Machine Learning - I (CS-DS 864)
Aug–Dec
***Instructor*: Prof. G.Srinivasaraghavan**

Lecture Notes

Perceptrons

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

Contents

List of Figures

List of Tables

1 Bit of Linear Algebra and Geometry

A point in a d-dimensional space is represented as a vector $\mathbf{x}$ — a d-tuple of real values $(x_1, x_2, \ldots, x_d) \in \mathscr{R}^d$ (we would henceforth use boldface symbols such as $\mathbf{x}$ to denote vectors and symbols in normal font to denote scalars). We would often think of a vector a column matrix, by default. Using this convention, $\mathbf{v}^T$ would then denote a row vector.

Given a set S of n points $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ in $\mathscr{R}^d$

1. An *Affine* combination of the points in S is a point of the form

$$\sum_{i=1}^{n} a_i \mathbf{v_i}, \quad a_i \in \mathscr{R} \text{ and } \sum_{i=1}^{d} a_i = 1$$

   The set of all affine combinations of the points in S is called the *Affine Hull* of S. For example the affine hull of any two points in $\mathscr{R}^d$ is just the line passing through the two points. Note that any point $\mathbf{v}$ on the line joining two points $\mathbf{v_1}, \mathbf{v_2}$ can written as

$$\mathbf{v} = \mathbf{v_1} + a.(\mathbf{v_2} - \mathbf{v_1}) = (1-a)\mathbf{v_1} + a\mathbf{v_2}$$

2. A *Convex Combination* is an affine combination with the additional restriction $a_i \geq 0$. The set of all convex combinations of a set of points S is called the *Convex Hull* of S. For example the convex hull of three points in $\mathscr{R}^3$ not on the same line is just the triangle with the three points as the corners.

3. A *hyperplane* in $\mathscr{R}^d$ is a set of points $\mathbf{v}$ that satisfy an equation of the form

$$\sum_{i=1}^{d} a_i.v_i = \mathbf{a}^T.\mathbf{v} = b$$

   for fixed constants $\{a_1, a_2, \ldots, a_d, b\}$. The constant b is often referred to as the *bias*. It is easy to verify that the vector $\mathbf{a}$ is *normal* to the hyperplane. To see this, consider any two points $\mathbf{v_1}, \mathbf{v_2}$ on the hyperplane. We will show that $\mathbf{v_2} - \mathbf{v_1}$ is normal to $\mathbf{a}$.

$$\mathbf{a}^T.(\mathbf{v_2} - \mathbf{v_1}) = \mathbf{a}^T.\mathbf{v_2} - \mathbf{a}^T.\mathbf{v_1} = b - b = 0$$

   Unless otherwise specified we usually assume a normalized representation of the hyperplane — in other words we assume that the vector $\mathbf{a}$ is a unit vector with $||\mathbf{a}||_2 = 1$.

4. Given a hyperplane $\mathbf{a}^T.\mathbf{v} = b$, and a point $\mathbf{u}$, the expression

$$b - \mathbf{a}^T.\mathbf{u}$$

   gives the shortest distance from $\mathbf{u}$ to the hyperplane.
   Proof: Consider the point $\mathbf{v}^*$ at a distance $b - \mathbf{a}^T.\mathbf{u}$ in the direction given by $\mathbf{a}$ from $\mathbf{u}$. Therefore

$$\mathbf{v}^* = \mathbf{u} + (b - \mathbf{a}^T.\mathbf{u})\mathbf{a}$$

   Clearly the distance between $\mathbf{v}^*$ and $\mathbf{u}$ is

$$||\mathbf{v}^* - \mathbf{u}||_2 = (b - \mathbf{a}^T.\mathbf{u})||\mathbf{a}||_2 = (b - \mathbf{a}^T.\mathbf{u})$$

   We use the Euclidean norm $||\mathbf{v}||_2$ throughout this discussion — so will omit the subscript 2 for the norms in the rest of this document. We now show that

