{(i)}(x)^2 + \bar{g}(x)^2 - 2g^{i}(x)\bar{g}(x)\right]\right] \\
&= \mathbb{E}_x\left[\mathbb{E}_{\mathcal{D}}\left[g^{(i)}(x)^2 - 2g^{(i)}f(x) + f(x)^2\right] + \bar{g}(x)^2 - 2\bar{g}(x)^2 + \bar{g}(x)^2\right] \\
&= \mathbb{E}_x\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(i)}(x) - f(x)\right)^2\right]\right] \\
&= \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_x\left[\left(g^{(i)}(x) - f(x)\right)^2\right]\right] \\
&= \mathbb{E}_{\mathcal{D}}\left[E_{out}(g^{(\mathcal{D})})\right]
\end{aligned}$$

3 The Perceptron

The perceptron is perhaps one of the most simple models of machine learning. Assume we are learning a linear model with d features. Our feature vector for each data point is $[x_1, x_2, \cdots, x_d]$. A perceptron computes the sum of a bias value b and each x_i multiplied with weight w_i. The perceptron classifies a data point based on the sign of the resulting sum.The formula can be written more compactly as

$$h(x) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + b\right)$$

A matrix representation of such with input vector $\mathbf{x}$ and weights $\mathbf{w}$ can be written as

$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$$

A graphical way of understanding a perceptron is that it is a hyperplane through $\mathbb{R}^d$ such that all points on one side of the plane belong to one class. For example, in a 2-D dataset, a perceptron would simply be the line that divides all points that classify as + from all points that classify as -.

Question A: It is not entirely obvious how the matrix representation of the perceptron can include the bias term b. If our input vector $\mathbf{x}$ is defined as the vector $[x_1, x_2, \cdots, x_d]$ and our weight vector $\mathbf{w}$ is defined similarily as $[w_1, w_2, \cdots, w_d]$, the matrix form $h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$ would be missing the bias term. How then, should we define $\mathbf{x}$ and $\mathbf{w}$ such that our perceptron can include the bias term?

Hint: *Include an additional element in* $\mathbf{w}$ *and* $\mathbf{x}$

Solution A: *We can simply add a 1 to the* $\mathbf{x}$ *vector and include the bias term* b *in the weight vector* $\mathbf{w}$*. Thus we write*

$$\mathbf{w} = [b, w_1, w_2, \cdots, w_d], \mathbf{x} = [1, x_1, x_2, \cdots, x_d]$$

Note that evaluating $\mathbf{w}^T\mathbf{x} = b + w_1x_1 + w_2x_2 + \cdots + w_dx_d$ *which is exactly how the perceptron works.*

Question B: The PLA (or the Perceptron Learning Algorithm) is a simple method of training a perceptron. First, an initial guess is made for the weight vector $\mathbf{w}$. Then, one misclassified point is chosen at random and the $\mathbf{w}$ is updated by

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y(t)\mathbf{x}(t)$$

Where $\mathbf{y}(t)$ and $\mathbf{x}(t)$ correspond to the misclassified point selected at the tth iteration. This process continues until all points are classified correctly.

The graph below shows an example 2D dataset. The circles represent positive values whereas the Xs represents negative values. In a 2D dataset, the perceptron is simply a line $w_1x_1 + w_2x_2 + b$. Perform the PLA algorithm manually on the data shown on the graph below, with an initial guess $w_1 = b = 0, w_2 = 1$. Write down the weights of each step as well as which point you are updating with.

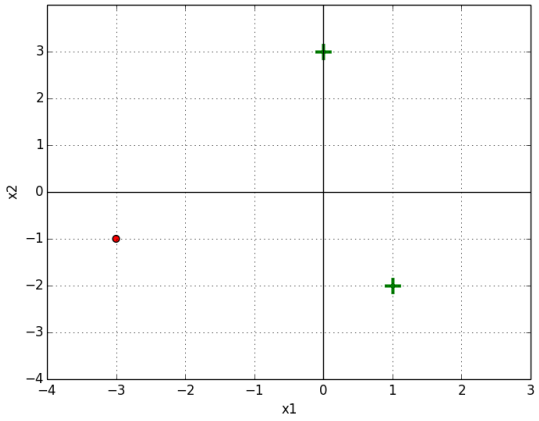

Figure 1: The green + are positive and the red dot is negative

Give your solution in the form a table showing the weights at each timestep and the $([x_1, x_2], y)$ value of the misclassified point that is chosen to update for the next iteration. You can iterate through the three points in any order.

