Mark Shinozaki
Homework #3

Q1. Answer the following with a yes or no along with proper justification.

a. Is the decision boundary of voted perceptron linear?
   - Yes, the decision boundary of voted perceptron is linear. Voted perceptron is an extension of the perceptron algorithm that uses multiple perceptron's and combines their decisions through a voting mechanism. Each perceptron in the ensemble still learns a linear decision boundary, and the final decision boundary is linear combination of individual decision boundaries.

b. Is the decision boundary of averaged perceptron linear?
   - Yes, the decision boundary of averaged perceptron is also linear. Averaged perceptron is another extension of the perceptron algorithm that updates the weight vector based on the average of all previous weight vectors. The final weight vector is linear combination of all the weight vectors, and thus the decision boundary is also linear.

Q2. (5 points) Consider the following setting. You are provided with n training examples: $(x_1, y_1, h_1), (x_2, y_2, h_2), \cdots, (x_n, y_n, h_n)$, where x_i is the input example, y_i is the class label (+1 or -1), and $h_i > 0$ is the importance weight of the example. The teacher gave you some additional information by specifying the importance of each training example. How will you modify the perceptron algorithm to be able to leverage this extra information? Please justify your answer.

Normal perceptron Algorithm -

$$w^{t+1} = \begin{cases} w^t + y^t x^t & (y^t(w^t \cdot x^t + b^t) <= 0) \\ w^t & (y^t(w^t \cdot x^t + b^t) > 0) \end{cases}$$

$$b^{t+1} = \begin{cases} b^t + y^t & (y^t(w^t \cdot x^t + b^t) \leq 0) \\ b^t & (y^t(w^t \cdot x^t + b^t) > 0) \end{cases}$$

Modified perceptron Algorithm

$$w^{t+1} = \begin{cases} w^t + c^t y^t x^t & (y^t(w^t \cdot x^t + b^t) <= 0) \\ w^t & (y^t(w^t \cdot x^t + b^t) > 0) \end{cases}$$

$$b^{t+1} = \begin{cases} b^t + c^t y^t & (y^t(w^t \cdot x^t + b^t) \leq 0) \\ b^t & (y^t(w^t \cdot x^t + b^t) > 0) \end{cases}$$

* One way to modify the perceptron algorithm to incorporate the importance weights is to use a variant called the weighted perceptron algorithm. We modify the update rule for the model weights by scaling the classification error by the importance weight of the example, the update rule becomes:

$w \leftarrow w + \eta y_i h_i x_i$

* In this setting where we have access to importance weights for each training example, we can modify the perceptron algorithm by using the weighted perceptron algorithm. The update rule for the model weights is modified by scaling the classification error by the importance weight of the example. This modification ensures that examples with higher importance weights are given more weight in the update rule, as they are more informative and contribute more to the final model. The choice of learning rate and importance weights can have a significant impact on the performance of the algorithm and should be chosen based on the specific task and data at hand.

Mark Shinozaki
Homework #3

Q3. (5 points) Consider the following setting. You are provided with n training examples: $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$, where x_i is the input example, and y_i is the class label (+1 or -1). However, the training data is highly imbalanced (say 90% of the examples are negative and 10% of the examples are positive) and we care more about the accuracy of positive examples. How will you modify the perceptron algorithm to solve this learning problem? Please justify your answer.

To modify the perceptron algorithm for imbalanced data, we can use a technique called class weighting. The idea behind class weighting is to give more weight to the minority class (in this case, the positive class) during training, so that the algorithm pays more attention to it.

Specifically, we can modify the update rule of the perceptron algorithm as follows:

- If the true label yi is positive and the predicted label ŷi is negative (i.e., ŷi = -1), then update the weight vector as:

```
w ← w + α * yi * x
```

- If the true label yi is negative and the predicted label ŷi is positive (i.e., ŷi = +1), then update the weight vector as:

```
w ← w + β * yi * x
```

- To conclude, the modified perceptron algorithm can be used to address imbalanced data by using class weighting. This technique involves giving more weight to the minority class during training, and modifying the update rule to account for the class imbalance.

- By setting the hyperparameters alpha and beta to control the weights given to the positive and negative classes, the algorithm can focus more on correctly classifying positive examples.

- However, it's important to note that this modification assumes that the test data has the same class distribution as the training data. If this assumption doesn't hold, other techniques like resampling or cost-sensitive learning may be more appropriate.