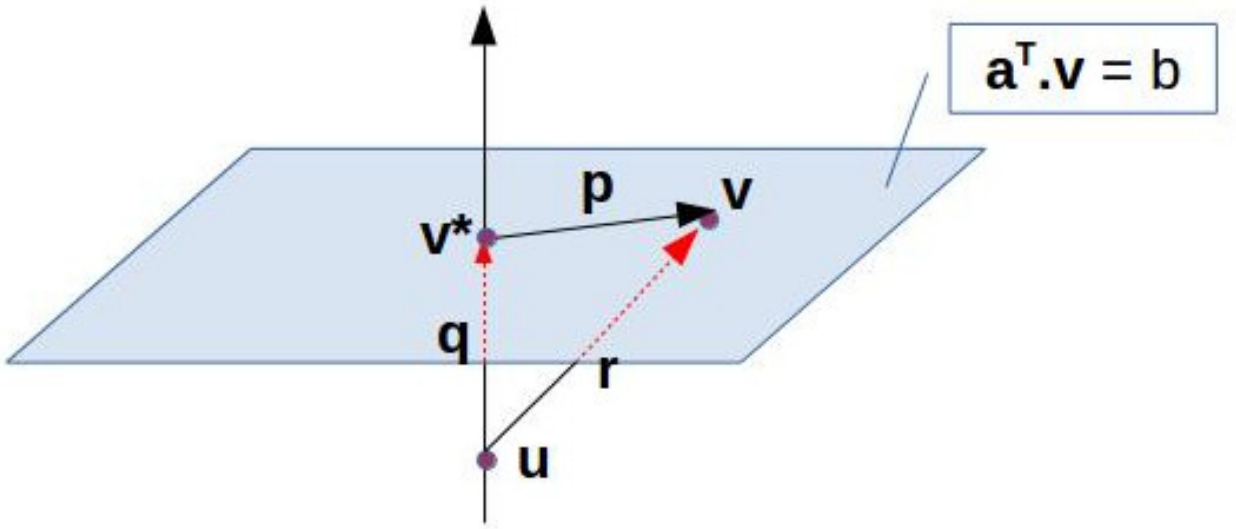

Figure 1: Distance from a Hyperplane

(a) $\mathbf{v}^*$ **lies on the hyperplane.** For this we need to show that $\mathbf{a}^T.\mathbf{v}^* = b$. Indeed

$$\begin{aligned}\mathbf{a}^T.\mathbf{v}^* &= \mathbf{a}^T.(\mathbf{u} + (b - \mathbf{a}^T.\mathbf{u})\mathbf{a}) \\ &= \mathbf{a}^T.\mathbf{u} + (b - \mathbf{a}^T.\mathbf{u})\mathbf{a}^T\mathbf{a} \\ &= \mathbf{a}^T.\mathbf{u} + b - \mathbf{a}^T.\mathbf{u} \text{ (because } \mathbf{a}^T\mathbf{a} = 1) \\ &= b\end{aligned}$$

(b) $\mathbf{v}^*$ **is the point on the hyperplane closest to u**. Let $\mathbf{v} \neq \mathbf{v}^*$ be some other point on the hyperplane. Let

$$\mathbf{p} = (\mathbf{v} - \mathbf{v}^*), \ \mathbf{q} = (\mathbf{v}^* - \mathbf{u}), \ \mathbf{r} = (\mathbf{v} - \mathbf{u}) \text{ and } \mathbf{r} = \mathbf{p} + \mathbf{q}$$

Clearly then $||\mathbf{p}|| = ||\mathbf{v} - \mathbf{v}^*|| > 0$. We also know from the construction of $\mathbf{v}^*$ that $\mathbf{p}^T.\mathbf{q} = 0$. We need to show that $||\mathbf{r}|| > ||\mathbf{q}||$. Now

$$\begin{aligned}||\mathbf{r}|| = \mathbf{r}^T.\mathbf{r} &= (\mathbf{p} + \mathbf{q})^T.(\mathbf{p} + \mathbf{q}) \\ &= \mathbf{p}^T.\mathbf{p} + \mathbf{p}^T.\mathbf{q} + \mathbf{q}^T.\mathbf{p} + \mathbf{q}^T.\mathbf{q} \\ &> ||\mathbf{q}||\end{aligned}$$

