CS 536 : Perceptrons - Geometry and Convergence 16:198:536

In the usual way, we have a collection of data points $\{(\underline{x}^i, y^i)\}_{i=1,\dots,m}$ sampled from some underlying distribution. Each $\underline{x}$ is a vector of k-features, and y represents some binary class the data point belongs to (will purchase/won't purchase, is a dog/is not a dog). We take the classes to be tagged with $y = 1$ or $y = -1$ in this case. Our goal is that for any new feature vector $\underline{x}$, we want to be able to predict the class value y from those features. **Perceptrons** assume that the class is a function of the features, $y = f(\underline{x})$, where f is assumed to be of the following form:

$$f(\underline{x}) = \text{sign}(b + x_1 w_1 + x_2 w_2 + \ldots + x_k w_k), \tag{1}$$

for a bias value b and a vector of weights $\underline{w}$. Note, this can be expressed in a more 'decision-esque' framework as:

$$f(\underline{x}) = \begin{cases} 1 & \text{if } \underline{w}.\underline{x} + b \geq 0 \\ -1 & \text{else.} \end{cases} \tag{2}$$

The goal of perceptrons, as a learning problem, is to determine a bias value b and a vector of weights $\underline{w}$ such that each point in the data set is correctly classified: $f(\underline{x}^i) = y^i$ for each $i = 1, \dots, m$.

Geometry

What do perceptrons 'look like'?

In one dimension ($k = 1$), this reduces to $f(x) = \text{sign}(b + wx)$, or for $w > 0$,

$$f(x) = \begin{cases} 1 & \text{if } x \geq -b/w \\ -1 & \text{else.} \end{cases} \tag{3}$$

In this case, the perceptron represents a simple thresholding function - if x is above the threshold $-b/w$, x is classified as $y = 1$, if x is below the threshold then x is classified as $y = -1$. Perceptrons can model data where all the positive class points lie on one side of a value, and negative class points lie on the other.

In general, for higher dimensions, the 'cutoff' or threshold is the line / hyperplane defined by $\underline{w}.\underline{x} + b = 0$. Everything on one side of this *linear separator* is classified as the positive class, everything on the other side is classified as the negative class.

Constraints

This interpretation suggests some immediate constraints on what kinds of data perceptrons can model. For instance, in dimension $k = 1$, the data $(x = -1, y = -1)$, $(x = 0, y = 1)$, $(x = 1, y = -1)$ cannot be modeled with a perceptron. There is no threshold value $-b/w$ that separates this points. Similarly in $k = 2$ dimensions, the classic example is $((0,0),1), ((0,1),-1), ((1,0),-1), ((1,1),1)$ - there is no line that can be drawn in two dimensions that separates the positive and negative class points in this data set. In general, perceptrons can capture data points where the classes are *linearly separable.* Non-linear separators represent a problem for later focus.

Learning Algorithm

But can perceptrons be learned? In particular, given a data set, how can we determine a bias value b and weight vector $\underline{w}$ that correctly classify each data point? Given that there are uncountably many possible choices for each, this seems to be a considerable problem , but the following algorithm can be used to construct a linear separator (*if one exists!*) in a systematic way:

The Perceptron Learning Algorithm:

- Begin with $\underline{w} = 0, b = 0$
- Identify a point that is misclassified, i.e., $f(\underline{x}^i) \neq y^i$
- Update the weights and biases in the following way:

$$\begin{aligned} \underline{w} &\leftarrow \underline{w} + y^i \underline{x}^i \\ b &\leftarrow b + y^i. \end{aligned} \tag{4}$$

- Repeat until all points are correctly classified

The reasoning behind the algorithm is this: suppose that $\underline{w}, b$ misclassifies $(\underline{x}, y)$, that is, $\underline{w}.\underline{x} + b \geq 0$ and $y = -1$ or $\underline{w}.\underline{x} + b < 0$ and $y = 1$. Let $\underline{w}' = \underline{w} + y\underline{x}$ and $b' = b + y$. In that case, we have

$$\underline{w}'.\underline{x} + b' = (\underline{w} + y\underline{x}).\underline{x} + (b + y) = (\underline{w}.\underline{x} + b) + y\left(||\underline{x}||^2 + 1\right). \tag{5}$$

Hence, if $\underline{w}.\underline{x} + b$ is too small (negative when y is positive), $\underline{w}'.\underline{x} + b'$ is larger. If $\underline{w}.\underline{x} + b$ is too large (positive when y is negative), $\underline{w}'.\underline{x} + b'$ is smaller. Either way, if $\underline{x}, y$ is misclassified under $\underline{w}, b$, then it is closer to being correctly classified under $\underline{w}', b'$. In effect, the learning algorithm identifies an incorrectly classified point, and then drags the weight vector and bias value in a direction toward correctly classifying the point.

