CS4248 Assignment 3

by Teekayu Klongtruajrok A0174348X

Architecture Overview

For this assignment, a text classifier object is constructed. The classification is performed by using a multi-class generalization of the perceptron algorithm. As elaborated in Figure 1, for a given input text vector $\overrightarrow{X} = \{x_1, \dots, x_k\} s.t. x_n \in \mathbb{R}$ for all words in a vocabulary, n number of perceptron models are trained for each class in $C = \{c_1, \dots, c_n\}$. The reason for this design is because a perceptron output only have two states: 1 and -1. In the context of this classification problem, output of 1 means that $\overrightarrow{X}$ belongs to the current class, and -1 if it's not. Therefore, a model for this text classifier is represented by $\mathbf{W} = \{W_1, \dots, W_n\} s.t. W_n = \{w_1, \dots, w_k\} s.t. w_n \in \mathbb{R} \forall x_k \in \overrightarrow{X}$.

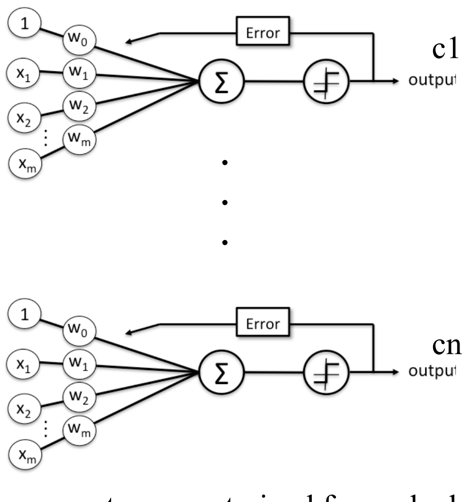

Figure 1: Separate perceptrons are trained for each class for a given input.

Prediction for a class is simply a max-pool of every class's affine transformation, as shown in Figure 2. This interpretation is valid because the weighted sum is the level of affinity between the input and the respective class. In the normal training process, the affine transformation is then split to class 1 or -1 by an activation function, with the higher value being assigned to 1 (which means the input belonged in this class). Therefore it makes sense to extend the idea to multiple classes, with the higher weighted sum meaning more affinity with the class.

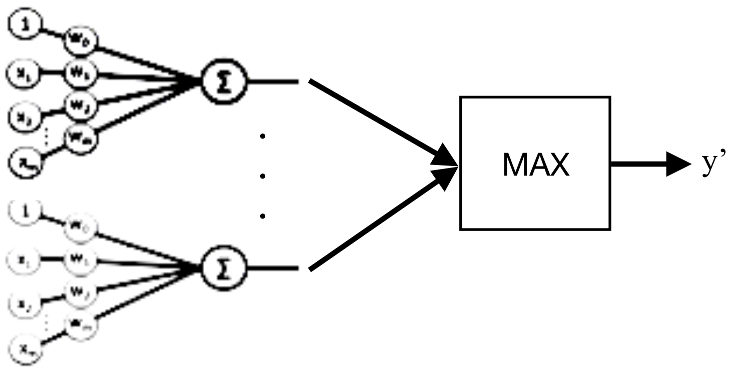

Figure 2: Predict a class by argmax of affine transformation for an input on each class's weights.

The weight update rule is described below:

$$w_i^{t+1} = w_i^t + (\eta * x_i * \hat{y}_j) \exists \hat{y}_j = \{-1,1\}, i \in [1,k], j \in [1,n]$$

The the updating coefficient is a product of the learning rate, the value of the input, and the class assignment status. Recall that perceptron having out 1 means it's a true-positive and -1 means its a false-positive. This means that the weight update is an addition if the class is correctly assigned and turned into subtraction if incorrectly assigned.

The input text vector is represented by a term frequency for each input, an example shown on Table 1. The value to be used for calculation is the term frequency. The raw input text is first split by spaces into a word list. Stopwords are removed, then the remaining words stemmed. The stemmed words are, then, counted and put into a hash table with structure similar to Table 1.

word	$word_1$	$\dots$	$word_k$
term frequency	4	$\dots$	1

Table 1: An example of input X representation.

$$\chi^2 = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})}$$

where
N_{00} is the number of training texts that do not contain w and are not in class c.
N_{01} is the number of training texts that do not contain w and are in class c.
N_{10} is the number of training texts that contain w and are not in class c.
N_{11} is the number of training texts that contain w and are in class c.

