Lecture 14: Local linear regression non-parametric estimation, perceptron and update algo, etc

Instructor: Prof. Ganesh Ramakrishnan

Basis function expansion & Kernel: Part 1

We saw the that for $p \in [0, \infty)$, under certain conditions on K, the following can be equivalent representations

- $$f(\mathbf{x}) = \sum_{j=1}^{p} w_j \phi_j(\mathbf{x})$$
- And
$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$
- For what kind of regularizers, loss functions and $p \in [0, \infty)$ will these dual representations hold?[1]

[1] Section 5.8.1 of Tibshi.

Basis function expansion & Kernel: Part 2

- We could also begin with

$$f(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(\mathbf{x}, \mathbf{x}_i)$$

  and impose no constraints on K.
- *E.g.:* $K_k(x_q, x) = I(||x_q - x|| \leq ||x_{(k)} - x_0||)$ where $x_{(k)}$ is the training observation ranked k^{th} in distance from x and $I(S)$ is the indicator of the set S
- This is precisely the Nearest Neighbor Regression model
- Kernel regression and density models are other examples of such *local regression* methods[2]

[2] Section 2.8.2 of Tibshi

Kernel weighted regression

Weights obtained using some kernel $K(.,.)$. Given a training set of points $\mathcal{D} = \left\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)\right\}$, we predict a regression function $f(x') = (\mathbf{w}^\top \phi(x') + b)$ for each test (or query point) x' as follows:

$$(\mathbf{w}', b') = \operatorname*{argmin}_{\mathbf{w}, b} \sum_{i=1}^{n} K(x', x_i) \left(y_i - (\mathbf{w}^\top \phi(x_i) + b)\right)^2$$

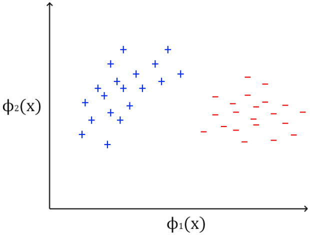
1. If there is a closed form expression for $(\mathbf{w}', b')$ and therefore for $f(x')$ in terms of the known quantities, derive it.
2. How does this model compare with linear regression and $k-$nearest neighbor regression? What are the relative advantages and disadvantages of this model?
3. In the one dimensional case (that is when $\phi(x) \in \Re$), graphically try and interpret what this regression model would look like, say when $K(.,.)$ is the linear kernel[3].

[3]Hint: What would the regression function look like at each training data

More on Kernels after some classification

1. We will delve a bit more into kernel density estimation etc after some treatment of classification

Perceptron



$w^\top \phi(x) + b \geq 0$ *for +ve points (y= +1)*
$w^\top \phi(x) + b < 0$ *for -ve points (y= -1)*
$w, \phi \in \mathbb{R}^m$

- Assuming the problem is linearly separable, there is a learning rule that converges in a finite time.
- A new (unseen) input pattern that is similar to an old (seen) input pattern is likely to be classified correctly

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

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

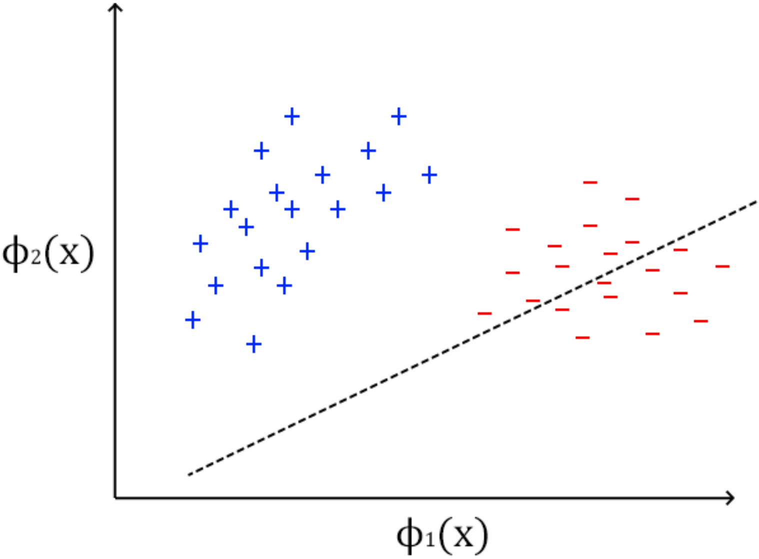
- $w^\top \phi(x) = 0$ is the separating hyperplane.

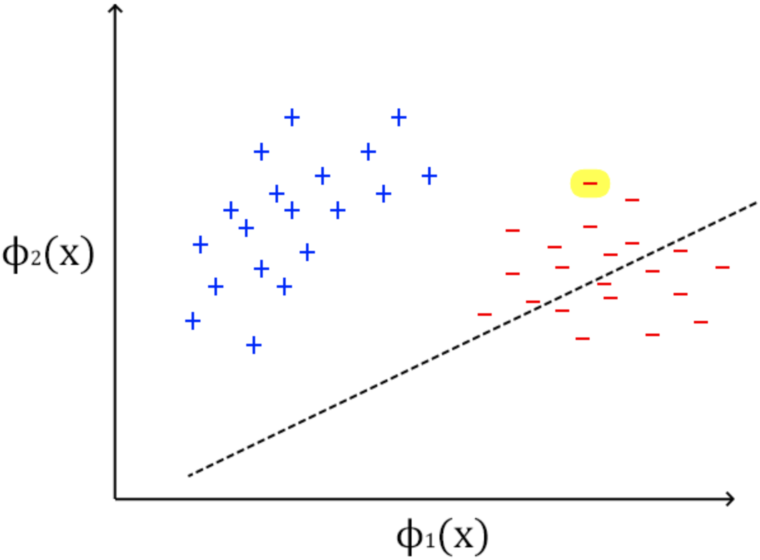
Perceptron Intuition

- Go over all the existing examples, whose class is known, and check their classification with a current weight vector
- If correct, continue
- If not, add to the weights a quantity that is proportional to the product of the input pattern with the desired output y (1 or -1)

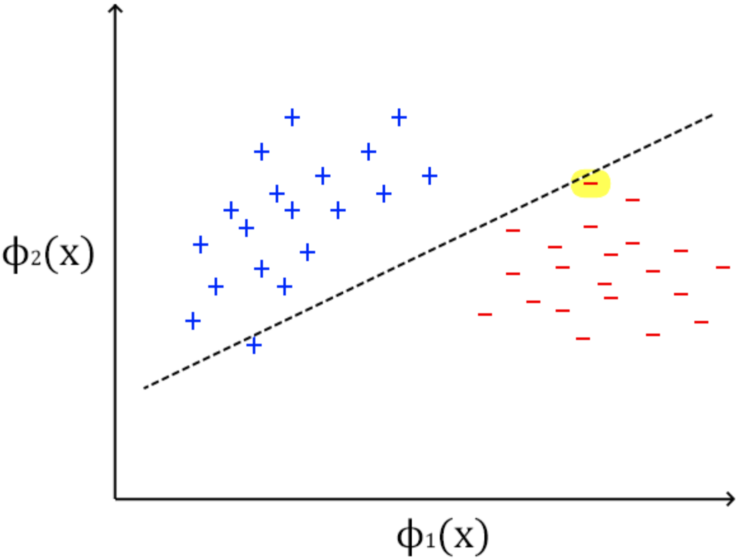
Perceptron Update Rule

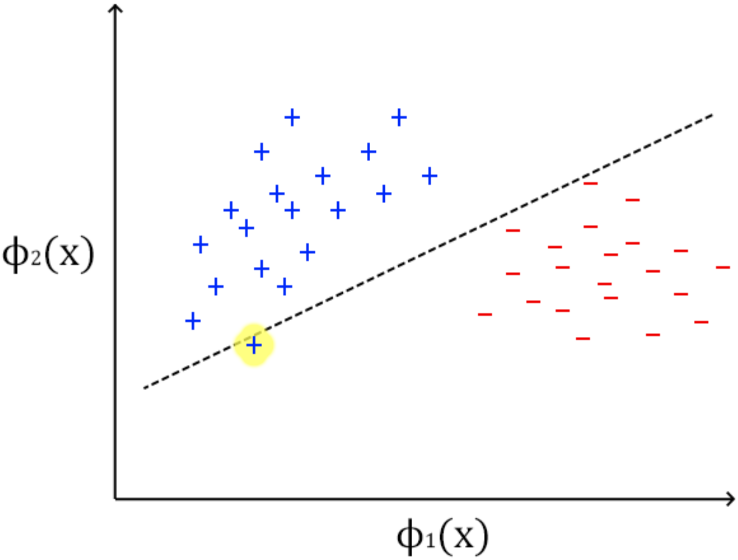
- Start with some weight vector $w^{(0)}$, and for $k = 1, 2, 3, \ldots, n$ (for every example), do:
  $w^{(k)} = w^{(k-1)} + \Gamma\phi(x')$
- where x' s.t. x' is misclassified by $(w^{(k)})^\top\phi(x)$
  i.e. $y'(w^{(k)})^\top\phi(x') < 0$


ϕ2(x)
ϕ1(x)

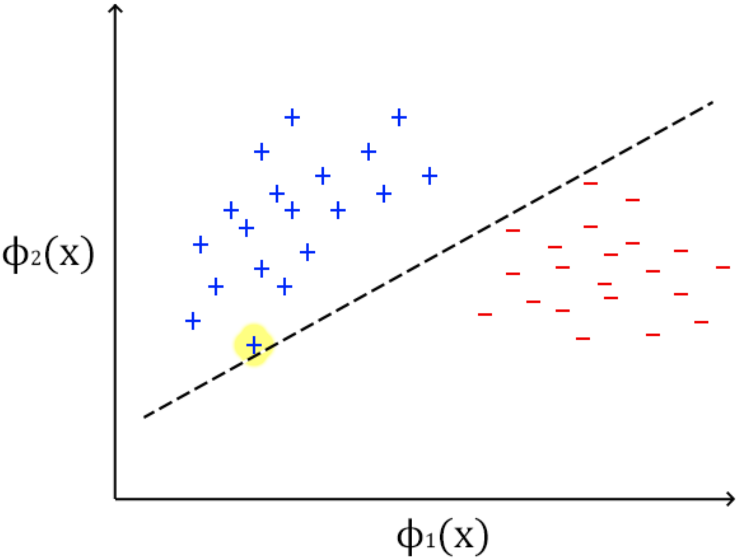


ϕ2(x)
ϕ1(x)


$\phi_2(x)$
$\phi_1(x)$



$\phi_2(x)$
$\phi_1(x)$



φ2(x)
φ1(x)

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