CS5339 Lecture Notes #1: The Perceptron Algorithm

Jonathan Scarlett

March 30, 2021

Useful references:

- Blog post by Jeremy Kun[1]
- MIT lecture notes,[2] lectures 1 and 2
- Chapter 4 (primarily Section 4.1.7) of Bishop's "Pattern Recognition and Machine Learning" book
- Section 9.1 of "Understanding Machine Learning" book

1 Binary Classification

The classification problem:

- As described in the introduction lecture, a *data set* is a collection of pairs: $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}_{t=1}^n$ where $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \{-1, +1\}$
- A *classifier* is a function $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \{-1, +1\}$ that takes $\mathbf{x}$ as input and tries to predict the corresponding label y
- *Linear classifiers* take the form

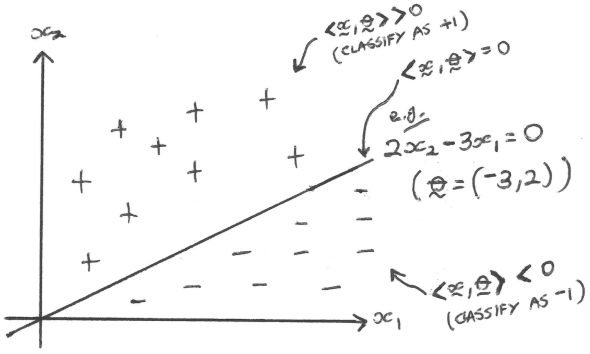
$$\text{Predict positive label} \iff \langle \boldsymbol{\theta}, \mathbf{x} \rangle > 0$$

  for some $\boldsymbol{\theta} \in \mathbb{R}^d$. This is equivalent to saying $f_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x})$.
  - The entries of $\boldsymbol{\theta}$ are written as $(\theta_1, \dots, \theta_d)$, and similarly for $\mathbf{x}$ and other vectors.
  - For two vectors $\mathbf{u}$ and $\mathbf{v}$ of the same length, $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u} = \sum_{t=1}^n u_t v_t$ is the standard inner product. Hence, $\langle \boldsymbol{\theta}, \mathbf{x} \rangle = \sum_{t=1}^n \theta_t x_t$ is a linear combination of the entries of $\mathbf{x}$ (weighted according to $\boldsymbol{\theta}$).
- In this lecture, assume that there exists a linear classifier (i.e., a choice of $\boldsymbol{\theta}$) that classifies everything in the data set $\mathcal{D}$ correctly. In this case, we say that $\mathcal{D}$ is *linearly separable*.

[1] http://jeremykun.com/2011/08/11/the-perceptron-and-all-the-things-it-cant-perceive/

[2] http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-867-machine-learning-fall-2006/lecture-notes/

- The linear classifier can be written as $f_{\boldsymbol{\theta}}(\mathbf{x}) = \text{sign}(\boldsymbol{\theta}^T \mathbf{x}) = \text{sign}(x_1\theta_1 + \ldots + x_d\theta_d)$, and corresponds to a "straight line" (or more generally, hyperplane) passing through the origin:



Motivating example 1 (spam detection):

- Let each $\mathbf{x}_t$ represent an email, and each y_t be $+1$ is it is a spam email (and -1 otherwise).
- For instance, one reasonable representation of an email is $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,d})$ where $x_{t,j}$ is the number of times the j-th word appears.
- For words like "viagra" and "free" we should expect $\theta_j > 0$
- For words like "NUS" and "confirmation" we should expect $\theta_j < 0$

Motivating example 2 (image authentication):

- Let each $\mathbf{x}_t$ represent an image of a face, obtained by arranging all the pixel values into a vector (e.g., $d = 10^4$ for a 100×100 image)
- Let $y_t = 1$ if the person in image $\mathbf{x}_t$ should be allowed entry, and otherwise $y_t = -1$
- (In both this example and the previous, the linearly separable assumption is highly questionable! But don't worry, we will increasingly move away from it throughout the course.)

2 The Perceptron Algorithm

Training error.

- For a given classifier parameter vector $\boldsymbol{\theta}$, define the *training error*

$$\hat{E}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{t=1}^{n} \text{Loss}(y_t, f_{\boldsymbol{\theta}}(\mathbf{x}_t)),$$

where

$$\text{Loss}(y, \hat{y}) = \mathbf{1}\{\hat{y} \neq y\} = \begin{cases} 1 & \hat{y} \neq y \\ 0 & \text{otherwise.} \end{cases}$$

The function $\mathbf{1}\{\cdot\}$ is referred to as the indicator function (1 if the event is true, 0 otherwise).

