CS 217: Artificial Intelligence and Machine Learning **9th Feb 2024**

Lecture 9: Perceptron Classifier Models

Lecturer: Swaprava Nath *Scribe(s): SG17,SG18*

Disclaimer: *These notes aggregate content from several texts and have not been subjected to the usual scrutiny deserved by formal publications. If you find errors, please bring to the notice of the Instructor.*

This lecture covers perceptrons - their training algorithm and proof of convergence, in detail. Towards the end, we also touch on the topic of decision trees.

9.1 Introduction to perceptron models

Formally, perceptrons are a linear deterministic binary classification method.

They use a linear function of the form $\mathbf{w}^\intercal\mathbf{x}$ to perform binary classification. Since it is a binary classifier, it segregates data into two classes, which we label **+1** and **-1** for ease of notation.

The perception outputs a prediction $\tilde{y}_i$ as:

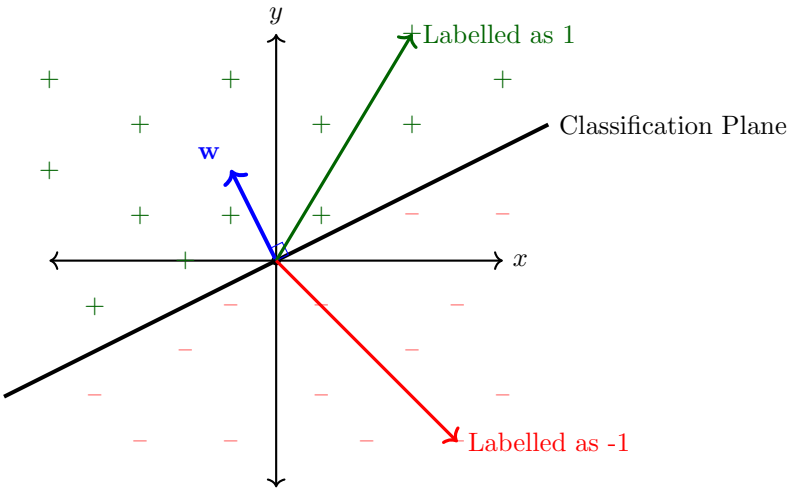
$$\tilde{y}_i = f_{\mathbf{w}}(\mathbf{x}_i) = sgn(\mathbf{w}^\intercal\mathbf{x}_i)$$

Here $sgn(x)$ is the *signum* function, which returns 1 if x is positive or 0, and -1 if negative.

Note that for ease of notation, the vector $\mathbf{x}$ contains the input data augumented by 1 as its 0^{th} component (the linear function also contains a constant intercept). It is hence a vector with $d+1$ dimensions where d is the number of features of the input.

In a perceptron model, we consider the hyperplane in $d+1$-dimensional space with normal vector $\mathbf{w}$ (referred to as the **classification plane**), and classify instances of $\mathbf{x}$ based on which side of the plane they lie on.

Hence, an input $\mathbf{x}_i$ is labelled **+1** if $\mathbf{w}^\intercal\mathbf{x}_i \geq 0$ and labelled **-1** if $\mathbf{w}^\intercal\mathbf{x}_i < 0$. Here's a visual representation of the same.



9.2 Finding correct parameter w for a perceptron

We follow a simple algorithm to find a $\mathbf{w}$ that performs said classification correctly:

1. Start with a random vector $\mathbf{w}_0$ (generally initialized to $\mathbf{0}$ for simplicity)
2. Randomly select an instance $(\mathbf{x}_i, y_i)$ from the dataset
3. If the predicted label for this datapoint is wrong, update $\mathbf{w}$ as given in pseudocode to correct the misclassification. Note that due to our model, the data instance $\mathbf{x}_i$ is misclassified if $y_i \cdot \mathbf{w}^\intercal \mathbf{x}_i < 0$.
4. Repeat steps 2 and 3 until there are no updates in $\mathbf{w}$ for a continiuous fixed number of iterations or the number of iterations crosses a predicided threshold.

Algorithm 1: Finding suitable parameter $\mathbf{w}$

$\mathbf{w}_0 = \mathbf{0}$
for $t = 0 \ldots$ *maxIter* **do**
  $i = \text{random}\{1, 2, \ldots, n\}$
  if $y_i \cdot \boldsymbol{w}^\intercal \boldsymbol{x}_i \leq 0$ **then**
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_i \cdot \mathbf{x}_i$
  else
    $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t$
  end
end

Equivalently, we could replace the conditional statement in step 3 by updating on *every* datapoint as given below which changes $\mathbf{w}$ only when the predicted output is incorrect.