t	b	w_1	w_2	x_1	x_2	y
0	0	0	1	1	-2	+1
⋮	⋮	⋮	⋮	⋮	⋮	

Solution B:

Table 1: This table shows what the **w** vector is for each iteration and which **x** vector is chosen to update the next iteration

t	b	w_1	w_2	x_1	x_2	y
0	0	0	1	1	-2	+1
1	1	1	-1	0	3	+1
2	2	1	2	1	-2	+1
3	3	2	0			

Question C: A dataset $S = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ is *linearly separable* if there exists a perceptron that correctly classifies all data points in the set. In otherways, there exists a hyperplane that divides positive data points and negative data points. We call a dataset n-dimensional if it spans $\mathbb{R}^n$. (Recall, $\text{span}(S) = \left\{ \sum_{i=1}^{k} \lambda_i v_i \middle| k \in \mathbb{N}, v_i \in S_{\mathbf{x}}, \lambda_i \in \mathbb{R} \right\}$. where $S_{\mathbf{x}}$ is the set of data points without classifications). In particular,

a set of colinear points is one-dimensional. In a 2D data set, how many data points are in the smallest data set that not linearly separable? How about for a n-dimensional set?

Solution C: *4 data points is the minimum number of points required to make dataset not linearly separable in 2 dimensions.* $n+2$ *is the minimum number of data points to make dataset not linearly separable in* n *dimensions.*

Question D: Assume a dataset is *not* linearly separable. Will the PLA algorithm ever converge? Why or why not?

Solution D: *No. The PLA algorithm continues until no data points are misclassified. Since there exists no hyperplane that can divide the points perfectly, the PLA algorithm will never converge.*

4 Gradient Descent

Stochastic gradient descent (SGD) is an important optimization method in machine learning. SGD seeks to minimize an objective function by gradually updating a weight vector $\mathbf{w}$. SGD is critical to Machine learning, as it is used everywhere from logistic regression to training neural networks. In this question, you will be asked to first implement SGD for linear regression using the squared loss function. Then, you will analyze how several parameters affect the learning process.

Linear Regression is a method to obtain a linear model to demonstrate a linear relationships between variables. Mathematically expressed, the linear model is

$$h(x) = \mathbf{w}^T \mathbf{x}$$

Linear regression chooses a linear model by minimizing the squared loss function, which is the sum of the squares of the differences between actual and predicted output values. The squared loss function can be summarized as

$$\mathscr{L}(\{(x_i, y_i)\}_{i=1}^N, w) = \sum_{i=1}^{n} (y_i - h(x_i))^2$$

For linear regression, the loss function is thus described as

$$\mathscr{L} = \sum_{i=1}^{n} (y_i - \mathbf{w}^T x_i)^2$$

Question A: SGD uses the gradient of the loss function to make incremental adjustments to the weight vector $\mathbf{w}$. Derive the gradient of the squared loss function for linear regression.

Solution A: *The gradient is* $-2x_i(y_i - \mathbf{w}^T x_i)$.

Implementation

The following questions uses the dataset hw1ds1.txt. The file has a header denoting which columns correspond to what values.

Question B: Implement SGD with a squared loss function in order to find the plane that best fits the data. What are your final weights? Use the following constants:

$$\epsilon = .0001, \eta = e^{-15}, \mathbf{w} = [.001, .001, .001, .001], \text{epoch} = 1000$$

- η is the step size.
- $\mathbf{w}$ is the initial weight vector.
- After each epoch (1000 iterations), measure the loss.
- Keep track of the loss reduction from epoch to epoch.
- Stop when the relative loss reduction compared to the first epoch is less than ϵ. That is, let $\Delta_{0,1}$ denote the loss reduction from the initial model to end of the first epoch. Then stop after epoch t if $\Delta_{t-1,t}/\Delta_{0,1} \leq \epsilon$.

