CS57800 Statistical Machine Learning

Homework 2

Andres Bejarano
Department of Computer Science
abejara@purdue.edu

October 7, 2015

1 Foundations

1.1

1. Boolean function:

$$f(x_1, x_2, x_3, x_4) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4) \vee (x_2 \wedge x_3) \vee (x_2 \wedge x_4) \vee (x_3 \wedge x_4)$$

2. Linear function:

$$f(x_1, x_2, x_3, x_4) = sgn(x_1 + x_2 + x_3 + x_4 - 1.5)$$

1.2

The size of CON_B is 2^n. Since there are two options (to appear or not) for each term, then there are 2 cases and n terms.

1.3

$$\begin{aligned}
\|\beta_n - \beta^*\|^2 &= \|\beta_n\|^2 - 2\beta_n\beta^* + \|\beta^*\|^2 \\
&= \|\beta_o + y_i x_i\|^2 - 2\beta^*(\beta_o + y_i x_i) + \|\beta^*\|^2 \\
&= \|\beta_o\|^2 + 2\beta_o y_i x_i + \|y_i x_i\|^2 - 2\beta^*\beta_o - 2\beta^* y_i x_i + \|\beta^*\|^2 \\
&\leq \|\beta_o\|^2 - 2\beta_o\beta^* + \|\beta^*\|^2 + 1 - 2(1) \\
&\leq \|\beta_o - \beta^*\|^2 - 1
\end{aligned}$$

From step 3 to step 4 the following considerations are applied: (1) By the concept of separability there exist a separating hyperplane such that the constraint $\beta^* y_i u_i^* \geq 1$ is satisfied (Ripley, 1996). Then it is followed that $\beta^* y_i x_i \geq 1$. (2) $\|y_i x_i\| \geq 1$ when at least one feature was classified by the hyperplane during an iteration. (3) Classification for β_o is obtained after the first iteration (assuming $\beta_o = 0$), then $\beta_o y_i x_i = 0$.

1.4

1. Initialize h to $x_1 \wedge \neg x_1 \wedge x_2 \wedge \neg x_2 \wedge ... \wedge x_n \wedge \neg x_n$
2. For each positive sample (final result is $h_i = 1$) remove all not satisfied expressions

In the worst case the algorithm does at most $n+1$ mistakes (it needs to check at least half plus one of the literals). Since there are $2n$ literals in h, each one of them must be checked. The the mistake bound for the algorithm is $n+1$.

1.5

1. Yes, both classifiers will converge. However, since the order to the data set affects the result they will return different solutions.
2. Since solutions converge then data is linearly separable. Therefore both classifiers work correctly learning from data. Hence, their training error will be 0.

1.6

When an update occurs, $y_i x_i$ is equivalent to $w_{i+1} - w_i$ where i is the index of the entry where the update occurred. Then we have:

$$\|\sum_{i \in N} y_i x_i\| = \|\sum_{i \in N} (w_{i+1} - w_i)\|$$

A sequence of type $\sum_{n=1}^{N} a_n - a_{n-1} = a_N - a_0$, then $\sum_{i \in N} w_{i+1} - w_i = w_{N+1} - w_0$. Assuming $w_0 = 0$ we have:

$$\|\sum_{i \in N} y_i x_i\| = \|w_{N+1}\|$$

Applying the same approach for $\|w_{N+1}\|$ as the resultant of sequence $\sum_{n=1}^{N} a_n - a_{n-1} = a_N - a_0$, then the above expression can be expressed as:

$$\|\sum_{i \in N} y_i x_i\| = \sqrt{\sum_{i \in N} \|w_{i+1}\|^2 - \|w_i\|^2}$$

Similarly as the first step, w_{i+1} is the resultant after an updating w_i, then $w_{i+1} = w_i + y_i x_i$. The expression is now:

$$\|\sum_{i \in N} y_i x_i\| = \sqrt{\sum_{i \in N} \|w_i + y_i x_i\|^2 - \|w_i\|^2}$$

Expanding the first component of the summation we got:

$$\|\sum_{i\in N} y_i x_i\| = \sqrt{\sum_{i\in N} \|w_i\|^2 + 2y_i x_i w_i + \|y_i x_i\|^2 - \|w_i\|^2}$$

$$\|\sum_{i\in N} y_i x_i\| = \sqrt{\sum_{i\in N} 2y_i x_i w_i + \|y_i x_i\|^2}$$

Since $2y_i x_i w_i \leq 0$ and $y_i x_i \leq x_i$ the above expression can be written as:

$$\|\sum_{i\in N} y_i x_i\| \leq \sqrt{\sum_{i\in N} \|x_i\|^2}$$ ■

2 Programming Report

Both Perceptron and Winnow algorithms were implemented in Python. For each one of them a couple of tasks were performed before running them:

1. Sentences are cleaned by removing grammatical and grouping symbols.
2. A dictionary of neutral words is implemented in order to improve the classification mechanism. Such neutral words (in neutral.csv) are mostly English prepositions and auxiliary words. It is assumed that such words are useless for classification since they do not affect on the overall sentiment label of the sentence.

