Lecture 15: Kernel perceptron, Neural Networks, SVMs etc

Instructor: Prof. Ganesh Ramakrishnan

Binary Classification using Perceptron

Perceptron Classifier

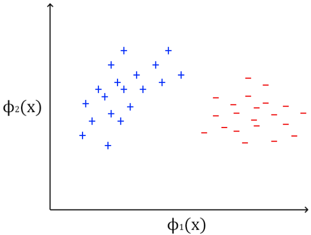
- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Assuming linearly separability, is there a learning rule that converges in finite time?

Perceptron Classifier

- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Assuming linearly separability, is there a learning rule that converges in finite time?
- Naive Idea: Perform linear regression by constraining $y \in \{+1, -1\}$.

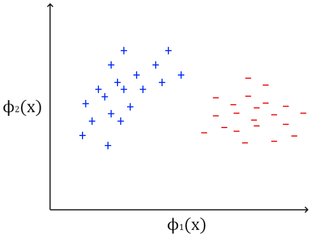
Perceptron Classifier

- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Assuming linearly separability, is there a learning rule that converges in finite time?
- Naive Idea: Perform linear regression by constraining $y \in \{+1, -1\}$.
- Can we do better? What is ideal?
- Desirable: Any new (unseen) input pattern similar to a seen pattern is **classified** correctly



Perceptron Classifier

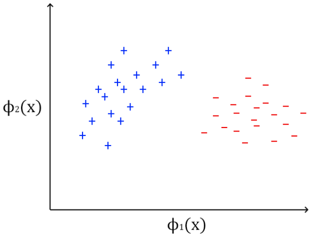
- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Assuming linearly separability, is there a learning rule that converges in finite time?
- Naive Idea: Perform linear regression by constraining $y \in \{+1, -1\}$.
- Can we do better? What is ideal?
- Desirable: Any new (unseen) input pattern similar to a seen pattern is **classified** correctly



Linear Classification?

Perceptron Classifier

- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Assuming linearly separability, is there a learning rule that converges in finite time?
- Naive Idea: Perform linear regression by constraining $y \in \{+1, -1\}$.
- Can we do better? What is ideal?
- Desirable: Any new (unseen) input pattern similar to a seen pattern is **classified** correctly



Linear Classification?

$\mathbf{w}^\top \phi(\mathbf{x}) + b \geq 0$ *for +ve points* $(y = +1)$

$\mathbf{w}^\top \phi(\mathbf{x}) + b < 0$ *for -ve points* $(y = -1)$

$\mathbf{w}, \phi \in \mathbb{R}^m$

Perceptron Classifier: Setting up Notation

- Often, b is indirectly captured by including it in $\mathbf{w}$, and using a ϕ as: $\phi_{aug} = [\phi, 1]$
- Thus, $\mathbf{w}^\top \phi(\mathbf{w})$

$$= \begin{bmatrix} w_1 & w_2 & w_3 & \dots & w_m & b \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \vdots \\ \phi_m \\ 1 \end{bmatrix}$$

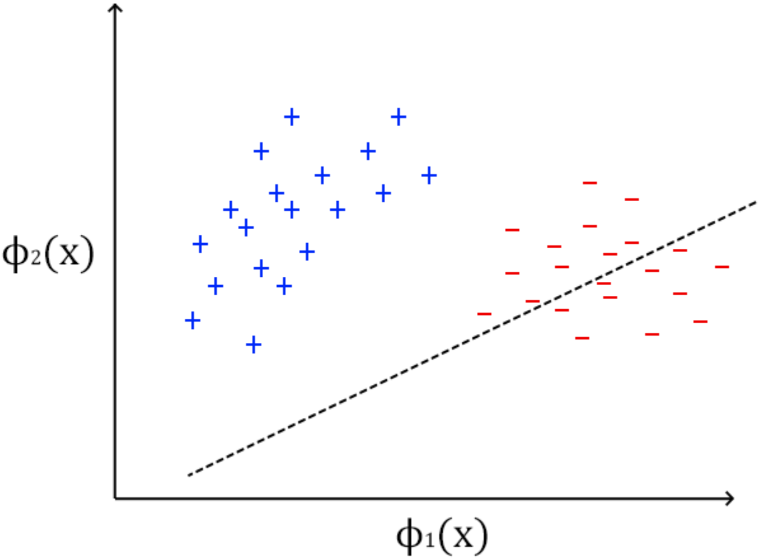
- $\mathbf{w}^\top \phi(\mathbf{x}) = 0$ is the separating hyperplane.