While this suggests that each step of the algorithm moves the model closer to correctly identifying the chosen point, it is potentially surprising that it doesn't move the model *away* from classifying other points. In fact, we can make the following claim:

Theorem: If the data is linearly separable, then the perceptron learning algorithm terminates in finite time, producing a linear separator $\underline{w}.\underline{x} + b = 0$ that correctly classifies all points.

We put off the proof of this result for a moment to develop first some simplifications and additional concepts, to support the proof of a much stronger result.

Simplification

It's convenient to simplify things by treating the bias value as simply another weight. In particular, imagine augmenting each feature vector by tacking a 1 in as a constant 'zero-th' feature:

$$(x_1, x_2, \ldots, x_k) \mapsto (1, x_1, x_2, \ldots, x_k), \tag{6}$$

effectively moving the problem into $k+1$ dimensions. In this case, we can similarly expand the weight vector treating the bias as the zeroth component:

$$b, (w_1, \ldots, w_k) \mapsto (b, w_1, w_2, \ldots, w_k). \tag{7}$$

While the decision threshold previously was given by $\underline{w}.\underline{x} + b = 0$, it can now be simply expressed as $\underline{w}.\underline{x} = 0$, i.e., the set of all x that are perpendicular to $\underline{w}$. Note, in this case, the update equations of the learning algorithm are simply

$$\underline{w} \leftarrow \underline{w} + y^i \underline{x}^i. \tag{8}$$

Additionally, it is convenient to treat the feature vectors as normalized, i.e., $||\underline{x}^i|| = 1$. Note that scaling each feature vector $\underline{x}$ by a positive constant (i.e., normalizing it) will not change the sign of the threshold value $\underline{w}.\underline{x}$.

In this simplified framework, it is convenient to define the following concept:

If the plane $\underline{w}.\underline{x} = 0$ is a linear separator for a data set, then the **margin** of this separator is defined to be the perpendicular distance from this plane to the nearest data point:

$$\gamma(\underline{w}) = \min_i \frac{|\underline{w}.\underline{x}^i|}{||\underline{w}||}, \tag{9}$$

where $|\underline{w}.\underline{x}|/||\underline{w}||$ is the projection of $\underline{x}$ onto the weight vector $\underline{w}$. *Geometrically, this comes from the fact that the weight vector $\underline{w}$ is perpendicular to the plane defined by $\underline{w}.\underline{x} = 0$.*

With this additional notion, we can state and prove a much stronger result:

Theorem: If a linear separator exists with weight vector $\underline{w}$, then the perceptron learning algorithm terminates in at most $1/\gamma(\underline{w})^2$ steps.

Proof of Termination

We want to show that if there is a linear separator, the learning algorithm terminates, and yields a weight vector that acts as a linear separator. Note, the algorithm only terminates when all points are correctly classified and there are no more errors, so any weight vector the algorithm terminates on *must* be a linear separator.

Let $\underline{w}_0, \underline{w}_1, \underline{w}_2, \ldots$ be the sequence of weight vectors produced by the perceptron learning algorithm. We would like to get a handle on this progression of weight vectors, understanding how they evolve, and ideally, how they terminate. Note that by the update equations, we have that

$$\underline{w}_{t+1} = \underline{w}_t + y^i \underline{x}^i, \tag{10}$$

where $\underline{x}^i, y^i$ is misclassified by $\underline{w}_t$. This relation yields a couple of important results.