■

5. It is often convenient to add an extra dimension to make the representation of hyperplanes homogeneous. The original domain of $\mathscr{R}^d$ is 'lifted' to an additional dimension by adding an extra component with a constant 1 to every point in $\mathscr{R}^d$. Thus for every point $\mathbf{v} \in \mathscr{R}^d$, the corresponding point with homogeneous coordinates will be $\mathbf{v}' = (1, v_1, v_2, \ldots, v_d) \in \{1\} \times \mathscr{R}^d$. Henceforth we denote the space $\{1\} \times \mathscr{R}^d$ as $\mathbb{R}^d$. This enables us to write the equation of a hyperplane as $\mathbf{a}^T.\mathbf{v}' = 0$. Also the distance of a point $\mathbf{u}$ in homogeneous coordinates from the hyperplane will be $|\mathbf{a}^T.\mathbf{u}|$ – the absolute value

of ($\mathbf{a}^T.\mathbf{u}$). Henceforth we will assume homogeneous coordinates going forward, unless specified otherwise. Note that it is enough to specify the coefficient vector $\mathbf{a}$ to specify a hyperplane. We denote the hyperplane with coefficients $\mathbf{a}$ as $H_{\mathbf{a}}$.

6. Given a hyperplane $H_{\mathbf{a}}$ the sets of points

$$H_{\mathbf{a}}^{+} = \{\mathbf{u} | \mathbf{a}^T.\mathbf{u} > 0\} \text{ and } H_{\mathbf{a}}^{-} = \{\mathbf{u} | \mathbf{a}^T.\mathbf{u} < 0\}$$

are said to be the positive and negative *half-spaces* respectively into which $H_{\mathbf{a}}$ divides $\mathbb{R}^d$.

2 Binary Classification Problems

Given a set of data points each with a two-valued (say $+1$ and -1) attribute (called the *label*) that depends on the other attributes (called the *features*) of the data object, the Binary Classification problem is the problem of arriving at a model that can correctly categorize the data points into $+1$ and -1 based on the features alone. In other words the model can predict the label given the features of an object.

A machine learning approach to the binary classification problem is to first train a model (from a class of candidate model hypotheses) on a training data set that has the label specified for each point in the dataset along with the features. The model arrived at after training is then used to predict the label of any new data object that is encountered. Several real-life problems can be formulated as binary classification problems. We give a few representative examples below.

1. A credit card company wants to determine if an applicant for a credit card is likely to be a defaulter or not based on the person's attributes (features) such as credit history, annual income, age and gender. Every potential customer is a data point. The label is $+1$ if the person is a potential defaulter and -1 otherwise.

2. An algorithm for medical diagnosis needs to distinguish MRI images of malignant tissues from those of normal tissues. The image characteristics are the features and the malignancy is the value. Every MRI image of the tissue involved is a data point. The label is $+1$ if the image represents a malignant tissue and -1 otherwise.

3. A spam filter needs to distinguish normal emails from spam. Each email is a data point. The features are derived from the text, subject and sender of the email. The label is a $+1$ if the email is categorized as spam and -1 otherwise.

4. An early warning system for a critical installation needs to decide if there is an impending breakdown / disaster. The data received from the sensors in the installation determine the features and the label is a $+1$ if the sensor data indicates an impending breakdown and -1 otherwise.

Binary Classification problems are often posed as Separability Problems where each data point is thought of as a feature vector $\mathbf{x}_i \in \mathscr{R}^d$ in the feature space along with a label $y_i \in \pm 1$

associated with the data point. The classification problem is then to find a model $M(\mathbf{v}) : \mathscr{R}^d \rightarrow \pm 1$ such that

$$\forall \mathbf{x}_i : M(\mathbf{x}_i) = y_i$$

A special case of binary classification, called *Linear Classification* is when the model is a hyperplane $H_{\mathbf{w}}$ with

$$M(\mathbf{v}) = sign(\mathbf{w}^T.\mathbf{v}), \;\; sign(t) = \begin{cases} +1 & t > 0 \\ -1 & t < 0 \end{cases}$$