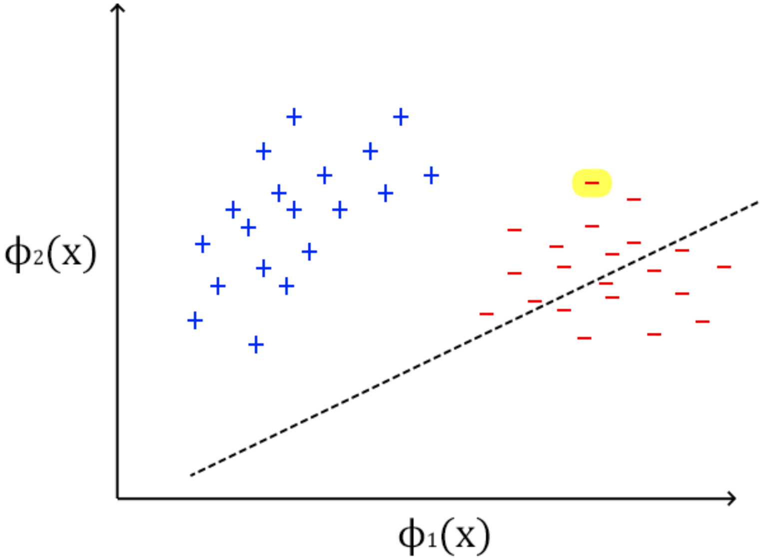
Perceptron Intuition

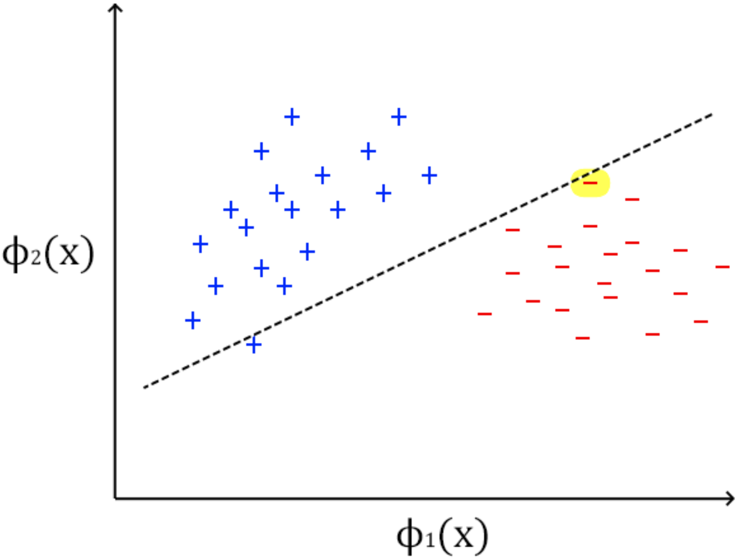
1. Go over all the existing examples, whose class is known, and check their classification with the current weight vector
2. If correct, continue
3. If not, marginally correct the weights
   - By adding to the weights a quantity that is proportional to the product of the input pattern with the desired output $y = \pm 1$

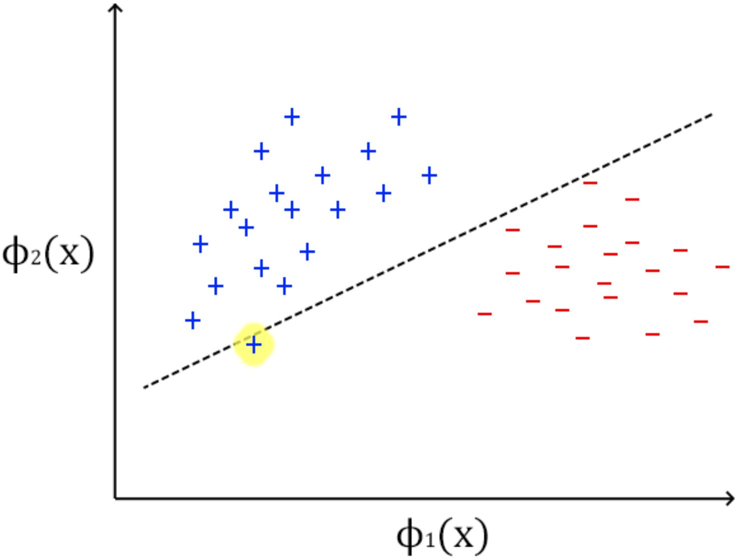
Perceptron Update Rule

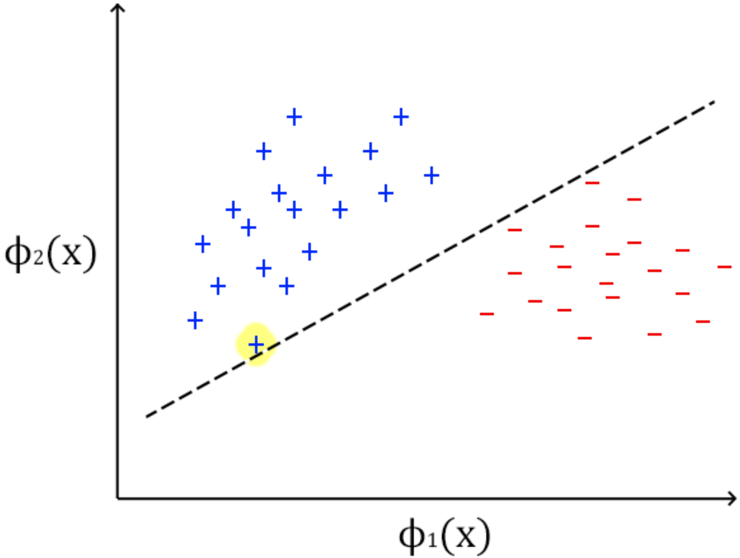
- Start with some weight vector $\mathbf{w}^{(0)}$, and for $k = 0, 1, 2, 3, \ldots, n$ (for every example), do: $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + y'\phi(x')$
- where $\mathbf{x}'$ s.t. $\mathbf{x}'$ is misclassified by $(\mathbf{w}^{(k)})^\top \phi(\mathbf{x})$
i.e. $y'(\mathbf{w}^{(k)})^\top \phi(\mathbf{x}') < 0$


ϕ2(x)
ϕ1(x)


ϕ2(x)
ϕ1(x)


ϕ2(x)
ϕ1(x)


ϕ2(x)
ϕ1(x)



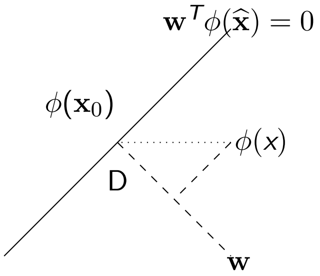
ϕ2(x)
ϕ1(x)

Some Notes about the Perceptron Algorithm

- Perceptron does not find the *best* seperating hyperplane, it finds *any* seperating hyperplane.
- In case the initial $\mathbf{w}$ does not classify all the examples, the seperating hyperplane corresponding to the final $\mathbf{w}^*$ will often *pass through an example*.
- The seperating hyperplane does not provide enough breathing space – this is what SVMs address!

Perceptron Update Rule: Further analysis

- Explicitly account for signed distance of (misclassified) points from the hyperplane $\mathbf{w}^T\phi(\widehat{\mathbf{x}}) = 0$
  Distance from hyperplane can be calculated as follows:



  perpendicular distance $= y\mathbf{w}^T(\phi(\mathbf{x}_0) - \phi(\mathbf{x}))$
  Since $\mathbf{w}^T(\phi(\mathbf{x}_0)) = 0$ we get distance $= -y\mathbf{w}^T(\phi(\mathbf{x}))$

Perceptron Update Rule: Further analysis

- Perceptron works for two classes ($y = \pm 1$). A point is misclassified if $y\mathbf{w}^T(\phi(\mathbf{x})) < 0$
- Perceptron Algorithm:
  - INITIALIZE: w=ones()
  - REPEAT: for each $< \mathbf{x}, y >$
    - If $y\mathbf{w}^T\Phi(\mathbf{x}) < 0$
    - then, $\mathbf{w} = \mathbf{w} + \Phi(\mathbf{x}).y$
    - endif
- **Intuition:**

Perceptron Update Rule: Further analysis

- Perceptron works for two classes ($y = \pm 1$). A point is misclassified if $y\mathbf{w}^T(\phi(\mathbf{x})) < 0$
- Perceptron Algorithm:
  - INITIALIZE: w=ones()
  - REPEAT: for each $< \mathbf{x}, y >$
    - If $y\mathbf{w}^T\Phi(\mathbf{x}) < 0$
    - then, $\mathbf{w} = \mathbf{w} + \Phi(\mathbf{x}).y$
    - endif
- **Intuition:**

$$
\begin{aligned}
y(\mathbf{w}^{(k+1)})^T\phi(\mathbf{x}) &= y(\mathbf{w}^k + y\phi^T(\mathbf{x})\phi(\mathbf{x}) \\
&= y(\mathbf{w}^k)^T\phi(\mathbf{x}) + y^2\|\phi(\mathbf{w})\|^2 \\
&> y(\mathbf{w}^k)^T\phi(\mathbf{x})
\end{aligned}
$$