- The word "training" refers to the fact that we are evaluating on the data set $\mathcal{D} = \{(\mathbf{x}_t, y_t)\}$ that is used for finding $\boldsymbol{\theta}$. Later, we will introduce the notion of "test error", in which we evaluate the error on *different* data that we haven't seen previously (this is what we are ultimately interested in being able to do!)

- The linearly separable assumption means there exists $\boldsymbol{\theta}$ such that $\hat{E}(\boldsymbol{\theta}) = 0$.

Introducing the perceptron update.

- We will present an algorithm that iterates through $\mathcal{D}$ and updates its current estimate of $\boldsymbol{\theta}$
- Specifically, if $\boldsymbol{\theta}$ is the current estimate, and we observe the pair $(\mathbf{x}_t, y_t)$, we do the following:
  - If $y_t = f_{\boldsymbol{\theta}}(\mathbf{x}_t)$, leave $\boldsymbol{\theta}$ unchanged (The classifier is already correct, so don't touch it!)
  - If $y_t \neq f_{\boldsymbol{\theta}}(\mathbf{x}_t)$, update to $\boldsymbol{\theta}_{\text{next}} = \boldsymbol{\theta} + y_t \mathbf{x}_t$
- Reasoning:
  - When we make a mistake (i.e., $y_t \neq f_{\boldsymbol{\theta}}(\mathbf{x}_t)$), it must be that the sign of $\boldsymbol{\theta}^T \mathbf{x}_t$ disagrees with y_t, or equivalently, $y_t \boldsymbol{\theta}^T \mathbf{x}_t < 0$.
  - But if we instead consider the updated classifier, we get

$$\begin{aligned} y_t \boldsymbol{\theta}_{\text{next}}^T \mathbf{x}_t &= y_t (\boldsymbol{\theta} + y_t \mathbf{x}_t)^T \mathbf{x}_t \\ &= y_t \boldsymbol{\theta}^T \mathbf{x}_t + y_t^2 \mathbf{x}_t^T \mathbf{x}_t \\ &= y_t \boldsymbol{\theta}^T \mathbf{x}_t + \|\mathbf{x}_t\|^2, \end{aligned}$$

    so this quantity either becomes "less negative", or even better, shifts to being positive.
  - Clearly, if we apply the update to the *same* pair $(\mathbf{x}_t, y_t)$ over and over, we will eventually classify that sample correctly.
  - But could it be the case that increasing $y_t \boldsymbol{\theta}^T \mathbf{x}_t$ for one sample decreases it for other samples? Could this behavior just go back and forth indefinitely?

Full description of the perceptron algorithm.

1. Initialize $\boldsymbol{\theta}^{(0)}$ to some value (e.g., $\mathbf{0}$), and initialize the index k to 0.
2. Repeatedly perform the following:
   - Select the next example $(\mathbf{x}_t, y_t)$ from the training set[3] and check whether $\boldsymbol{\theta}^{(k)}$ classifies it correctly.
   - If it is incorrect (i.e., $y_t (\boldsymbol{\theta}^{(k)})^T \mathbf{x}_t < 0$), set $\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} + y_t \mathbf{x}_t$ and increment $k \leftarrow k + 1$.

3 Analysis of Convergence and Correctness

Assumptions and theorem statement.

[3] If we reach the end of the training set, we cycle back to $t = 1$. In fact, we don't have to cycle through in order; we could use some other pre-specified order.

- Assumption 1. There exists $R \in (0, \infty)$ such that every input $\mathbf{x}_t$ in $\mathcal{D}$ satisfies $\|\mathbf{x}_t\| \leq R$ (i.e., the input vectors are bounded)

- Assumption 2. There exists a parameter $\boldsymbol{\theta}^*$ and positive constant $\gamma > 0$ such that

$$\min_{t=1,\ldots,n} y_t(\boldsymbol{\theta}^*)^T \mathbf{x}_t \geq \gamma. \tag{1}$$

  This is a "strict" form of the linearly separable assumption.

- Theorem. Under the initial vector $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, for any data set $\mathcal{D}$ satisfying the above assumptions, the perceptron algorithm produces a vector $\boldsymbol{\theta}^{(k)}$ classifying every example correctly after at most

$$k_{\max} = \frac{R^2 \|\boldsymbol{\theta}^*\|^2}{\gamma^2}$$

  update steps, where $\boldsymbol{\theta}^*$, γ and R are defined in the two assumptions.