2.1 Linear Separability

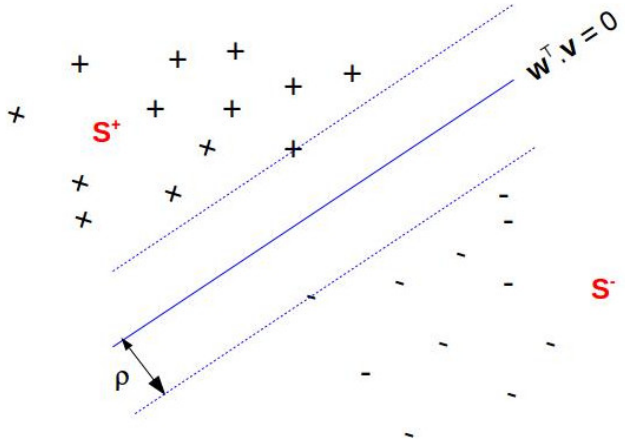

Figure 2: Linear Separability

Given two sets of points S^+ and S^- as in Figure 2, they are said to be linearly separable if there exists a hyperplane $H_{\mathbf{w}}$ in $\mathbb{R}^d$ such that $S^+ \subset H^+_{\mathbf{w}}$ and $S^- \subset H^-_{\mathbf{w}}$.

Consider a dataset S of points where each point $\mathbf{x}_i$ has been labelled $y_i \in \pm 1$. Let S^+ be the subset of points with label $+1$ and S^- the subset with label -1. The dataset is said to be separable if S^+ and S^- are separable. The following property of a separating hyperplane $H_{\mathbf{w}^*}$ is easy to establish:

$$\forall \mathbf{x}_i \in S : y_i \mathbf{w}^{*T}.\mathbf{x}_i > 0 \tag{1}$$

Conversely for a hyperplane $H_{\mathbf{w}}$ (not passing through any of the points in S), a point $\mathbf{x}_j \in S^+$ is such that $\mathbf{x}_j \notin H^+_{\mathbf{w}}$ or $\mathbf{x}_j \in S^-$ but $\mathbf{x}_j \notin H^-_{\mathbf{w}}$, if and only if

$$y_j.\mathbf{w}^T.\mathbf{x}_j < 0 \tag{2}$$

We say a hyperplane $H_{\mathbf{w}}$ *correctly classifies* a point $\mathbf{x}_i$ if $y_i \mathbf{w}^T.\mathbf{x}_i > 0$.

3 Perceptron Classification

In this section we present the classical algorithm due to Rosenblatt for finding a linear classifier for a linearly separable dataset. The algorithm works by initializing the hyperplane to an arbitrarily chosen coefficient vector and iteratively correcting it whenever a misclassified point (by

the current hyperplane) is found. We show that this algorithm is guaranteed to converge to a valid separator whenever the dataset is separable. We will discuss issues about generalizability — how successful the model will be when used to classify new data points that were not part of the training dataset — of the model produced by the Perceptron algorithm later in the course. The Perceptron algorithm is described here only as a motivational example to illustrate the fact that in spite of considerable conceptual difficulties, 'learning' in some sense is indeed possible. Perceptron theory in fact also forms the conceptual foundation for Support Vector Machines (SVM) that we will discuss later in the course.

The Perceptron algorithm is given below. The input to the algorithm is a dataset $\{(\mathbf{x}_i, y_i) \in (\mathbb{R}^d, \pm 1) \mid i = 1, \ldots, n\}$. The output is a vector of coefficients $\mathbf{w}^*$ (often called the *weight* vector) for a separating hyperplane.