Mark Shinozaki
Homework #3

Q4. (10 points)

You were just hired by MetaMind. MetaMind is expanding rapidly, and you decide to use your machine learning skills to assist them in their attempts to hire the best. To do so, you have the following available to you for each candidate i in the pool of candidates $\mathcal{I}$: (i) Their GPA, (ii) Whether they took Data Mining course and achieved an A, (iii) Whether they took Algorithms course and achieved an A, (iv) Whether they have a job offer from Google, (v) Whether they have a job offer from Facebook, (vi) The number of misspelled words on their resume. You decide to represent each candidate $i \in \mathcal{I}$ by a corresponding 6-dimensional feature vector $f(x^{(i)})$. You believe that if you just knew the right weight vector $w \in \Re^6$ you could reliably predict the quality of a candidate i by computing $w \cdot f(x^{(i)})$. To determine w your boss lets you sample pairs of candidates from the pool. For a pair of candidates (k, l) you can have them face off in a "DataMining-fight." The result is $\texttt{score}\,(k \succ l)$, which tells you that candidate k is at least $\texttt{score}\,(k \succ l)$ better than candidate l. Note that the score will be negative when l is a better candidate than k. Assume you collected scores for a set of pairs of candidates $\mathcal{P}$.

Describe how you could use a perceptron based algorithm to learn the weight vector w. Make sure to describe the basic intuition; how the weight updates will be done; and pseudo-code for the entire algorithm.

Perceptron algorithm is a binary classification machine learning algorithm that works as follows:

- Activation = weights * inputs + Bias;
- Predict 1 if Activation > 0.0 ;
- Predict 0 if Activation <= 0.0;

The implementation of the perceptron based algorithm for the given scenario is outlined as follows:

Collected scores for a set of pairs of candidates = P ;

The result is score (k l), which tells you that candidate k is at least score (k l) better than candidate l.

Note that the score will be negative when l is a better candidate than k.

Then for the perceptron based algorithm :

Activation A = Weights * Inputs + Bias

=> A = w * P + B.

Consequently, predict 1 if Activation > 0.0 & predict 0 if Activation <= 0.0 where :

i. 1 denotes selction of the candidate for the job ;

ii. 0 denotes failure of the candidate for selction for the job.

The Pseudocode :

Mark Shinozaki
Homework #3

The Pseudocode of the algorithm is as follows :

1. Set Bias B = demand of MetaMind for no. of candidates ;

2. For each set of candidates in P based on the score (k, l) :

Activation A = Weights * Inputs + Bias ie. A = w * P + B.

Predict 1 if Activation > 0.0 & predict 0 if Activation <= 0.0

If k>l, then A > 0.0 and thus candidates qualify for the final list ;

Otherwise if l k<=l, then A < = 0.0 and the candidates fail to appear in the final list ;

3. Get final list of candidates F in descending order of selection capability and merit.

4. Select first B candidates from the list F according to demand.

Q5. (15 points) Suppose we have n_+ positive training examples and n_- negative training examples. Let C_+ be the center of the positive examples and C_- be the center of the negative examples, i.e., $C_+ = \frac{1}{n_+} \sum_{i:\ y_i=+1} x_i$ and $C_- = \frac{1}{n_-} \sum_{i:\ y_i=-1} x_i$. Consider a simple classifier called CLOSE that classifies a test example x by assigning it to the class whose center is closest.

- Show that the decision boundary of the CLOSE classifier is a linear hyperplane of the form $sign(w \cdot x + b)$. Compute the values of w and b in terms of C_+ and C_-.

The CLOSE classifier assigns a test example x to the positive class if its distance from the center of positive examples C+ is less than or equal to its distance from the center of negative examples C-, i.e., ||x - C+|| ≤ ||x - C-||. This inequality can be rewritten as:

||x - C+||^2 - ||x - C-||^2 ≤ 0.

Mark Shinozaki
Homework #3

Expanding the norms, we get:

```
(x - C+).(x - C+)-(x-C-).(x-C-) ≤ 0
```

Expanding the dot products, we get:

```
x.x-2x.C+ + C+.C+ - x.x +2x.C- -C-.C- ≤ 0
```

Simplifying, we get:

```
2(x.C- - x.C+) + C+.C+ - C-.C- ≤ 0
```

Rearranging, we get:

```
x.(C+ - C-) + (C+.C+ - C-.C-)/2 ≤ 0
```

Explanation:

The inequality x.(C+ - C-) + (C+.C+ - C-.C-)/2 ≤ 0 is the decision boundary of the CLOSE classifier. It separates positive and negative classes based on the distance of a test example x from the centers of the two classes. The first term measures how much x deviates from the midpoint between the centers, while the second term captures the relative magnitudes of the centers. The CLOSE classifier assigns x to the positive class if and only if it satisfies the inequality.