Figure 3: Formula for chi-squared correlation coefficient.

Because perceptrons does not guarantee convergence, or even the best convergence, efforts need to be put into pre-processing the data to increase the chance of convergence. One of the ways to do this is feature engineering. A value measuring how relevant the word is to the class is the correlation coefficient shown in Figure 3. Sometimes the denominator is 0 when there is a word that appears across all classes. When this is the case, this means the particular word is not unique, hence is useless in inference. In order to avoid divide-by-zero exceptions, 0 can simply be returned from the function calculating chi-square. A good criterion to use is to ignore any word than is less than 1.

Experiments

- Vary the learning rate
- Train with both the sigmoid and the step function
- Train with feature-selection turned on

Findings

From the experiments conducted, the learning rate that yield the best result is 1e-e. Step activation function performed best. Due to time constraint, feature-selection experiment was not performed because the function involves looping through the entire dataset for each unseen word to calculate chi-square, making it extremely computationally intensive.

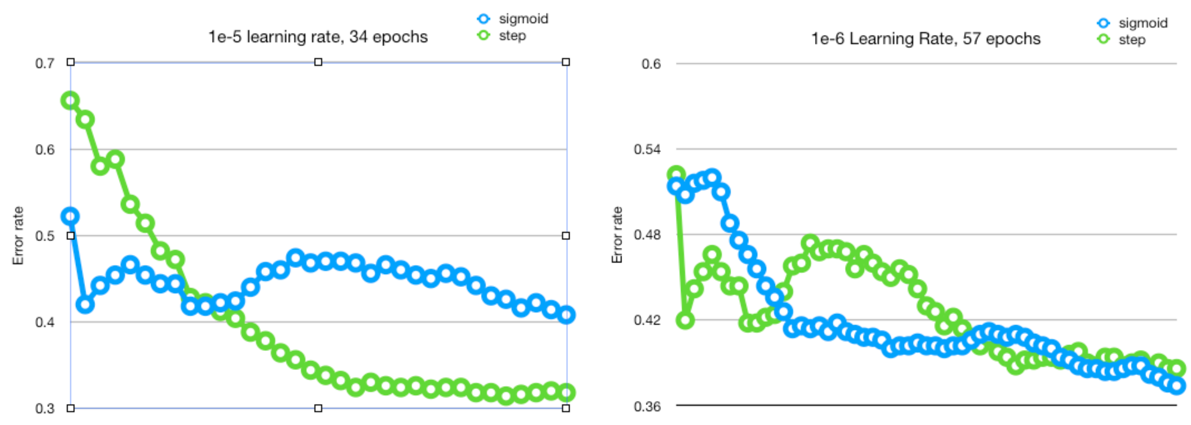

Figure 4: *(Left)* Step function outperforming sigmoid function with 1e-5 learning rate. *(Right)* Sigmoid function outperforming step function with 1e-6 learning rate. *(Both)* Step function with higher learning rate outperforms everything.

Comparison chart between different activation function and learning rates is shown in Figure 4 (these are the 2 best performing training instances). With smaller learning rate, the sigmoid function shows a much smoother convergence than the step function, meaning the error decrease is more consistent, therefore making it a better choice in general. Because the sigmoid function is is exponential in nature, it requires a much smaller learning rate than the step function, lest the program encounters floating point overflow exception, therefore an optimal learning rate for the sigmoid must be lower than the step. With a higher learning rate, the sigmoid performs much worse while the step function thrive. As a matter of fact, the result of the step activation with higher learning rate performs better than everything else with smaller number of epochs. This could either mean the the step function is the best activation function or the optimal learning rate for the sigmoid activation function had not been discovered yet.

The average of 5-fold cross-validation result for the step activation with 1e-5 learning rate is 32.36% error rate.

Future Improvements

In order to further refine the model, more experiments can be done with learning rate. From the experiments performed, the learning rate is responsible for the most changes in the model's performance. Different activation function also have different performance threshold for different learning rates. For example, the sigmoid function is an exponential, therefore the tolerance for numerical calculation is much lower than the step function (because when a weight becomes too large, a floating point overflow exception occurs), therefore the optimal learning rate for the sigmoid activation function is lower than the step function.

Feature-selection using the chi-squared criterion was not performed in this iteration of the experiment, therefore the model accuracy can definitely be further improved. Introducing feature-selection must be done to improve the model's performance. Both more time to train and speeding up the chi-squared calculation are ways to perform this improvement.