Algorithm 1 Perceptron — Primal Form

1: Initialize $t \leftarrow 0$, $\mathbf{w}_t \leftarrow$ Random Vector.
2: **while** $\exists i_t$ such that $y_{i_t} \mathbf{w}_t^T . \mathbf{x}_{i_t} < 0$ **do**
3: $\quad \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{i_t} . \mathbf{x}_{i_t}$
4: $\quad t \leftarrow t + 1$
5: **end while**
6: $\mathbf{w}^* \leftarrow \mathbf{w}_{t-1}$

Theorem 1 *For any separable dataset $S = \{(\mathbf{x}_i, y_i) \in (\mathbb{R}^d, \pm 1) \mid i = 1, \ldots, n\}$, the perceptron algorithm converges to a valid separator in a finite number of steps.*

Proof: Since the dataset is separable, let $H_{\mathbf{w}^*}$ be the separating hyperplane. Let the candidate hyperplane in the t^{th} iteration of the algorithm be $H_{\mathbf{w}_t}$. For this proof we do not assume that $\mathbf{w}^*, \mathbf{w}_t$ are unit vectors. Also let R be the length of the vector representing the point farthest from the origin

$$R = \max_{\mathbf{x}_i \in S} ||\mathbf{x}_i|| \tag{3}$$

We establish the following at every iteration of the perceptron algorithm.

1. $||\mathbf{w}_t||^2 \leq tR^2 + c_0$, where $c_0 = ||w_0||^2$ is a constant (the length of the initial coefficient vector $\mathbf{w}_0$). We use the fact that for any two vectors $\mathbf{a}$ and $\mathbf{b}$, the inner product $\mathbf{a}^T.\mathbf{b}$ is commutative i.e. $\mathbf{a}^T.\mathbf{b} = \mathbf{b}^T.\mathbf{a}$.

$$\begin{aligned} ||\mathbf{w}_t||^2 &= \mathbf{w}_t^T.\mathbf{w}_t = (\mathbf{w}_{t-1} + y_{i_{t-1}}\mathbf{x}_{i_{t-1}})^T.(\mathbf{w}_{t-1} + y_{i_{t-1}}\mathbf{x}_{i_{t-1}}) \\ &= ||\mathbf{w}_{t-1}||^2 + y_{i_{t-1}}^2.||\mathbf{x}_{i_{t-1}}||^2 + 2y_{i_{t-1}}(\mathbf{w}_{t-1}^T.\mathbf{x}_{i_{t-1}}) \\ &\leq ||\mathbf{w}_{t-1}||^2 + R^2 \\ &\leq tR^2 + c_0 \end{aligned}$$

The first inequality follows from Equation 2, the fact that $H_{\mathbf{w}_{t-1}}$ does not classify $\mathbf{x}_{i_{t-1}}$ correctly (that's how $\mathbf{x}_{i_{t-1}}$ was chosen) and Equation 3. Extending the recurrence gives us the second inequality.

2. $\mathbf{w}_t^T.\mathbf{w}^* \geq t.\rho + c_1$ for some constants $\rho > 0$ and $c_1 = \mathbf{w}_0^T.\mathbf{w}^*$.

$$\begin{aligned}\mathbf{w}_t^T.\mathbf{w}^* &= \mathbf{w}_{t-1}^T.\mathbf{w}^* + y_{i_{t-1}}\mathbf{x}_{i_{t-1}}^T.\mathbf{w}^* \\ &\geq \mathbf{w}_{t-1}^T.\mathbf{w}^* + \rho \\ &\geq t\rho + c_1\end{aligned}$$

where $\rho = \min_{\mathbf{x_i} \in S} y_i \mathbf{x}_i^T.\mathbf{w}^* = \min_{\mathbf{x_i} \in S} y_i \mathbf{w}^{*T}.\mathbf{x}_i$. Note that $\rho' = \frac{\rho}{||\mathbf{w}^*||}$ is the distance from the separating hyperplane $H_{\mathbf{w}^*}$ of the point in S that is closest to it, as shown in Figure 2. The quantities ρ' and ρ are often called the *Geometric Margin* and the *Functional Margin* respectively, of the separator $H_{\mathbf{w}^*}$ with respect to the dataset S. The inequality follows from Equation 1 and the final lower bound is a straightforward extension of the recurrence.