Taking the dot product of each side with itself (effectively taking the norm squared of both sides) we get the following relation:

$$\underline{w}_{t+1}.\underline{w}_{t+1} = \underline{w}_t.\underline{w}_t + 2y^i \underline{w}_t.\underline{x}^i + (y^i)^2 \underline{x}^i.\underline{x}^i, \tag{11}$$

or (noting that $y^i = \pm 1$ and each $\underline{x}^i$ is normalized to be a unit vector,

$$||\underline{w}_{t+1}||^2 = ||\underline{w}_t||^2 + 2y^i \left(\underline{w}_t.\underline{x}^i\right) + 1. \tag{12}$$

Because $\underline{x}^i, y^i$ is misclassified by $\underline{w}_t$, $y^i \left(\underline{w}_t.\underline{x}^i\right) \leq 0$, hence

$$||\underline{w}_{t+1}||^2 \leq ||\underline{w}_t||^2 + 1. \tag{13}$$

Unwrapping this recursion yields the following relation, $||\underline{w}_T||^2 \leq ||\underline{w}_0||^2 + T$. Taking $\underline{w}_0 = 0$ as specified in the algorithm, we have

$$||\underline{w}_T|| \leq \sqrt{T}. \tag{14}$$

Hence, we expect the norm of the weight vector to be bounded over time, at most $\sqrt{T}$. What else can be said about the progression of weight vectors? Note that at this point, we have not used the fact that the data is linearly separable. Suppose that the data points can be separated with weight vector $\underline{w}$, i.e., $\underline{w}.\underline{x}^i \geq 0$ when $y^i = 1$ and

$\underline{w}.\underline{x}^i < 0$ when $y^i = -1$, or $y^i \left(\underline{w}.\underline{x}^i\right) = |\underline{w}.\underline{x}^i|$ for each data point. We make no claim that the progression of weight vectors will converge to $\underline{w}$, but it is useful to note how the weight vectors relate to $\underline{w}$. In particular:

$$\underline{w}.\underline{w}_{t+1} = \underline{w}.\underline{w}_t + y^i \left(\underline{w}.\underline{x}^i\right), \tag{15}$$

or

$$\frac{\underline{w}.\underline{w}_{t+1}}{||\underline{w}||} = \frac{\underline{w}.\underline{w}_t}{||\underline{w}||} + \frac{|\underline{w}.\underline{x}^i|}{||\underline{w}||}. \tag{16}$$

Minimizing with respect to all of the $\underline{x}^i$, we can relate the above back to the idea of the margin - in particular:

$$\frac{\underline{w}.\underline{w}_{t+1}}{||\underline{w}||} \geq \frac{\underline{w}.\underline{w}_t}{||\underline{w}||} + \gamma(\underline{w}). \tag{17}$$

Unwrapping this recursion as before yields the following relation:

$$\frac{\underline{w}.\underline{w}_T}{||\underline{w}||} \geq \frac{\underline{w}.\underline{w}_0}{||\underline{w}||} + \gamma(\underline{w})T, \tag{18}$$

or

$$\frac{\underline{w}.\underline{w}_T}{||\underline{w}||} \geq \gamma(\underline{w})T. \tag{19}$$

Here we utilize the Cauchy-Shwartz inequality to bound that dot product, that $|\underline{w}.\underline{w}_T| \leq ||\underline{w}||||\underline{w}_T||$. Putting all this together:

$$\gamma(\underline{w})T \leq \frac{\underline{w}.\underline{w}_T}{||\underline{w}||} \leq \frac{|\underline{w}.\underline{w}_T|}{||\underline{w}||} \leq ||\underline{w}_T|| \leq \sqrt{T}. \tag{20}$$

Hence, for every time T during the progression of the learning algorithm, we must have that

$$\gamma(\underline{w})T \leq \sqrt{T}, \tag{21}$$

or

$$T \leq \frac{1}{\gamma(\underline{w})^2}. \tag{22}$$

We may conclude at this point that the index of any weight vector in this progression is bounded by at most $1/\gamma(\underline{w})^2$. Since this is finite, this implies that there can only be finitely many such weight vectors, i.e., the algorithm must terminate in finite time. The proof additionally gives us a bound on the total number of steps:

$$T \leq \frac{1}{\max_{\underline{w}} \gamma(\underline{w})^2}, \tag{23}$$

where the maximum is taken over all separating weight vectors $\underline{w}$.

The main takeaway from this result is that the perceptron learning algorithm terminates in finite time if there is a linear separator. However, it is worth noting the following: the upper bound on the number of steps *increases with the maximum margin of any linear separator.* That is, for data sets with *small margin separators*, we typically would expect (or at least cannot exclude) that the learning algorithm will take a large number of steps. This makes a lot of sense - if any separator has a small margin, small perturbations in the weight vector will cause the perceptron to misclassify points, pushing some data points over the boundary. It will be very difficult to identify a separating hyperplane, shifting the weight vector point by point as we do. The larger the largest margin, the easier to find and more stable any given linear separator will be.