Jessica Wang
Machine Learning & Data Mining
Set 1

I. Basics

Problem A.

A hypothesis set is the set of all possible hypotheses generated by a learning algorithm for modeling an unknown target function. The learning algorithm produces a hypothesis believed to be similar to the target function and the hypothesis set is the space of all such hypotheses which the learning algorithm may propose or output.

Problem B.

The hypothesis set of a linear model is given by all possible hypotheses of the form

$$f(x|w,b) = w^T x - b$$

Problem C.

Overfitting describes a modeling error which occurs when a machine learning algorithm produces a hypothesis which captures the random error or noise of the data. The model is fitted to the training data too well which negatively impacts the model's ability to generalize to new, unseen, non-training data which allows for a much greater testing error than training error.

Problem D.

One way to prevent overfitting is to use cross-validation which allows the model to be tested against an "unseen" subset of the data. Increasing the training data size if possible will also help prevent overfitting. Another way to prevent overfitting is simplify the model used through regularization which will reduce the variance and expected error, allowing the model to better generalize to new, unseen data.

Problem E.

Training data refers to the data used to build a model and train an algorithm while test data refers to the data used to assess how well the model built performs. A model should never be changed based on information from test data because modifying a model simply based on test data is to torture the data and let the results determine the hypothesis rather than letting the observations determine a potential hypothesis.

Problem F.

The two assumptions we make about how our dataset is sampled is that the data is gathered independently and identically.

Problem G.

For the machine learning problem of deciding whether or not an email is spam, the input space X could be a string represented as a 'bag of words' feature vector where the presence of a particular word is indicated by a value of one and the absence of the word indicated by zero. The output space Y could be {-1, +1} where a value of one indicates the email should be classified as spam and a value of negative one indicates the email should not be classified as spam.

Jessica Wang
Machine Learning & Data Mining
Set 1

Problem H.

The k-fold cross-validation procedure is as follows: the original sample data is randomly partitioned into k subsets of data, k – 1 of these subsets are used as training data and the single other subset of data is set aside as validation data for testing the model, and this is repeated so that each of the k subsets of data may be used once as validation data while the rest is considered training data. The k-fold cross-validation procedure is used to build a model which allows for a maximal number of data subsets to be used as training data while still retaining data that may be used for validation.

II. Bias-Variance Tradeoff

Problem A.

For a model f_s trained on a dataset S to predict a target $y(x)$ for each x, we would like to show that

$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x[Bias(x) + Var(x)]$$

given the following definitions:

$$F(x) = \mathbb{E}_s[f_S(x)]$$
$$E_{out}(f_S) = \mathbb{E}_x\left[\left(f_S(x) - y(x)\right)^2\right]$$
$$Bias(x) = \left(F(x) - y(x)\right)^2$$
$$Var(x) = \mathbb{E}_s[(f_S(x) - F(x))^2]$$

We begin by substituting our definition of $E_{out}(f_S)$ into the right hand side of the equation we would like to prove,

$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_s\left[\mathbb{E}_x\left[\left(f_S(x) - y(x)\right)^2\right]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S\left[\left(f_S(x) - y(x)\right)^2\right]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S\left[\left(F(x) + f_S(x) - y(x) - F(x)\right)^2\right]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S\left[\left(\left(F(x) - y(x)\right) + \left(f_S(x) - F(x)\right)\right)^2\right]\right]$$

If we let $F(x) - y(x)$ be denoted by a and $f_S(x) - F(x)$ be denoted by b,

$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S[(a+b)^2]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S[a^2 + 2ab + b^2]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S[a^2] + \mathbb{E}_S[b^2] + \mathbb{E}_S[2ab]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[\mathbb{E}_S[(F(x) - y(x))^2] + \mathbb{E}_S[(f_S(x) - F(x))^2] + \mathbb{E}_S[2ab]\right]$$
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[[(F(x) - y(x))^2] + \mathbb{E}_S[(f_S(x) - F(x))^2] + \mathbb{E}_S[2ab]\right]$$

We can substitute terms in through our given definitions,

$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x\left[Bias(x) + Var(x) + \mathbb{E}_S[2ab]\right]$$

