CS188 Spring 2019 Section 11: Perceptrons and Logistic Regression

1 Perceptron

The Algorithm The perceptron algorithm works as follows:

1. Initialize all weights to 0: $\mathbf{w} = \mathbf{0}$
2. For each training sample, with features $\mathbf{f}(x)$ and class label $y^* \in \{-1, 1\}$, do:
   (a) Take the dot product, s, between the sample features and the current weights: $s = \mathbf{w}^\top \mathbf{f}(x)$
   (b) Predict a class, $\hat{y}$ for the sample as follows:
   $\hat{y} = +1$ if $s \geq 0$, $\hat{y} =$ -1 otherwise.
   (c) Compare the predicted label $\hat{y}$ to the true label y^*:
   - If $\hat{y} = y^*$, do nothing
   - Otherwise, if $\hat{y} \neq y^*$, then update your weights: $\mathbf{w} \leftarrow \mathbf{w} + y^* * f(x)$
3. If you went through every training sample without having to update your weights (all samples predicted correctly), then terminate. If any at least one sample was predicted incorrectly, then repeat step 2

Updating weights
Let us now examine and justify the procedure for updating our weights.
Recall that in step 2b above, we assigned our predicted label $\hat{y}$ to be either 1 or -1. To update the weights, we first check if the predicted label is correct. If it is, i.e. $\hat{y} = y^*$, then do nothing–"don't fix what's not broken", as they say. When they are not equal, then update the weight vector as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + y^* * \mathbf{f}(x)$$

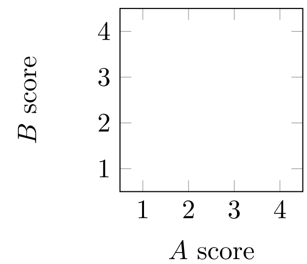
where y^* is the true label, which is either 1 or -1, and x the training sample which we mis-classified.

One way to look at this is to see it as a balancing act. If our weights, when multiplied by our sample's features, give us a negative score s when we wanted a positive score (i.e. $y^* = 1$), then our weights are probably too small (for positive-valued features, or too large for negative-valued). Since $y^* = 1$, we, according to this update rule, will add the feature values to our weights to try and make them closer to an optimal set of weights. If our product yields a positive score, where we wanted a negative score, then our weights are probably too big, and so we would like to decrease them. To do so, noting that $y^* = -1$, we will subtract our mis-classified sample from our weight vector.

2 Perceptron

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 4. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

#	Movie Name	A	B	Profit?
1	Pellet Power	1	1	-
2	Ghosts!	3	2	+
3	Pac is Bac	2	4	+
4	Not a Pizza	3	4	+
5	Endless Maze	2	3	-



(a) First, you would like to examine the linear separability of the data. Plot the data on the 2D plane above; label profitable movies with + and non-profitable movies with − and determine if the data are linearly separable.

(b) Now you decide to use a perceptron to classify your data. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, $f_1 =$ score given by A and $f_2 =$ score given by B.

Run one pass through the data with the perceptron algorithm, filling out the table below. Go through the data points in order, e.g. using data point #1 at step 1.

step	Weights	Score	Correct?
1	[-1, 0, 0]	$-1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = -1$	yes
2			
3			
4			
5			

Final weights:

(c) Have weights been learned that separate the data?

(d) More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Circle the scenarios for which a perceptron using the features above can indeed perfectly classify movies which are profitable according to the given rules:

(a) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be profitable, and otherwise it won't be.

(b) Your reviewers are art critics. Your movie will be profitable if and only if each reviewer gives either a score of 2 or a score of 3.

(c) Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree.

Q3. Optimization

We would like to classify some data. We have N samples, where each sample consists of a feature vector $\mathbf{x} = \{x_1, \cdots, x_k\}$ and a label $y = \{0, 1\}$.

We introduce a new type of classifier called logistic regression, which produces predictions as follows:

$$P(Y = 1|X) = h(\mathbf{x}) = s\left(\sum_i w_i x_i\right) = \frac{1}{1 + \exp(-(\sum_i w_i x_i))}$$

$$s(\gamma) = \frac{1}{1 + \exp(-\gamma)}$$

where $s(\gamma)$ is the logistic function, $\exp x = e^x$, and $\mathbf{w} = \{w_1, \cdots, w_k\}$ are the learned weights.

Let's find the weights w_j for logistic regression using stochastic gradient descent. We would like to minimize the following loss function for each sample:

$$L = -[y \ln h(\mathbf{x}) + (1 - y) \ln(1 - h(\mathbf{x}))]$$

(a) Find dL/dw_i. Hint: $s'(\gamma) = s(\gamma)(1 - s(\gamma))$.

(b) Write the stochastic gradient descent update for w_i. Our step size is η.