This is a linear equation in x with w = C+ - C- and b = (C+.C+ - C-.C-)/2. Therefore, the decision boundary of the CLOSE classifier is a linear hyperplane of the form sign(w.x + b), where w = C+ - C- and b = (C+.C+ - C-.C-)/2

Mark Shinozaki
Homework #3

Q6. (10 points) Please read the following paper and write a brief summary of the main points in at most TWO pages.

Pedro M. Domingos: A few useful things to know about machine learning. Communications of ACM 55(10): 78-87 (2012)
`https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf`

Summary of the article

In the first part of the article, the authors go over a summary of important topics and a brief overview of the kind of concepts that machine learning covers. This article discusses machine learning systems, which are programs that automatically learn from data, and their use in various fields such as web search, spam filters, and fraud detection. It explains the three components of machine learning: representation, evaluation, and optimization, and how they apply to classification, which is the most commonly used type of machine learning. The article also provides examples of machine learning algorithms for each component discusses some the key issues in choosing them. The next part of the article also discusses fundamental parts of machine learning, it discusses that a goal of machine learning is to have generalization beyond the training set. Essentially, its easy to do well on the training set by just memorizing examples but warns that its not indictive of success on new data. The article also discusses the importance of separating the training and the test data sets to reduce contamination of the classifier by test data, and the need for cross-validation. The article emphasizes that data alone is not enough, and some knowledge or assumptions beyond that data must be embodied in the learner to generalize beyond it. Finally, it explains that the choice of representation is crucial as it determines the kinds of knowledge easily expressed in it, and that the most useful learners are those that embody knowledge relevant to the domain. The next part of the article discusses two major problems in machine learning, overfitting and curse of dimensionality. Overfitting is when the model learns to capture the noise in the training data, causing it to perform poorly on new data. The text provides several techniques to combat overfitting, including regularization and statistical significance tests. The curse of dimensionality refers to the difficulty in generalizing correctly when the number of features in the input space increases. As the number of features increases, a fixed-sized training set covers a dwindling fraction of the input space, making generalization exponentially harder. Additionally, the similarity-based reasoning that machine learning algorithms depend on break downs in high dimensions. In high dimensions, examples become increasingly alike, and the choice of the nearest neighbor (and therefore of the class) becomes effectively random. Next the article discusses the most common type of guarantee is a bound on the number of examples needed for good generalization. The article explains the simple argument behind these bounds and their pessimistic nature due to the hypothesis space. Theoretical guarantees are useful for algorithm design and understanding but not as a criterion for practical decisions, The most important factor for success in machine learning projects is feature engineering. Learning is easier when independent features correlate well with the class. On the other hand, if the features are correlated or noisy, learning becomes more difficult, and overfitting may occur. The article concludes by emphasizing the importance of human expertise in feature engineering and iterative nature of machine learning projects. In the final section of the article the text discusses some of the more challenges that exist in machine learning, one such issue refers to the limitations of Occam's razor. It argues that there is no necessary connection between the number of parameters of a model and tis tendency to overfit, and that smaller hypothesis spaces may not necessarily result in better generalization performance. Furthermore, the fact that a function can be

represented does not mean it can be learned, and the existence of many local optima can make it difficult for learner to find the true function. The text also emphasizes the importance of trying different learners and representations and suggests that finding methods to learn deeper representations is one of the major research frontiers in machine learning. Finally in conclusion, the wrapping up part of this article, It argues that there is no necessary connection between the number of parameters of a model and its tendency to overfit, and that smaller hypothesis spaces may not necessarily result in better generalization performance. Furthermore, the fact that a function can be represented does not mean it can be learned, and the existence of many local optima can make it difficult for a learner to find the true function. The text also emphasizes the importance of trying different learners and representations and suggests that finding methods to learn deeper representations is one of the major research frontiers in machine learning. Finally, it cautions against the common mistake of assuming that correlation implies causation and suggests that machine learning models should be used as guides to action rather than definitive proof of causation.