Now, we just need to show that $\mathbb{E}_S[2ab]$ or $\mathbb{E}_S[2(F(x) - y(x))(f_S(x) - F(x))]$ is zero,

$$\mathbb{E}_S[2(F(x) - y(x))(f_S(x) - F(x))] = 2(F(x) - y(x))\mathbb{E}_S[(f_S(x) - F(x))]]$$

However, we know that $\mathbb{E}_s[(f_S(x) - F(x))] = 0$ because $F(x) = \mathbb{E}_s[f_S(x)]$ therefore,

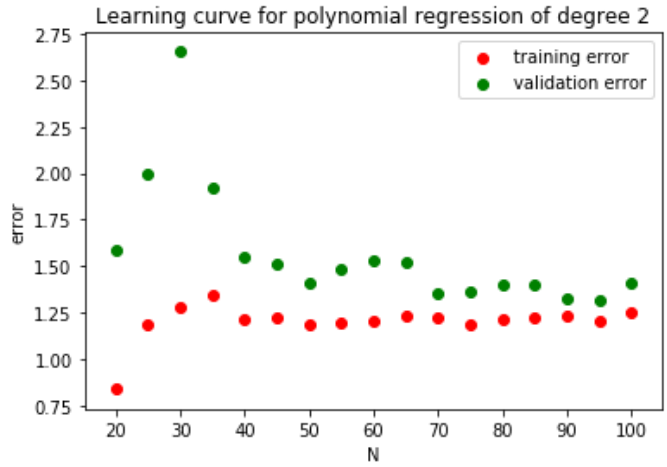
$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x[Bias(x) + Var(x) + 0]$$

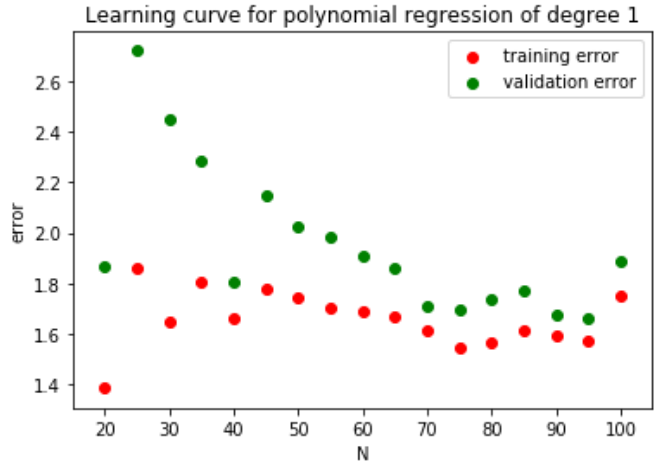
Jessica Wang
Machine Learning & Data Mining
Set 1

$$\mathbb{E}_s[E_{out}(f_S)] = \mathbb{E}_x[Bias(x) + Var(x)]$$

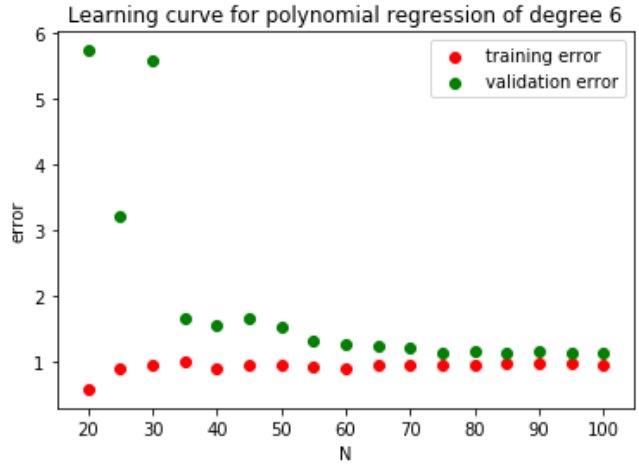
Problem B.

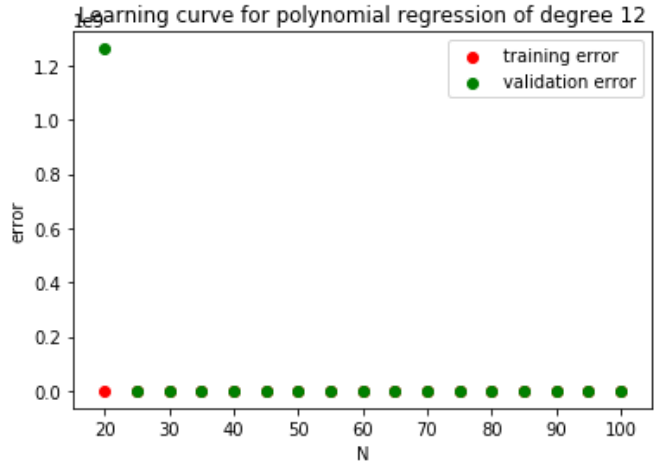
See Jupyter notebook for code.