Putting the above two together we get

$$\begin{aligned}(\sqrt{t}R + c_0).||\mathbf{w}^*|| &\geq ||\mathbf{w}_t||.||\mathbf{w}^*|| \\ &\geq \mathbf{w}_t^T.\mathbf{w}^* \\ &\geq t\rho + c_1 \\ \sqrt{t}R.||\mathbf{w}^*|| &\geq t\rho + c_1 - c_0.||\mathbf{w}^*|| \\ t^2\rho^2 + \alpha.t + \beta^2 \leq 0, \quad &\text{where } \beta = (c_1 - c_0||\mathbf{w}^*||), \alpha = 2\rho.\beta - R^2.||\mathbf{w}^*||^2 \\ t \leq \frac{R^2.||\mathbf{w}^*||^2}{\rho^2} &= \left(\frac{R}{\rho}\right)^2 ||\mathbf{w}^*||^2\end{aligned}$$

To verify that this bounds the value of t we only have to observe that

1. the above quadratic is convex (second derivative is $2\rho^2 > 0$) and hence is an 'upward facing cup' and the values of t for which the quadratic will evaluate to < 0 are the ones between the two roots of the quadratic, and
2. the roots of the quadratic are well defined (finite) — this is easily checked by verifying that the discriminant of the quadratic is positive

$$\begin{aligned}\alpha^2 - 4\rho^2.\beta^2 &= R^4||\mathbf{w}^*||^4 + 4\rho.R^2||\mathbf{w}^*||^2.(c_0||\mathbf{w}^*|| - c_1), \quad \text{(Using definitions for } \alpha, \beta) \\ &= R^4||\mathbf{w}^*||^4 + 4\rho.R^2||\mathbf{w}^*||^2.\left(||\mathbf{w}_0||^2.||\mathbf{w}^*|| - \mathbf{w}_0^T.\mathbf{w}^*\right), \quad \text{(Using definitions for } c_0, c_1) \\ &\geq R^4||\mathbf{w}^*||^4 + 4\rho.R^2||\mathbf{w}^*||^2.\left(||\mathbf{w}_0||^2.||\mathbf{w}^*|| - ||\mathbf{w}_0||.||\mathbf{w}^*||\right), \quad \left(||\mathbf{w}_0||.||\mathbf{w}^*|| \geq \mathbf{w}_0^T.\mathbf{w}^*\right) \\ &= R^4||\mathbf{w}^*||^4 + 4\rho.R^2.||\mathbf{w}_0||.||\mathbf{w}^*||^3.(||\mathbf{w}_0|| - 1)\end{aligned}$$

It is clear from the above derivation that selecting any $\mathbf{w}_0$ for which the length $||\mathbf{w}_0||$ is at least 1 will guarantee that the discriminant is positive, giving us well defined values for the roots of the quadratic. Clearly the positive square root of the discriminant is $R^2.||\mathbf{w}^*||^2 + \delta$ for some $\delta \geq 0$. Therefore

$$t \leq \frac{-\alpha + R^2.||\mathbf{w}^*||^2 + \delta}{\rho^2} \leq \left(\frac{R}{\rho}\right)^2 ||\mathbf{w}^*||^2 + \frac{\delta}{\rho^2}$$

This gives a finite bound on the number of iterations for the Perceptron Algorithm before it converges to the hyperplane that correctly classifies all the points in S. Note that all the parameters on the RHS of the final inequality are known quantities that depend only on the dataset given and any one (assumed) valid separator $\mathbf{w}^*$. Incidentally this is a more rigorous derivation of the bound that we derived in the class where we assumed $c_0 = c_1 = 0$ for convenience. ■

It may be worthwhile highlighting a few observations above the perceptron and its convergence.

1. Notice that the upper bound proof for the convergence of the perceptron did not assume anything about the order in which the misclassified points $\mathbf{x}_{i_t}$ were chosen for the update. Notice that in Step 2 of the algorithm we could pick any i for which $y_{i_t}\mathbf{w}_t^T.\mathbf{x}_{i_t} < 0$. If no such i_t exists (this means all the points are correctly classified) then the current $\mathbf{w}_t$ is the required separator. However when there is more than one such point, any of those could be picked for the update. Convergence would happen in any case, irrespective of the order in which the misclassified points are picked. Yes of course the number of steps in which the algorithm actually converges would indeed depend on the order in which the points are picked.

