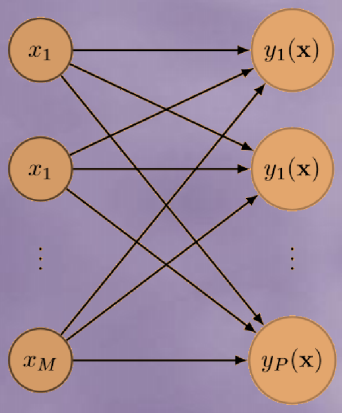

Neural Networks and Deep Learning

Daniel Khashabi [1]
KHASHAB2@ILLINOIS.EDU

0.1 1

1

0.2 1

1

0.3 1

1

0.4 Problems

0.4.1 The perceptron algorithm

In this question, we will be asking you about Perceptrons and their variants.Let $D = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ where the j-th example $\mathbf{x}^{(j)}$ is associated with the label $y^{(j)} \in \{-1, +1\}$. Each example $\mathbf{x}^{(j)}$ is a bit-vector of length n, i.e. $\mathbf{x}^{(j)} \in \{0, 1\}^n$, with the interpretation that the i-th bit of the vector $(x_i^{(j)})$ is 1 if the element described by $\mathbf{x}^{(j)}$ has the i-th attribute on.

1. 1. Let us first consider a Perceptron where the positive example $\mathbf{x}$ satisfies $\mathbf{w} \cdot \mathbf{x} \geq \theta$, where $\mathbf{w} \in \mathbb{R}^n$, $\theta \in \mathbb{R}$ and $\mathbf{x}$ is some example $\mathbf{x}^{(j)}$ from D.
    1. Suggest an equivalent representation of this Perceptron in the form of $\underline{\mathbf{w}' \cdot \mathbf{x}' \geq 0}$ given an example $\mathbf{x}^{(j)}$, where $\mathbf{x}' \in \{0, 1\}^{n'}$ for some suitable integer n'.

       Define $n' = n + 1$

[1] This is part of my notes; to find the complete list of notes visit `http://web.engr.illinois.edu/~khashab2/learn.html`. This work is licensed under a Creative Commons Attribution-NonCommercial 3.0 License. This document is updated on May 9, 2013.

Define $\mathbf{w}' = \langle \mathbf{w}, -\theta \rangle$

Define $\mathbf{x}' = \langle \mathbf{x}^{(j)}, 1 \rangle$

2. In the following table, we describe a specific data set S. Using an initialization of $\mathbf{w}' = \mathbf{0}$, i.e. the zero vector, and a learning rate of $R = 1$, complete the columns under **(a)** of the table using the Perceptron learning algorithm.
   Answer: I just rewrite the main points of the algorithms and run this on the values given in the table bellow:
   i. Initialize the weights vector, $\mathbf{w}' = 0$.
   ii. For each sample i, if the the output value didn't match the real value ($f(\mathbf{w}'.\mathbf{x}'^{(i)}) \neq y'^{(i)}$) we do this iteration: $\mathbf{w}' \leftarrow \mathbf{w}' + \eta.\mathbf{x}'^{(i)}.y'^{(i)}$

	S			(a)		(b)	
j	$\mathbf{x}_1^{(j)}$	$\mathbf{x}_2^{(j)}$	$y^{(j)}$	Mistake? Y/N	Updated $\mathbf{w}'$	Mistake? Y/N	Updated $\mathbf{w}'$
Initialization				——	$\mathbf{0}$	——	$\mathbf{0}$
1	1	1	$+1$	N	$\mathbf{0}$	Y	$(1, 1, 1)$
2	1	0	-1	Y	$(-1, 0, -1)$	Y	$(0, 1, 0)$
3	0	1	$+1$	Y	$(-1, 1, 0)$	Y	$(0, 2, 1)$

(b) Using the same data set used above, we now consider a Perceptron with margin $\gamma > 0$. We can also represent this with $\mathbf{w}' \cdot \mathbf{x}' \geq 0$ like in Perceptron but using a different update rule for the weights.