Jessica Wang
Machine Learning & Data Mining
Set 1





Problem C.

Based on the learning curves, the polynomial regression model of degree 1 has the highest bias. This is because the model is too simple and even with no variance, the model class has high training error. Therefore, since it has the highest training error, this model has the highest bias.

Problem D.

Based on the learning curves, the polynomial regression model of degree 12 has the highest variance. We see this on the plot because the gap between the training error and validation error is the largest among all the models. Therefore, it severely overfits the data and it is too complex and is subject to instability when generalizing to new, unseen data.

Problem E.

The learning curve of the quadratic model tells me that the model would not significantly improve if we had additional training points because the training error and validation error appear to have stabilized and converged rather nicely already as displayed by the learning curve.

Jessica Wang
Machine Learning & Data Mining
Set 1

Problem F.

Training error is generally lower than validation error because the algorithm trains on the training set and validates or assesses the model on the test set and since the training set was used to build the model, this data has already been seen by the algorithm whereas the algorithm has not seen the test data and thus, training error is generally lower than validation error since the validation data is independent of the training process.

Problem G.

Based on the learning curves, I would expect the model of degree 6 to perform the best on some unseen data drawn from the same distribution as the training data since the validation error is the smallest for large values of N and most optimal bias-variance tradeoff.

III. The Perceptron

Problem A.

See Jupyter notebook for code. Printed output is shown below:

```
t = 0, w = [ 0.  1.], b = 0.0, [x1, x2] = [ 1 -2], y = 1
t = 1, w = [ 1. -1.], b = 1.0, [x1, x2] = [0 3], y = 1
t = 2, w = [ 1.  2.], b = 2.0, [x1, x2] = [ 1 -2], y = 1
final w = [ 2.  0.], final b = 3.0
```

Problem B.

In a 2D dataset, there are 4 data points in the smallest dataset that is not linearly separable, such that no three points are collinear. In this dataset, if we imagine the four points to be classifiable as two points labeled ‘+’ and two points labeled ‘-‘, arranging the four points in a “square” shape where opposite corners are the same classification, as shown below, means the dataset is not linearly separable.

In a 3D dataset, there are 5 data points in the smallest dataset that is not linearly separable, such that no four points are coplanar. In this dataset, if we imagine the five points to be classifiable as three points labeled ‘+’ and two points labeled ‘-‘, if we arrange the three points labeled ‘+’ on a single plane and placed one of the points labeled ‘-‘ on one side of the plane, and the other point labeled ‘-‘ on the other side, the dataset would not be linearly separable.

To generalize for an N-dimensional set, in which no < N-dimensional hyperplane contains a non-linearly-separable subset, there are N+2 data points in the smallest dataset that is not linearly separable.

Jessica Wang
Machine Learning & Data Mining
Set 1

Problem C.

If a dataset is not linearly separable, the perceptron learning algorithm will never converge. This is because there would never exist a hyperplane such that the perceptron learning algorithm can classify all the points correctly and terminate.

IV. Stochastic Gradient Descent

Problem A.

To define *x* and *w* such that the model includes the bias term, we should include an additional element in *w* and *x* as $x = (1, x_1, x_2, \ldots x_d)$ and $w = (w_0, w_1, w_2, \ldots w_d)$ where w_0 is the bias term.

Problem B.

The square loss function *L* is given by:

$$L(f) = \sum_{i=1}^{N} (y_i - w^T x_i)^2$$

To derive the gradient of the squared loss function with respect to w for linear regression,

$$\delta_w L(w,b) = \delta_w \sum_{i=1}^{N} L(y_i, f(x_i|w,b))$$

$$= \sum_{i=1}^{N} \delta_w L(y_i, f(x_i|w,b))$$

$$= \sum_{i=1}^{N} -2\big(y_i - f(x_i|w,b)\big)\delta_w f(x_i|w,b)$$

$$= \sum_{i=1}^{N} -2\big(y_i - f(x_i|w,b)\big)x_i$$

Problem C.

See Jupyter notebook for code.

The SGD function outputs a vector with the loss after each epoch and a matrix of the weights after each epoch with one row per epoch.

Problem D.

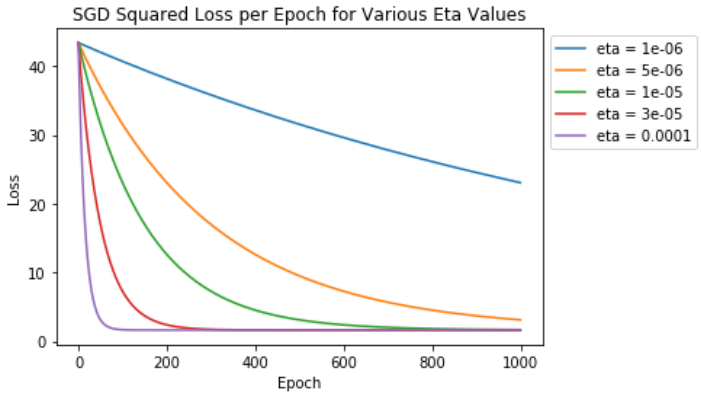
See Jupyter notebook for code and animations.

As the starting point varies, the convergence behavior or SGD changes such that for starting points further away from convergence, the time for which it takes the SGD to reach convergence increases. For some datasets, the time of convergence is similar whereas for other datasets, partially due to the step size, the time for convergence is longer. When the step size increases, the time it takes to converge shortens.

Jessica Wang
Machine Learning & Data Mining
Set 1

Problem E.

See Jupyter notebook for code.



As the learning rate increases, the SGD converges sooner and faster.

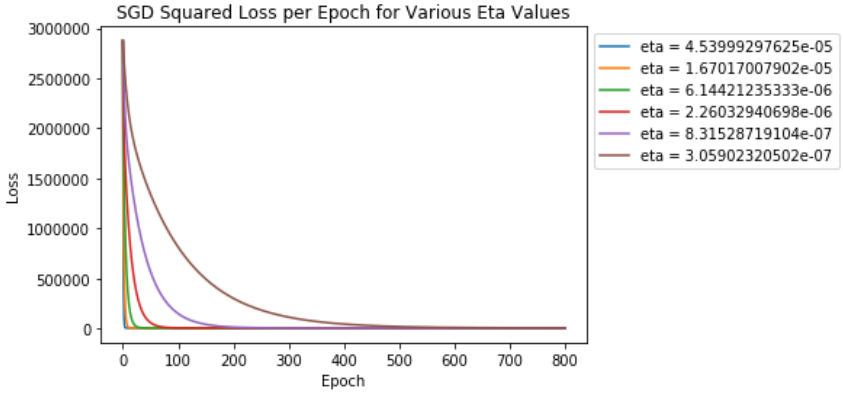
Problem F.

See Jupyter notebook for code. Printed output of final weights is shown below (bias is first term):

```
[ -0.22720591  -5.94229011   3.94369494 -11.72402388   8.78549375]
```

Problem G.

See Jupyter notebook for code.



As the learning rate increases, the training error decreases at a faster rate allowing for convergence to occur sooner.

Problem H.

See Jupyter notebook for code. Printed output of final weights is shown below (bias is first term and this result is a very near match for the printed output of final weights from SGD):

```
[ -0.31644251  -5.99157048   4.01509955 -11.93325972   8.99061096]
```

Jessica Wang
Machine Learning & Data Mining
Set 1

Problem I.

It may be more computationally efficient to use SGD even though a closed form solution exists when the data is very large since performing matrix multiplication on such large sets of data may be both time and space inefficient.

Problem J.

Based on the SGD convergence plots generated, a stopping condition that is more sophisticated than a pre-defined number of epochs is to stop when the progress is sufficiently small (ie. relative reduction is less than 0.001)

Problem K.

The convergence behavior of the weight vector differs between the perceptron and SGD algorithms in that the SGD provides a smoother path to the global minimum because it moves directionally based on gradient using the step size whereas the convergence behavior of the weight vector to the perceptron is “jumpier” since it could overshoot in either direction with each iteration or update.