The default parameters are used for defining experiments with the data sets. If no parameter is indicated then Perceptron is selected, the maximum number of iterations is 10 and the feature set is unigrams.

The training process runs the number of indicated iterations. It can stop sooner if the number of errors during is 0.

2.1 Perceptron Results

This algorithm converged faster to a error-free weight vector for the training set. In Figures 1, 2, and 3 it is shown the convergence behaviour for each feature set.

2.1.1 Experiments: Max Iterations = 10

Unigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	0.975	0.983	0.992	0.987	0.987
Validating	0.518	0.712	0.739	0.726	0.726
Testing	0.510	0.701	0.733	0.717	0.717

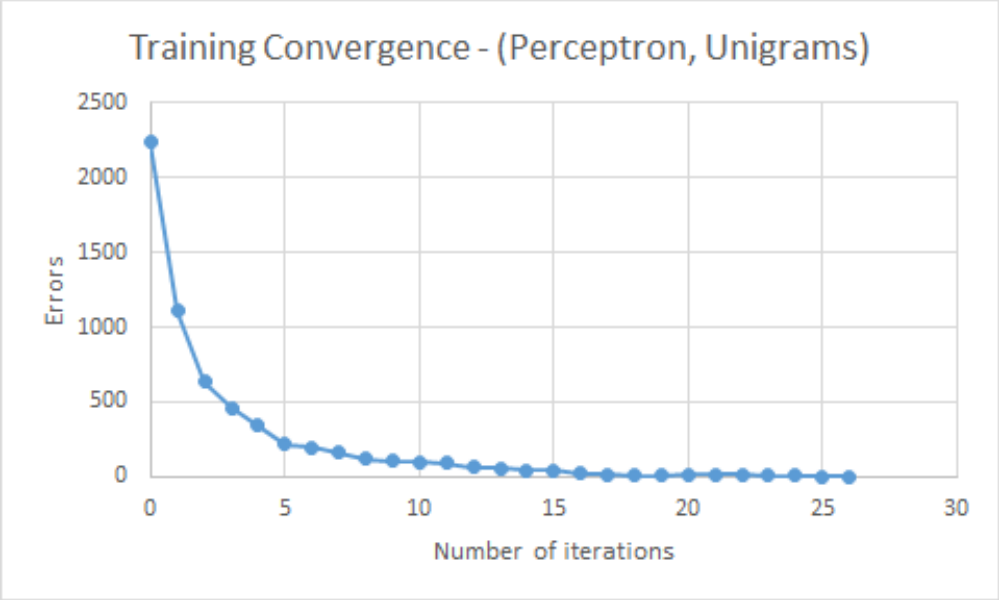

Figure 1: Training convergence for Perceptron using Unigrams set

Bigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	1.000	1.000	0.999	0.999	0.999
Validating	0.294	0.631	0.377	0.504	0.472
Testing	0.348	0.665	0.447	0.556	0.534

Both Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	0.999	0.999	1.000	0.999	0.999
Validating	0.540	0.737	0.733	0.735	0.735
Testing	0.527	0.720	0.727	0.724	0.724

2.1.2 Experiments: Max Iterations = 20

Unigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	0.997	0.998	0.999	0.998	0.998
Validating	0.537	0.732	0.735	0.734	0.734
Testing	0.521	0.715	0.725	0.720	0.720