1. Let the margin $\gamma > 0$ and learning rate $R > 0$. For a given $(\mathbf{x}^{(j)}, y^{(j)})$, write down the update rule for the Perceptron with margin.
   Answer: We change our criteria base of the margin; so we say the output is positive if we have $\mathbf{w}' \cdot \mathbf{x}' \geq \gamma$ and it's negative if we have $\mathbf{w}' \cdot \mathbf{x}' < -\gamma$. If the output is in range $(-\gamma, +\gamma)$, we call it margin mistake. So for the case of margin perceptron we update the weight vector whenever we make incorrect prediction or margin mistake similar to the previous one.
2. We described a specific data set S in a table earlier. Using an initialization of $\mathbf{w}' = \mathbf{0}$, that is, the zero vector, a learning rate of $R = 1$ and margin $\gamma = 1.5$, complete the columns under **(b)** of the table using the *Perceptron with margin* learning algorithm.
   Answer: Shown in the above table.

(c) Suppose we have the same data set S and now we would like to learn a linear separator of the form $\mathbf{w}' \cdot \mathbf{x}' \geq 0$, the canonical representation for any separating hyperplane. This time however, we would like to learn the weights $\mathbf{w}'$ by *minimizing* the error made by the linear separator over S.
We define the error made by $\mathbf{w}'$ over S using the *hinge loss* function, defined as $L(y^{(j)}, \mathbf{x}^{(j)}, \mathbf{w}') = \max(0, 1 - y^{(j)}\mathbf{w}' \cdot \mathbf{x}^{(j)'})$, where $\mathbf{x}^{(j)'}$ is the representation of example $\mathbf{x}^{(j)}$ in the form of $\mathbf{x}'$ in the canonical representation.
Thus the goal of learning is to minimize the following error:

$$E = \text{Error}(\mathbf{w}', D) = \sum_{j=1}^{m} L(y^{(j)}, \mathbf{x}^{(j)}, \mathbf{w}') = \sum_{j=1}^{m} \max(0, 1 - y^{(j)}\mathbf{w}' \cdot \mathbf{x}^{(j)'})$$

One way to do this is to make use of Stochastic Gradient Descent.

1. Write the pseudocode for Stochastic Gradient Descent using this hinge loss function with a fixed learning rate of $R > 0$.
   Answer: In the stochastic gradient descent, instead of going over training samples in order, we shuffle them. To minimize the loss function mentioned in the question,

we take its derivative with respect to the weight vector.

$$\frac{\partial E}{\partial \mathbf{w}'} = \begin{cases} -y\,\mathbf{x} & \text{if } y\,\mathbf{x}\cdot\mathbf{w}' < 1 \\ 0 & \text{if } y\,\mathbf{x}\cdot\mathbf{w}' \geq 1 \end{cases}$$

Which is essentially the same as perceptron iteration with margin 1, i.e. when we are doing perceptron iteration with margin 1 are minimizing the E loss function mentioned above.

i. First we initialize our weight vector using some random values or zero vector: $\mathbf{w}' = \mathbf{0}$

ii. Shuffle the training samples, and in the shuffled set do perceptron iterations. For each sample i, if the the output value didn't satisfy the margin = 1 requirement, we do this iteration: $\mathbf{w}' \leftarrow \mathbf{w}' + \eta.\mathbf{x}'^{(i)}.y'^{(i)}$

2. Suggest a condition on the problem definition that will make the Stochastic Gradient Descent algorithm identical to the Perceptron with Margin algorithm.
Answer: We can change the loss function in this way :

$$E' = \text{Error}(\mathbf{w}', D) = \sum_{j=1}^{m} L(y^{(j)}, \mathbf{x}^{(j)}, \mathbf{w}') = \sum_{j=1}^{m} \max(0, \gamma - y^{(j)}\mathbf{w}' \cdot \mathbf{x}^{(j)'})$$