Algorithm 2: Finding suitable weight $\mathbf{w}$

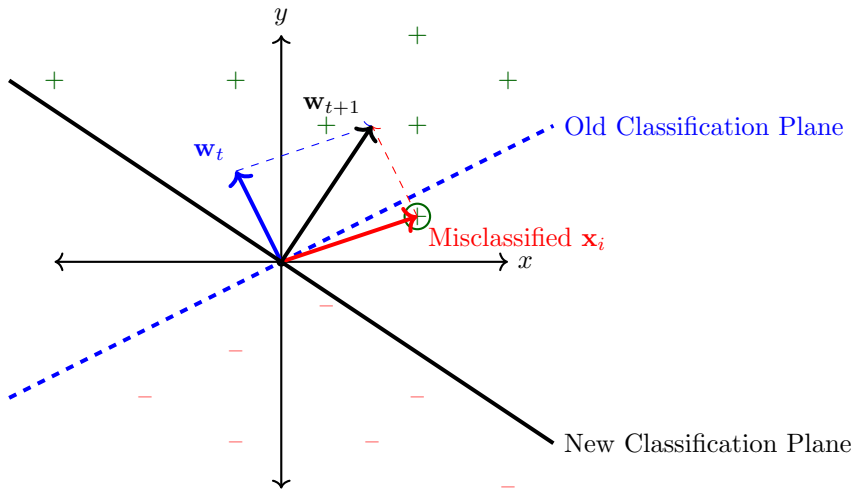
$\mathbf{w}_0 = 0$
for $t = 0 \ldots$ *maxIter* **do**
  $i = \text{random}\{1, 2, \ldots, n\}$
  $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \frac{1}{2} \cdot (y_i - \text{sgn}(y_i \cdot \mathbf{w}^\intercal \mathbf{x}_i)) \cdot \mathbf{x}_i$
end

9.2.1 Remarks on the algorithm

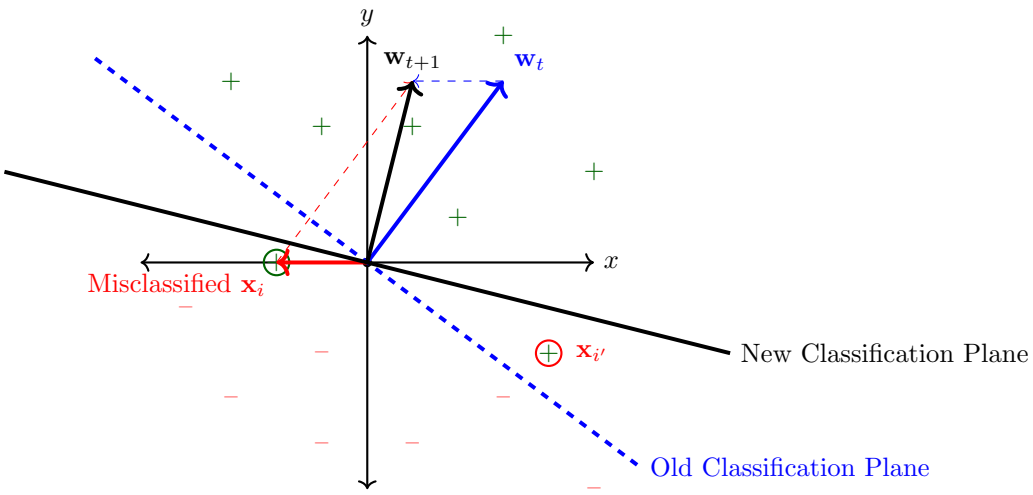
Classification algorithms we have studied in the past, including Linear and Logistic regression are called *batch algorithms* as they require the whole dataset to be processed together.

The perceptron algorithm however is an *online algorithm* which uses the data points one-by-one. Every update tilts the classification plane in such a way that it resolves, or atleast reduces the margin of misclassification, of the particular data point.

A visual representation of the algorithm is shown ahead.



Here, on setting $\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{x}_i$, we can see that the perceptron model corrected the classification plane so that $\mathbf{w}_{t+1}$ correctly predicts $\mathbf{x}_i$. However, that might not always be the case.



This time, on correcting $\mathbf{w}_t$ for the misclassified point, the new $\mathbf{w}_{t+1}$ still fails to correctly classify $\mathbf{x}_i$. Furthermore, it misclassifies an additional point $\mathbf{x}_{i'}$ which is clearly undesirable. Note that this particular dataset is not *linearly separable* (by observation) and hence, any linear perceptron model will misclassify **atleast one** point even with corrections applied to $\mathbf{w}$.