- Idea of the proof (below):
  - Show that $(\boldsymbol{\theta}^*)^T \boldsymbol{\theta}^{(k)}$ increases *at least* linearly in k. i.e., $\boldsymbol{\theta}^*$ and $\boldsymbol{\theta}^{(k)}$ are "highly correlated"
  - Show that $\|\boldsymbol{\theta}^{(k)}\|^2$ increases *at most* linearly in k. i.e., the "high correlation" just mentioned isn't merely due to $\boldsymbol{\theta}^{(k)}$ growing huge.
  - Deduce that $\boldsymbol{\theta}^*$ and $\boldsymbol{\theta}^{(k)}$ must be close for large enough k

The proof.

- Part one:
  - Observe that

$$\begin{aligned}(\boldsymbol{\theta}^*)^T \boldsymbol{\theta}^{(k+1)} &= (\boldsymbol{\theta}^*)^T (\boldsymbol{\theta}^{(k)} + y_t \mathbf{x}_t) \\ &= (\boldsymbol{\theta}^*)^T \boldsymbol{\theta}^{(k)} + y_t (\boldsymbol{\theta}^*)^T \mathbf{x}_t \\ &\geq (\boldsymbol{\theta}^*)^T \boldsymbol{\theta}^{(k)} + \gamma.\end{aligned}$$

  - Applying this recursively with $\boldsymbol{\theta}^{(0)} = \mathbf{0}$, we obtain

$$(\boldsymbol{\theta}^*)^T \boldsymbol{\theta}^{(k)} \geq k\gamma.$$

- Part two:
  - Recall that updates are only made when a mistake occurs, i.e., for each k the corresponding sample indexed by t gives $\langle \boldsymbol{\theta}^{(k)}, y_t \mathbf{x}_t \rangle \leq 0$. We can then write

$$\begin{aligned}\|\boldsymbol{\theta}^{(k+1)}\|^2 &= \|\boldsymbol{\theta}^{(k)} + y_t \mathbf{x}_t\|^2 &(2)\\ &= \|\boldsymbol{\theta}^{(k)}\|^2 + 2\langle \boldsymbol{\theta}^{(k)}, y_t \mathbf{x}_t \rangle + \|\mathbf{x}_t\|^2 &(3)\\ &\leq \|\boldsymbol{\theta}^{(k)}\|^2 + \|\mathbf{x}_t\|^2. &(4)\end{aligned}$$

– Applying the assumption $\|\mathbf{x}_t\| \leq R$ and recursing (with $\boldsymbol{\theta}^{(0)} = \mathbf{0}$), we obtain

$$\|\boldsymbol{\theta}^{(k)}\|^2 \leq kR^2.$$

- Part three:

  – The famous *Cauchy-Schwarz inequality* states that $\langle \mathbf{v}, \mathbf{w} \rangle \leq \|\mathbf{v}\| \cdot \|\mathbf{w}\|$ for any $\mathbf{v}, \mathbf{w} \in \mathbb{R}^d$. One way to understand this is that $\langle \mathbf{v}, \mathbf{w} \rangle = \|\mathbf{v}\| \cdot \|\mathbf{w}\| \cdot \cos\big(\text{angle}(\mathbf{v}, \mathbf{w})\big)$ and $\cos(a) \in [-1, 1]$.

  – Applying it with $\mathbf{v} = \boldsymbol{\theta}^{(k)}$ and $\mathbf{w} = \boldsymbol{\theta}^*$, we obtain

$$\begin{aligned} 1 &\geq \frac{\langle \boldsymbol{\theta}^{(k)}, \boldsymbol{\theta}^* \rangle}{\|\boldsymbol{\theta}^{(k)}\| \cdot \|\boldsymbol{\theta}^*\|} && (5) \\ &\geq \frac{k\gamma}{\|\boldsymbol{\theta}^*\| \cdot \sqrt{kR^2}} \qquad \text{(by Part 1)} && (6) \\ &= \frac{\sqrt{k}\gamma}{\|\boldsymbol{\theta}^*\| \cdot R} \qquad \text{(by Part 2).} && (7) \end{aligned}$$

  – Re-arranging gives $k \leq \frac{\|\boldsymbol{\theta}^*\|^2 R^2}{\gamma^2}$, i.e., it is impossible for k to go beyond this number of updates. Stated differently, after this many updates, every example *must* be classified correctly.

Non-separable case.