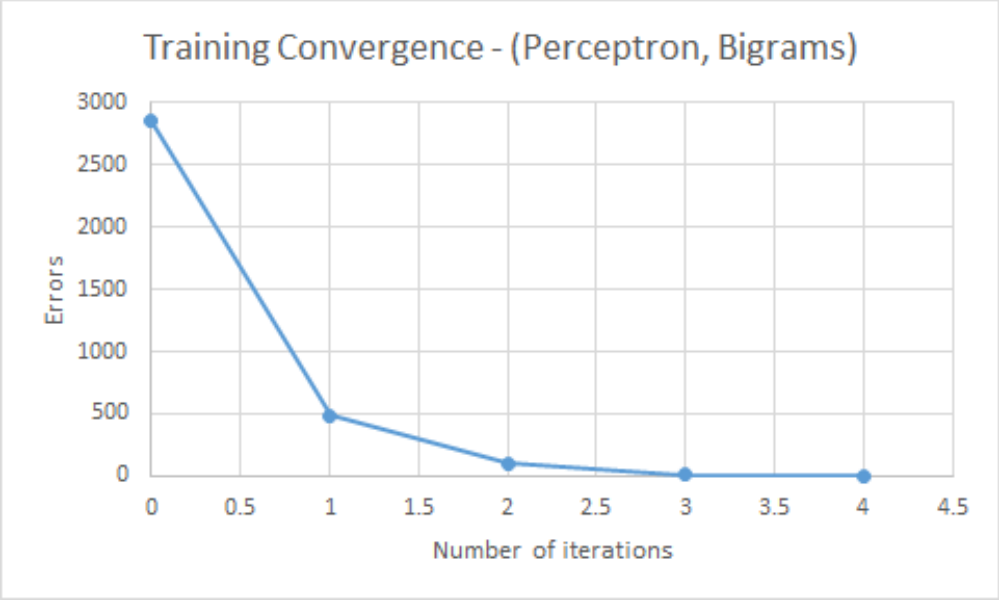

Figure 2: Training convergence for Perceptron using Bigrams set

Bigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	1.000	1.000	0.999	0.999	0.999
Validating	0.294	0.631	0.377	0.504	0.472
Testing	0.348	0.665	0.447	0.556	0.534

Both Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	1.000	1.000	1.000	1.000	1.000
Validating	0.540	0.740	0.727	0.733	0.733
Testing	0.523	0.721	0.716	0.719	0.719

2.1.3 Experiments: Max Iterations = 30

Unigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	1.000	1.000	1.000	1.000	1.000
Validating	0.527	0.717	0.750	0.734	0.733
Testing	0.518	0.704	0.747	0.725	0.725

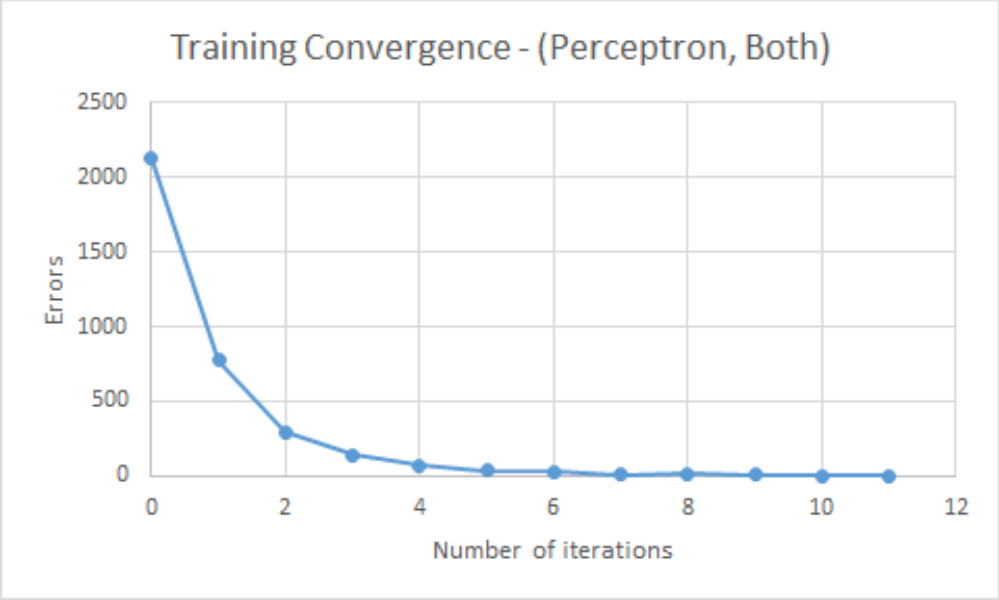

Figure 3: Training convergence for Perceptron using Both set

Bigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	1.000	1.000	0.999	0.999	0.999
Validating	0.294	0.631	0.377	0.504	0.472
Testing	0.348	0.665	0.447	0.556	0.534

Both Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	1.000	1.000	1.000	1.000	1.000
Validating	0.540	0.740	0.727	0.733	0.733
Testing	0.523	0.721	0.716	0.719	0.719

2.2 Winnow Results

The Winnow algorithm was implemented according to the specifications. However, after several tries and tunings, the algorithm didn't converged to an error-free weight vector. Therefore the metrics for this algorithm are not as good as Perceptron.

2.2.1 Experiments: Max Iterations = 10, 20, 30

For 10, 20 and 30 iterations no considerable changes were found. The Bigrams set got a slightly better performance than the other two sets.

Unigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	0.000	0.501	1.000	0.750	0.668
Validating	0.000	0.500	1.000	0.750	0.667
Testing	0.000	0.493	1.000	0.746	0.661

Bigrams Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	0.000	0.502	0.999	0.750	0.668
Validating	0.161	0.485	0.732	0.608	0.583
Testing	0.177	0.492	0.764	0.628	0.599

Both Metrics					
Set	*Accuracy*	*Precision*	*Recall*	*Average*	*F-Score*
Training	0.000	0.501	1.000	0.750	0.668
Validating	0.000	0.500	1.000	0.750	0.667
Testing	0.000	0.493	1.000	0.746	0.661