Hence, a step involving $\mathbf{x}_i$ may not necessarily result in a parameter $\mathbf{w}$ that resolves the misprediction, but it still change the value $\mathbf{w}^\intercal \mathbf{x}_i$ in the correct direction. The proof of the same is as follows.

Proof. Consider a misclassified example $(\mathbf{x}_i, y_i)$; i.e. $\text{sgn}(\mathbf{w}_t^\intercal \mathbf{x}_i) \neq y_i$.

We will show how the algorithm improves the value of $y_i \mathbf{w}_{t+1}^\intercal \mathbf{x}_i$, making it **more positive** with every update.

On updating $\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \cdot \mathbf{x}_i$,

$$\begin{aligned} y_i \mathbf{w}_{t+1}^\intercal \mathbf{x}_i &= y_i \cdot ((\mathbf{w}_t + y_i \mathbf{x}_i)^\intercal \mathbf{x}_i) \\ &= y_i \mathbf{w}_t^\intercal \mathbf{x}_i + y_i^2 \mathbf{x}_i^\intercal \mathbf{x}_i \\ &= y_i \mathbf{w}_t^\intercal \mathbf{x}_i + \|\mathbf{x}_i\|^2 \end{aligned}$$

As $\|\mathbf{x}_i\|^2 \geq 0$,

$$y_i \mathbf{w}_{t+1}^\intercal \mathbf{x}_i \geq y_i \mathbf{w}_t^\intercal \mathbf{x}_i$$

Since we made the value of $y_i \mathbf{w}^\intercal \mathbf{x}_i$ more positive, it is more likely that $y_i \mathbf{w}^\intercal \mathbf{x}_i \geq 0$ and we have succeeded in improving the model. □

9.2.2 Summary

We have shown how perceptron is a *mistake-driven online learning algorithm*. Untill now, we have shown how updates are made to the $\mathbf{w}$ parameter of the perceptron to make corrections in the classification of our data points.

We will now prove that this algorithm is guaranteed to converge for *linearly separable* training data.

9.3 Proof of convergence

Theorem 9.1. *If a dataset with $||\boldsymbol{x}_i||_2 < 1 \ \forall i$ is linearly separable, that is, there exists a vector $\boldsymbol{w}^*$ with margin of separation γ then the perceptron algorithm always finds a correct solution (not necessarily the same solution) in at most $\frac{1}{\gamma^2}$ updates.*

Proof. We define linear separability as the existence of a hyperplane that divides the input space into two half-spaces, such that all points of one class lie in one half-space and those of the other class lie in the other half-space. Hence,

$$\exists \ \mathbf{w}^* \text{ such that } (y_i \cdot \mathbf{w}^{*\intercal} \mathbf{x}_i) \geq 0 \ \forall \ (\mathbf{x}_i, y_i) \in \mathcal{D}$$

We define the margin of separation as $\gamma = \min_i \{|\mathbf{w}^{*\intercal} \mathbf{x}_i|\} = \min_i \{\|\mathbf{w}^*\| \cdot \|\mathbf{x}_i\| \cdot \cos\theta_i\}$.

We need to track two quantities:

- $\mathbf{w}_i^\intercal \mathbf{w}^*$
- $\|\mathbf{w}_t\|_2^2$

Claim 1: After every update, $\mathbf{w}_i^\intercal \mathbf{w}^*$ increases by at least γ.

$$\begin{aligned} \mathbf{w}_{t+1} \mathbf{w}^* &= (\mathbf{w}_t + y_i \mathbf{x}_i) \mathbf{w}^* \\ &= \mathbf{w}_t \mathbf{w}^* + y_i \mathbf{x}_i^\intercal \mathbf{w}^* \end{aligned}$$

Since $y_i \mathbf{x}_i^\intercal \mathbf{w}^* \geq \gamma$, we have $\mathbf{w}_{t+1} \mathbf{w}^* - \mathbf{w}_t \mathbf{w}^* \geq \gamma$.

Claim 2: $\|\mathbf{w}_t\|_2^2$ increases by at most 1 at every update.

$$\begin{aligned}\|\mathbf{w}_{t+1}\|_2^2 = \mathbf{w}_{t+1}^\intercal \mathbf{w}_{t+1} &= (\mathbf{w}_t + y_i\mathbf{x}_i)^\intercal(\mathbf{w}_t + y_i\mathbf{x}_i) \\ &= \|\mathbf{w}_t\|_2^2 + 2y_i \cdot \mathbf{w}_t^\intercal \mathbf{x}_i + \|\mathbf{x}_i\|_2^2\end{aligned}$$

Since $y_i \cdot \mathbf{w}_t^\intercal \mathbf{x}_i < 0$ and $\|\mathbf{x}_i\|_2^2 \leq 1$, we have $\mathbf{w}_{t+1}^\intercal \mathbf{w}_{t+1} \leq \mathbf{w}_t^\intercal \mathbf{w}_t + 1$.