2. The same point could get picked up multiple times during the algorithm for the update. Any data point could potentially toggle several times between being a correctly classified point and an incorrectly classified point.

3. In general a separable dataset can have several valid separators. The perceptron algorithm will output one of the separators. The separator produced would change with the order in which the misclassified input data points are picked up for update and the choice of the initial $\mathbf{w}_0$.

Exercise 1 *Implement the Perceptron Algorithm. Generate datasets of various sizes and dimensions along with a known separating hyperplane. Observe the following:*

1. *Check how close the output $\mathbf{w}^*$ is to the known separator.*

2. *Measure the running time (in terms of the number of iterations the algorithm takes to find the separator) of the perceptron algorithm.*

3. *Plot the number of iterations taken by the Perceptron for convergence as a function of the (i) the number of points in the dataset, (ii) the dimensionality of the dataset, (iii) known separation between the two datasets.*

4. *Compare the number of iterations observed in practice against the theoretical bound of Theorem 1.*

Exercise 2 *Consider this variation of the Perceptron algorithm. The update step has a constant* learning rate $\eta > 0$ *which controls the rate at which the candidate separator gets corrected. The modified update step would look like*

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y_{i_t}\mathbf{x}_{i_t}$$

Show that the learning rate has no effect on the upper bound on the number of steps required for the perceptron to halt.

Exercise 3 *Redo the proof of convergence of a perceptron if implemented in $\mathscr{R}^d$ instead of homogeneous coordinates $\mathbb{R}^d$. Explain the extra constant factor that you get in the upper bound on the number of iterations.*

Exercise 4 *Show that for every $t > 0$, $\mathbf{w}_t$ does a 'better' job of classifying $\mathbf{x}_{i_{t-1}}$ than $\mathbf{w}_{t-1}$.*

Exercise 5 *How would the analysis of the perceptron change if we introduced an additional step in the algorithm to ensure that the weight vector $\mathbf{w}$ is always a unit vector — renormalize $\mathbf{w}_t$ after every update?*

The perceptron algorithm has several problems though, some of which are listed below:

1. The algorithm can take a long time to terminate.
2. If the dataset is not separable then the algorithm will not converge. Also non-convergence is hard to detect.
3. The separator output by the algorithm is not 'unique'. The output depends on the sequence in which the points are 'corrected'.

4 Dual Formulation of the Perceptron Algorithm

The Perceptron algorithm has just one simple update step that adds a signed copy of one of the misclassified points to the current $\mathbf{w}$. So starting with the zero vector as the initial $\mathbf{w}$, the final $\mathbf{w}^*$ can in fact be thought of as the weighted sum

$$\mathbf{w}^* = \sum_{i=1}^{n} a_i.y_i.\mathbf{x}_i$$

where a_i denotes the number of times the point $\mathbf{x}_i$ was picked up during the course of the algorithm for updating $\mathbf{w}$. With this insight we can now recast the perceptron algorithm as one that tries to find a_i — the number of times the signed copy of each point in S needs to be added to $\mathbf{w}$ for it to eventually become a separator. The perceptron algorithm can now be rewritten entirely in terms of the dual variables a_i as shown in Algorithm 2.

Algorithm 2 Perceptron — Dual Form

```
Initialize t ← 0, a ← 0 ∈ R^n.
while ∃ i_t such that y_{i_t} Σ_{j=1}^{n} a_j.y_j.x_j^T.x_{i_t} < 0 do
    a_{i_t} ← a_{i_t} + 1
    t ← t + 1
end while
Return a* ← a
```

Notice that the dual space is of dimension n - the size of the dataset S. Again a few observations about the dual formulation are in order:

1. The number of times a data point got added to the weight vector is an indication of how 'difficult' it was to 'learn' that data point. The other way to interpret this is that the dual variable a_i gives an indication of tbe amount of 'information' the data point $\mathbf{x}_i$ carries for the learning. The hard cases (those with a large a_i) are the ones that force you to refine your model repeatedly and hence are contributing more to the learning. In that sense it is these points that carry more 'information'.