Solution B: *See the ipython notebook for a coded solution The weights should be around* $[-6, 4, -12, 9]$. *Bias should be around 0.*

Question C: Now, vary the learning rate and calculate the error at each epoch. Then plot all of them over each other and explain what is happening. Use these values of η:

$$[e^{-12}, e^{-13}, e^{-14}, e^{-15}, e^{-16}, e^{-17}, e^{-18}]$$

Solution C: *When the learning rate is too slow, SGD takes a very long time to converge and the error decreases very slowly. When the learning rate is too high, SGD overcorrects and oscillates around the optimal solution, potentially never convering.*

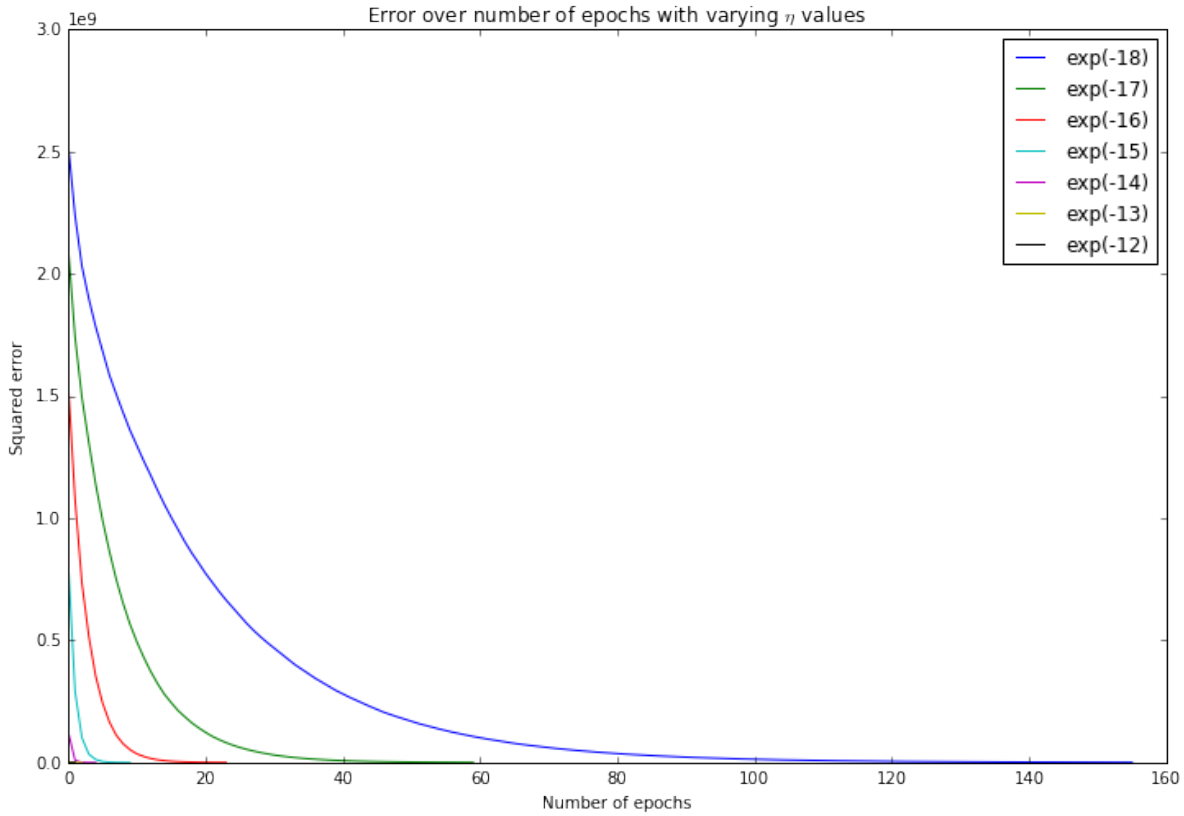

Figure 2: The total squared error is not important here. What matters is that the students got the right shape.

Question D: Below is the closed form solution for linear regression with least squares. Does the result from this analytical solution match up with what you got from SGD?

$$\mathbf{w} = (\mathbf{x}^T\mathbf{x})^{-1}(\mathbf{x}^T\mathbf{y})$$

Solution D: *They should be reasonably close. The TAs got*

$$w_1 = -5.9994, w_2 = 3.9997, w_3 = -12.0012, w_4 = 9.0004, b = 0.0181$$

using linear regression

The last few questions are conceptual and require 1 - 2 sentence answers.

Question E: Is there any reason to use SGD when a closed form solution exists?

Solution E: *Yes. In many cases, calculating the closed form solution is computationally intractable. In these cases, SGD can be used to get an arbitrarily good approximation.*

Question F: What happens if SGD is used on a non-convex learning problem?

Solution F: *SGD will be able to converge on a solution, but this solution will not necessarily be the optimal solution. This is because SGD finds local minima, not global minima.*