Now suppose $\mathbf{w}_0 = 0$. After k iterations we have:

$$\mathbf{w}_k^\intercal \mathbf{w}^* \geq k\gamma$$

$$\|\mathbf{w}_k\|^2 < k \quad \Rightarrow \quad \|\mathbf{w}_k\| < \sqrt{k}$$

Since $\mathbf{w}_k^\intercal \mathbf{w}^* = \|\mathbf{w}_k\| \cdot \|\mathbf{w}^*\| \cos\theta_i$, we obtain $\sqrt{k} > k\gamma$, thus $\boxed{k < \frac{1}{\gamma^2}}$.

This enforces an upper bound on the number of iterations. Hence, our algorithm converges to an optimal parameter $\mathbf{w}*$ in atmost $\frac{1}{\gamma^2}$ iterations if the dataset is *linearly separable.* □

9.3.1 Limitations of Perceptrons

1. This algorithm does not give us a rate of convergence.
2. In some datasets, $\frac{1}{\gamma^2}$ is extremely large, rendering the upper bound on number of iterations pointless.
3. May not converge if dataset is not *linearly separable.*

9.4 Loss function view of Perceptrons

Our goal of choosing $\mathbf{w}$ to maximize $y_i(\mathbf{w}^\intercal \mathbf{x}_i)$ can also be framed as:

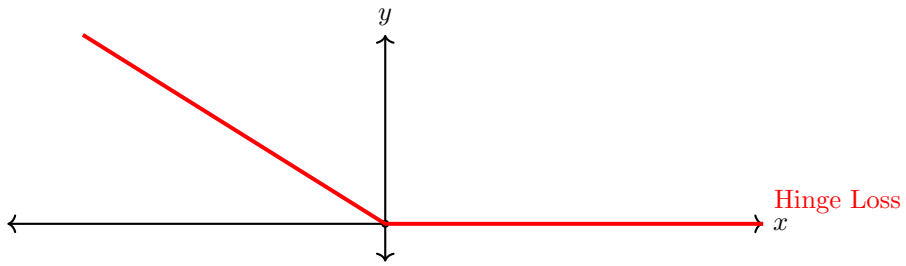
$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i \in \text{ misclassified}} -y_i \cdot (\mathbf{w}^\intercal \mathbf{x}_i)$$

Thus, we can make a loss function L_i as such:

$$L_i = \max\{0, -y_i \mathbf{w}^\intercal \mathbf{x}_i\}$$

and $L = \sum_{i=1}^n L_i$.

The L_i we have defined is also known as **hinge loss**, as it appears as a hinge at the origin:



Since this is not differentiable at origin, we take the derivative there to be 0 or some other arbitrary value. It does not usually matter because, statistically, we will have to face such a situation with a probability of 0.

In this way, we can view our Perceptron algorithm as the Stochastic Gradient Descent of this loss function, as:

1. Like SGD, we calculate the gradient for a single data point at a time.
2. The derivative of this loss function comes out to be exactly the update we have been using so far.

The proof of the same is below:

Proof. We are performing a stochastic gradient descent (SGD) update: $\mathbf{w}_{t+1} = \mathbf{w}_t - \nabla_{\mathbf{w}} L_i$.

Note that

$$\nabla_{\mathbf{w}} L_i = \begin{cases} 0, & \text{if the prediction is correct} \\ -y_i \mathbf{x}_i, & \text{if the prediction is incorrect} \end{cases}$$

Thus, the update is only performed when $y_i \neq \mathbf{w}_t^\intercal \mathbf{x}_i$, and it is done as follows: $\mathbf{w}_{t+1} = \mathbf{w}_t + y_i \mathbf{x}_i$, which is exactly the same as the Perceptron Algorithm. □

9.5 Decision Trees

At the end of the lecture, we touched upon decision trees, another tool used for classification problems. Decision trees are, as you probably guessed, trees, where each node considers a feature or a function of the features of the input vector, and the tool accordingly traverses to one of the children of the node based on the value of the same. It simulates a human-like case-by-case classification mechanism. A diagram to help visualise the same can be found below:

Suppose our dataset has 4 features *(cyl, disp, origin, year)* and has to be classified into 2 classes *good* and *bad*.

Fuel Efficiency	Cyl	Disp	Origin	Year
good	3	low	⋮	⋮
bad	4	med	⋮	⋮
⋮	5	high	⋮	⋮
⋮	6	⋮	⋮	⋮

A decision tree for the same might look like the following: