CS 188
Summer 2019

Section Handout 13 Solutions

1 Perceptron Notes

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 Kernel Notes

Perceptron Weights
Now let's revisit the weight w_y corresponding to class y of a perceptron. What is the final value of w_y? Can it be any real vector? No, from our update rule, we see that w_y either stays the same or gets updated as linear combination of the features.

i.e. $w_y = 0 + f(x_1) - f(x_5) + \ldots$ or compactly, $w_y = \sum_i \alpha_{i,y} f(x_i)$.

Key observation: we can construct weight vectors (**primal representation**) from update counts (**dual representation**) That is, it is sufficient to keep track with $\alpha_y = \langle \alpha_{1,y} \alpha_{2,y} \ldots \alpha_{n,y} \rangle$

Dual Perceptron
How do we classify a new example x?

$$\begin{aligned}
\text{score}(y, x) &= w_y \cdot f(x) \\
&= \left(\sum_i \alpha_{i,y} f(x_i) \right) \cdot f(x) \\
&= \sum_i \alpha_{i,y} \left(f(x_i) \cdot f(x) \right) \\
&= \sum_i \alpha_{i,y} K(x_i, x)
\end{aligned}$$

From the result of the derivation, we can see that if someone tells us the value of K for each pair of examples, we never need to build the weight vectors (or the feature vectors)!

The dual perceptron algorithm works as follows:

1. Initialize all dual weights (counts) to 0: $\alpha = \mathbf{0}$
2. For each training sample x_n, do:
   (a) Predict a class, $\hat{y}$ for the sample as follows:
   $\hat{y} = \arg\max_y \sum_i \alpha_{i,y} K(x_i, x_n)$
   (b) Compare the predicted label $\hat{y}$ to the true label y^*:
   - If $\hat{y} = y^*$, do nothing
   - Otherwise, lower count of wrong class (for this instance), raise count of right class (for this instance).

   $$\alpha_{y,n} = \alpha_{y,n} - 1$$
   $$\alpha_{y^*,n} = \alpha_{y^*,n} + 1$$

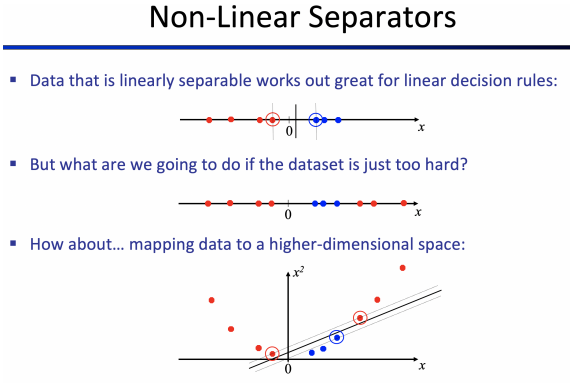
   note the similarity to the regular perceptron's update rule!

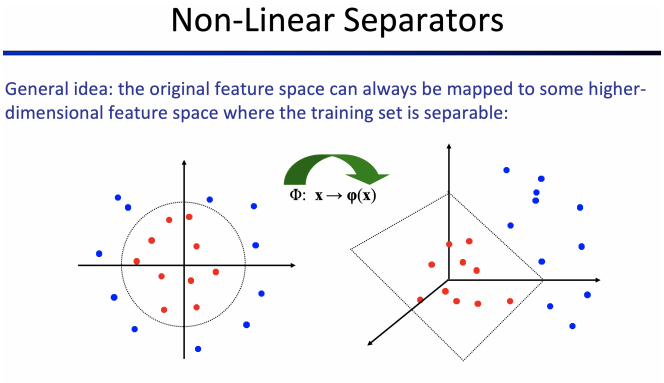
   $$w_y = w_y - f(x_n)$$
   $$w_{y^*} = w_{y^*} + f(x_n)$$
3. If you went through every training sample without having to update your weights (all samples predicted correctly), then terminate. If any at least one sample was predicted incorrectly, then repeat step 2.

Why kernelize?
If we had a black box (kernel) K that told us the dot product of two examples x and x', then we could work entirely with the dual representation, which means that we don't have to ever take dot products ("kernel trick")!

$$\begin{aligned}\text{score}(y, x) &= w_y \cdot f(x) \\ &= \sum_i \alpha_{i,y} K(x_i, x)\end{aligned}$$

So far, we have seen a very strange way of doing a very simple calculation. So why kernels? Remember that regular perceptron can only handle data that are linearly separable! Consider the following examples. How can we modify the linear separating ability of perceptron to handle non-linear decision boundaries?





Here is where kernels step in! Kernels implicitly map original vectors to higher dimensional spaces, take the dot product there, and hand the result back!

(Optional:) Some example of valid kernels include:

1. Linear kernel: $K(x, x') = x \cdot x' = \sum_i x_i x'_i$
2. Quadratic kernel: $K(x, x') = (x \cdot x' + 1)^2 = \sum_{i,j} x_i x_j x'_i x'_j + 2 \sum_i x_i x'_i + 1$
3. Gaussian RBF kernel: infinite dimensional representation: $K(x, x') = \exp\left(-\|x - x'\|^2\right)$

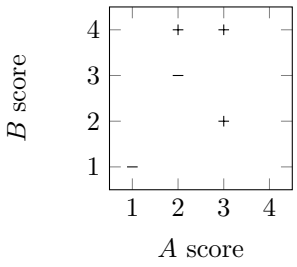
At this point, some astute readers might wonder: Why can't we just add these features on our own (e.g. add all pairs of features instead of using the quadratic kernel)? Yes, in principle, we can just compute them and run the existing algorithms. However, the number of features can quickly blow up as we increase the complexity of our features.