- The "strict" separability assumption (1) is crucial to make the above proof work.
- If the separation is not strict (i.e., $\gamma = 0$), it could take an arbitrarily long time to converge.
- What if the data set is non-separable?
- Extensions. Several variations of the perceptron algorithm exist, some of which are discussed in the supplementary document lec1a.pdf:

  – Variations ensuring a margin at least a constant fraction (e.g., half) of the best possible margin γ

  – Variable increments (i.e., update by $\eta^{(k)} y_t \mathbf{x}_t$ instead of just $y_t \mathbf{x}_t$)

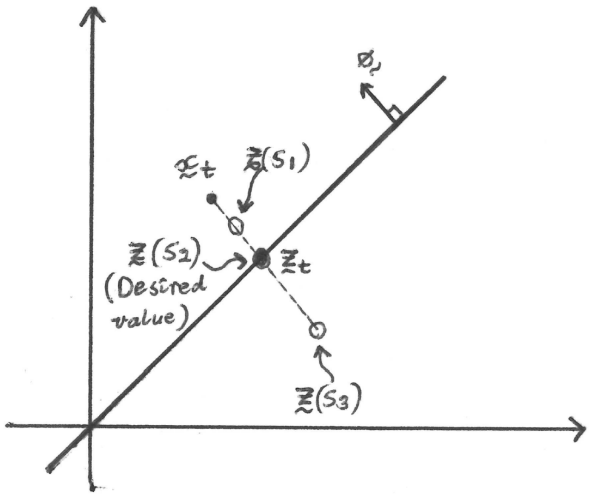
  – Batch updates (i.e., update according to multiple $(\mathbf{x}_t, y_t)$ at once, not just one at a time)

4 Margin and Geometry

- For fixed $\boldsymbol{\theta}$ separating positive from negative samples, the highest possible γ satisfying Eq. (1) is the one such that equality holds: $\gamma = \min_{t=1,\ldots,n} y_t \boldsymbol{\theta}^T \mathbf{x}_t$. Let's look further at this choice.
- Claim: Upon setting $\gamma = \min_{t=1,\ldots,n} y_t \boldsymbol{\theta}^T \mathbf{x}_t$, the quantity $\gamma_{\text{geom}} = \frac{\gamma}{\|\boldsymbol{\theta}\|}$ is the smallest distance from any example $\mathbf{x}_t$ to the decision boundary specified by $\boldsymbol{\theta}$.
- Proof:

  – The decision boundary is the set (hyperplane) of points satisfying $\langle \boldsymbol{\theta}, \mathbf{x} \rangle = 0$

  – The vector $\boldsymbol{\theta}$ points perpendicular to this hyperplane (see the figure below)



- Take a point $\mathbf{x}_t$ and define the vector $\mathbf{z}_t = \mathbf{x}_t - s\frac{y_t\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}$ with s chosen so that $\mathbf{z}_t$ lies on the hyperplane. Then since $\frac{\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}$ has unit norm, s is the distance we are looking for (if we choose t to index the nearest point to the hyperplane).
- Since $\mathbf{z}_t$ lies on the decision boundary hyperplane (which is specified by zero inner product), we have $\boldsymbol{\theta}^T\mathbf{z}_t = 0$, and therefore

$$\begin{aligned} 0 &= y_t\boldsymbol{\theta}^T\mathbf{z}_t \\ &= y_t\boldsymbol{\theta}^T\left(\mathbf{x}_t - s\frac{y_t\boldsymbol{\theta}}{\|\boldsymbol{\theta}\|}\right) \qquad \text{(by definition of } \mathbf{z}_t\text{)} \\ &= y_t\boldsymbol{\theta}^T\mathbf{x}_t - s\|\boldsymbol{\theta}\|. \qquad \text{(since } \boldsymbol{\theta}^T\boldsymbol{\theta} = \|\boldsymbol{\theta}\|^2\text{)} \end{aligned}$$

- If we consider t being the index such achieving the minimum in the definition of γ, we obtain $y_t\boldsymbol{\theta}^T\mathbf{x}_t = \gamma$, and consequently

$$s = \frac{\gamma}{\|\boldsymbol{\theta}\|}$$

as claimed.

- Discussion.
  - We can view $\gamma_{\text{geom}}^{-1}$ as a measure of difficulty (smaller γ_{geom} is harder)
  - The bound $k_{\max}$ in the above theorem for the perceptron algorithm can be expressed as $k_{\max} = \left(\frac{R}{\gamma_{\text{geom}}}\right)^2$. It is not *directly*[4] dependent on the dimension d.
  - It is not *directly*[5] dependent on the number of samples n.

[4]The subtlety is that γ_{geom} depends on the input vectors $\mathbf{x}_t \in \mathbb{R}^d$, so it is unclear how to compare two different d values.
[5]Adding more samples do a data set, even in a way that is sure to preserve linear separability, could decrease γ_{geom}.