Since $y(\mathbf{w}^k)^T\phi(\mathbf{x}) \leq 0$, we have $y(\mathbf{w}^{(k+1)})^T\phi(\mathbf{x}) \; > \; y(\mathbf{w}^k)^T\phi(\mathbf{x}) \Rightarrow$ more hope that this point is classified correctly now.

Perceptron Update Rule: Further analysis

- Tries to minimize the error function E (sum of unsigned distances from hyperplane to misclassified points) over misclassified examples

$$E = -\sum_{(\mathbf{x},y)\in\mathcal{M}} y\mathbf{w}^T\phi(\mathbf{x})$$

where $\mathcal{M} \subseteq \mathcal{D}$ is the set of misclassified examples.

- **Gradient Descent (Batch Perceptron) Algorithm** $\nabla_{\mathbf{w}} E = -\sum_{(\mathbf{x},y)\in M} y\phi(\mathbf{x})$

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^k - \eta\nabla_{\mathbf{w}} E \\ &= \mathbf{w}^k + \eta \sum_{(\mathbf{x},y)\in\mathcal{M}} y\phi(\mathbf{x})\end{aligned}$$

Perceptron Update Rule: Further analysis

- Batch update considers all misclassified points simultaneously

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^k - \eta \nabla_{\mathbf{w}} E \\ &= \mathbf{w}^k + \eta \sum_{(\mathbf{x},y)\in\mathcal{M}} y\phi(\mathbf{x})\end{aligned}$$

- Perceptron update $\Rightarrow$ *Stochastic Gradient Descent*:
$\nabla_{\mathbf{w}} E = -\sum_{(\mathbf{x},y)\in\mathcal{M}} y\phi(\mathbf{x}) = -\sum_{(\mathbf{x},y)\in\mathcal{M}} \nabla_{\mathbf{w}} E(\mathbf{x})$ s.t. $E(\mathbf{x}) = -y\mathbf{w}^T\phi(\mathbf{x})$

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^k - \eta \nabla_w E(\mathbf{x}) \\ &= \mathbf{w}^k + \eta y\phi(\mathbf{x}) \qquad \text{(for any } (\mathbf{x}, y) \in \mathcal{M}\text{)}\end{aligned}$$

Perceptron Update Rule: Further analysis

- **Formally,**:- If $\exists$ an optimal separating hyperplane with parameters $\mathbf{w}^*$ such that,

$$\forall\ (\mathbf{x}, y),\ \ y\phi^T(\mathbf{x})\mathbf{w}^* \geq 0$$

then the perceptron algorithm converges.
Proof:- We want to show that

$$\lim_{k\to\infty} \|\mathbf{w}^{(k+1)} - \rho\mathbf{w}^*\|^2 = 0 \tag{1}$$

(If this happens for some constant ρ, we are fine.)

Perceptron Update Rule: Further analysis

- **Formally,**:- If $\exists$ an optimal separating hyperplane with parameters $\mathbf{w}^*$ such that,

$$\forall\ (\mathbf{x}, y),\ \ y\phi^T(\mathbf{x})\mathbf{w}^* \geq 0$$

  then the perceptron algorithm converges.
  Proof:- We want to show that

$$\lim_{k\to\infty} \|\mathbf{w}^{(k+1)} - \rho\mathbf{w}^*\|^2 = 0 \tag{1}$$

(If this happens for some constant ρ, we are fine.)

$$\|\mathbf{w}^{(k+1)} - \rho\mathbf{w}^*\|^2 = \|\mathbf{w}^k - \rho\mathbf{w}^*\|^2 + \|y\phi(\mathbf{x})\|^2 + 2y(\mathbf{w}^k - \rho\mathbf{w}^*)^T\phi(\mathbf{x}) \tag{2}$$

- For convergence of perceptron, we need L.H.S. to be less than R.H.S. at every step, although by some small but non-zero value (with $\theta \neq 0$)

$$\|\mathbf{w}^{(k+1)} - \rho\mathbf{w}^*\|^2 \leq \|\mathbf{w}^k - \rho\mathbf{w}^*\|^2 - \theta^2 \tag{3}$$