Key observation: Kernels let us compute with these features implicitly! For example, computing the implicit dot product in quadratic kernels takes much less space and time per dot product.

3 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. 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	[-1, 0, 0]	$-1 \cdot 1 + 0 \cdot 3 + 0 \cdot 2 = -1$	no
3	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 2 + 2 \cdot 4 = 14$	yes
4	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 3 + 2 \cdot 4 = 17$	yes
5	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 2 + 2 \cdot 3 = 12$	no

Final weights: [-1, 1, -1]

(c) Have weights been learned that separate the data? With the current weights, points will be classified as positive if $-1 \cdot 1 + 1 \cdot A + -1 \cdot B \geq 0$, or $A - B \geq 1$. So we will have incorrect predictions for data points 3:

$$-1 \cdot 1 + 1 \cdot 2 + -1 \cdot 4 = -3 < 0$$

and 4:

$$-1 \cdot 1 + 1 \cdot 3 + -1 \cdot 4 = -2 < 0$$

Note that although point 2 has $w \cdot f = 0$, it will be classified as positive (since we classify as positive if $w \cdot f \geq 0$).

(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. Can classify (consider weights $[-8, 1, 1]$)

(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. Cannot classify

(c) Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree. Cannot classify

Q4. Perceptron and Kernels

A kernel is a mapping $K(x, y)$ from pairs vectors in $\mathbb{R}^d$ into the real numbers such that $K(x, y) = \Phi(x) \cdot \Phi(y)$ where Φ is a mapping from $\mathbb{R}^d$ into $\mathbb{R}^D$ where D is possibly different from d and even infinite. We say that a mapping $K(x, y)$ for which such Φ exists is a valid kernel.

(a) The following binary class data has two features, A and B.

Index	A	B	Class
1.	1	1	1
2.	0	3	-1
3.	1	-1	1
4.	3	0	-1
5.	-1	1	1
6.	0	-3	-1
7.	-1	-1	1
8.	-3	0	-1

(i) Select all true statements:

- ☐ This data is linearly separable.
- ■ This data is linearly separable if we use a feature map $\phi((A, B)) = (A^2, B^2, 1)$.
- ■ There exists a kernel such that this data is linearly separable.
- ■ For all datasets in which no data point is labeled in more than one distinct way, there exists a kernel such that the data is linearly separable.
- ☐ For all datasets, there exists a kernel such that the data is linearly separable.
- ☐ For all valid kernels, there exists a dataset with at least one point from each class that is linearly separable under that kernel.
- ☐ None of the above.

We will be running both the primal (normal) binary (not multiclass) perceptron and dual binary perceptron algorithms on this dataset. We will initialize the weight vector w to $(1, 1)$ for the primal perceptron algorithm. Accordingly, we will initialize the α vector to $(1, 0, 0, 0, 0, 0, 0, 0)$ for the dual perceptron algorithm with the kernel $K(x, y) = x \cdot y$. Pass through the data using the indexing order provided. There is no bias term.

Write your answer in the box provided. Show your work outside of the boxes to have a chance at receiving partial credit.

(ii) What is the first misclassified point?

Point 2.

(iii) For the *primal* perceptron algorithm, what is the weight vector after the first weight update?

The weight vector after the first weight update will be:

$$w = (1, 1) - (0, 3) = (1, -2) \quad (1)$$

For your convenience, the data is duplicated on this page.

Index	A	B	Class
1.	1	1	1
2.	0	3	-1
3.	1	-1	1
4.	3	0	-1
5.	-1	1	1
6.	0	-3	-1
7.	-1	-1	1
8.	-3	0	-1

(iv) For the *dual* perceptron algorithm, what is the α vector after the first weight update?

The α vector after the first update will be:

$$\alpha = (1, -1, 0, 0, 0, 0, 0, 0) \quad (2)$$

(v) What is the second misclassified point?

Point 4.

(vi) For the *primal* perceptron algorithm, what is the weight vector after the second weight update?

The weights after the second weight update will be:

$$w = (1, -2) - (3, 0) = (-2, -2) \quad (3)$$

(vii) For the *dual* perceptron algorithm, what is the α vector after the second weight update?

The α vector after the second update will be:

$$\alpha = (1, -1, 0, -1, 0, 0, 0, 0) \quad (4)$$

(b) Consider the following kernel function: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2$ where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$. Find a valid Φ map for this kernel. That is, find a vector-to-vector function ϕ such that $\phi(\mathbf{x}) \cdot \phi(\mathbf{y}) = K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2$. Show work to have a chance at receiving partial credit. Any precise answer format is acceptable.

Expanding $(x \cdot y)^2 = (x_1 y_1 + x_2 y_2)^2 = x_1^2 y_1^2 + 2 x_1 y_1 x_2 y_2 + x_2^2 y_2^2$ so the mapping $\Phi(x) = [x_1^2, \sqrt{2} x_1 x_2, x_2^2]$ is valid.