Exercise 6 *Show that the perceptron algorithm in its dual form is correct.*

Exercise 7 *The output of the dual form of the perceptron algorithm is a vector* $\mathbf{a}^* \in \mathscr{R}^n$. *Show how* $\mathbf{a}^*$ *will be used to classify any new data point* $\mathbf{x} \in \mathbb{R}^d$.

Exercise 8 *Formulate the dual form of the perceptron algorithm as a linear program. Specify the linear constraints and the objective function, the optimal solution to which will be the 'best'* $\mathbf{a}^*$.

5 Multiclass Discrimination

The perceptron algorithm can be extended to deal with data classification into more than two labels. In this case every data point is classified into a class represented by a label from a finite set $\{y_1, \ldots, y_k\}$ of k labels.

The idea for multiclass discrimination comes from the following observation about binary classification. Suppose $\mathbf{w} \in \mathbb{R}^d$ is a separator between two classes of points S^+ and S^- where $\forall \mathbf{x} \in S^+$, $\mathbf{w}^T.\mathbf{x} > 0$ and $\forall \mathbf{x} \in S^-$, $\mathbf{w}^T.\mathbf{x} < 0$. Another way to look at this is that there are two hyperplanes $\mathbf{w}_+$ and $\mathbf{w}_-$ where $\mathbf{w} = \mathbf{w}_+ - \mathbf{w}_-$ and

$$\forall \mathbf{x} \in S^+ : \mathbf{w}_+^T.\mathbf{x} > \mathbf{w}_-^T.\mathbf{x}, \text{ and } \forall \mathbf{x} \in S^- : \mathbf{w}_+^T.\mathbf{x} < \mathbf{w}_-^T.\mathbf{x}$$

Therefore if we had the models $\mathbf{w}_+$ and $\mathbf{w}_-$ and we had to classify a new data point $\mathbf{z}$ we would do it as follows: compute both $\mathbf{w}_+^T.\mathbf{z}$ and $\mathbf{w}_-^T.\mathbf{z}$ and label the point as

$$\begin{cases} +1 & \text{if } \mathbf{w}_+^T.\mathbf{z} = \max(\mathbf{w}_+^T.\mathbf{z}, \mathbf{w}_-^T.\mathbf{z}) \\ -1 & \text{otherwise} \end{cases}$$

The $\mathbf{w}$ we arrived at earlier using a perceptron is a *separator*. The $\mathbf{w}_+, \mathbf{w}_+$ referred to above would be called *discriminators* — each plays the role of discriminating one class from the rest.

Exercise 9 *Show how you can tweak the perceptron algorithm to directly update* $\mathbf{w}_+$ *and* $\mathbf{w}_-$ *incrementally as appropriate. Finally when the algorithm terminates* $\mathbf{w}_+$ *and* $\mathbf{w}_-$ *should be the correct discriminators for the two classes respectively. Also visualize/interpret* $\mathbf{w}_+$ *and* $\mathbf{w}_-$ *geometrically.*

This observation can now be extended to a finite set of $\{y_1, \ldots, y_k\}$ of k labels. Given a k sets of data points $S_1, \ldots, S_k \subset \mathbb{R}^d$ the objective is to find k 'separator' models $\mathbf{w}_1, \ldots, \mathbf{w}_k$ where $y_i = c(\mathbf{w}_i)$ is the label associated with set S_i and model $\mathbf{w}_i$ such that

$$\forall \mathbf{x} \in S_i \ : \ c\left(\arg\max_j \{\mathbf{w}_j^T . \mathbf{x}\}\right) = y_i$$

One interpretation of the above is that we need to find a set of hyperplanes such that each point is associated with the hyperplane that is farthest from it.

Exercise 10 *Extend the 2-class perceptron algorithm to the multiclass scenario. Can you also prove a similar bound to establish that even in the multiclass case the extended perceptron algorithm will indeed halt after a finite bounded number of steps? VIsualize/characterize the discriminators for the multiclass case.*

Exercise 11 *Implement your multiclass perceptron algorithm. See if there is a significant difference in the error bounds achieved with a given dataset size between the 2-class case and the multiclass case? How does the error rate depend on the number of classes k, if at all?*