FoDA L28

Other Classifiers

SVMs, Neural Nets,

Decision Trees, etc.

<u>Input</u> $(X, y) \in \mathbb{R}^d \times \{-,+\}$

$x_i \in \mathbb{R}^d$

$y_i \in \{-1, +1\}$

Classification

<u>Goal</u> function

$f : \mathbr{R}^d \to \{-,+\}$

or $g : \mathbb{R}^d \to \mathbb{R}$

linear $g(x_i) = \langle \alpha, (1, x_i) \rangle$

so

$f(x_i) = y_i$ on as many x_i, y_i as possible

$\text{sign}(g(x_i)) = y_i$

<u>Supervised</u>

Cross-Validation

$(X, y) \to$ Train

$\quad\quad\ \ \to$ Test

<u>Goal</u>

generalize on new data

iid new (x, y) and

Non-linear Kernel Classifiers

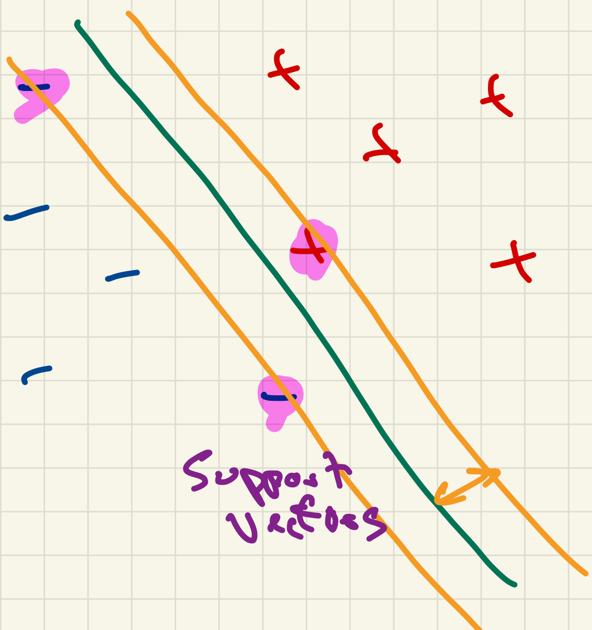
$\alpha \quad x = (1, \ldots)$

mistake counter

$\alpha \in \mathbb{R}^n$

$$g(x) = \langle w, x \rangle = \sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle$$

$$= \sum_{i=1}^{n} \alpha_i y_i K(x_i, x)$$



$$\alpha = (\alpha_1, \alpha_2 \ldots \alpha_n)$$

mostly $\alpha_j = 0$

Non-0 α_j = support vectors

$$\alpha = (\underset{i=1}{2}, 0, 0, \underset{i=4}{1}, 0, 0, \underset{i=7}{2}, 0, \underset{i=9}{1})$$

Cost function

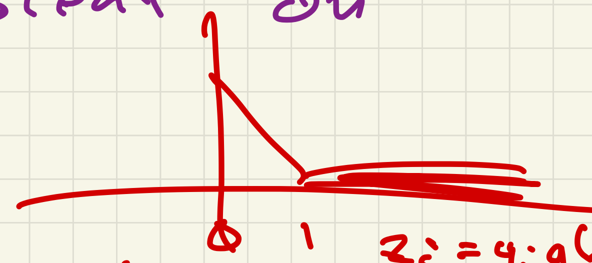
$$h(\alpha) = \sum_{i=1}^{n} \ell\left(y_i\, g_\alpha(x_i)\right) \qquad g_\alpha(x_i) = \sum_{j=1}^{n} \alpha_j y_j K(x_j, x_i)$$

$\alpha \in \mathbb{R}^n$, but can restrict to k support vectors.

Run gradient descent on $h(\alpha)$

Find (a) Perceptron

(b) SGD w/ hinge loss

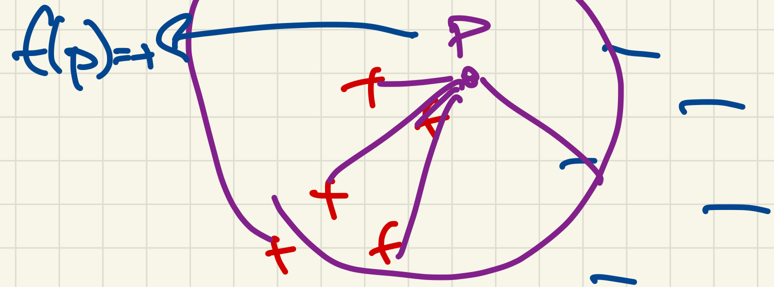


$z_i = y_i\, g(x_i)$

KNN classifiers

Uses distance $D: X \times X \to \mathbb{R}$

No training

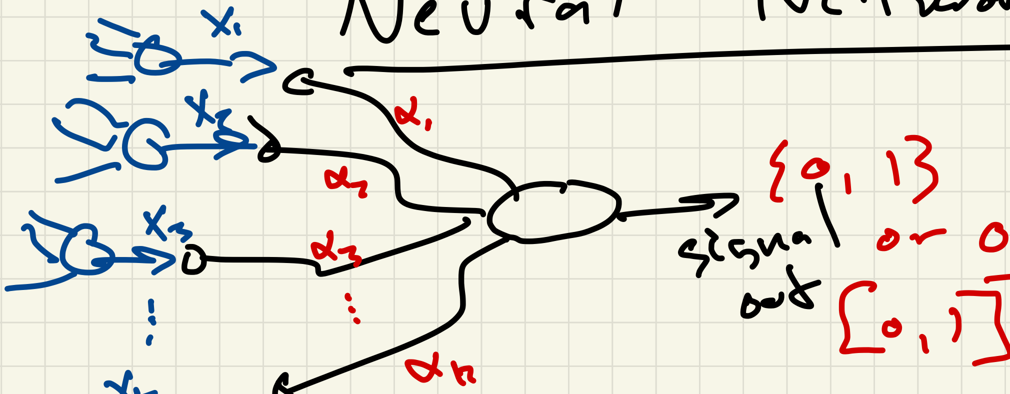


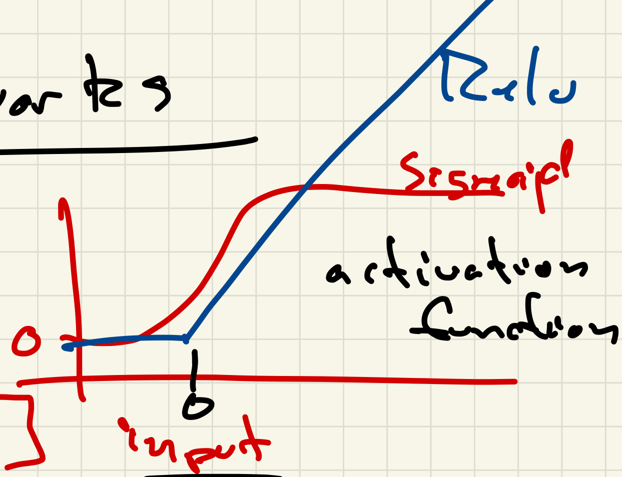
$f(p) \to \{-1, +1\}$

= Find the k closest points $x_1, x_2 \ldots x_k \in X$ to p.

Then vote: which sign is most prevalent $\Rightarrow \{-, +\}$

Neural Networks





$$\sum_{i=1}^{n} \alpha_i x_i = \langle \alpha, x \rangle$$

$\alpha_i \in \mathbb{R}$

$x_i \in [0,1]$

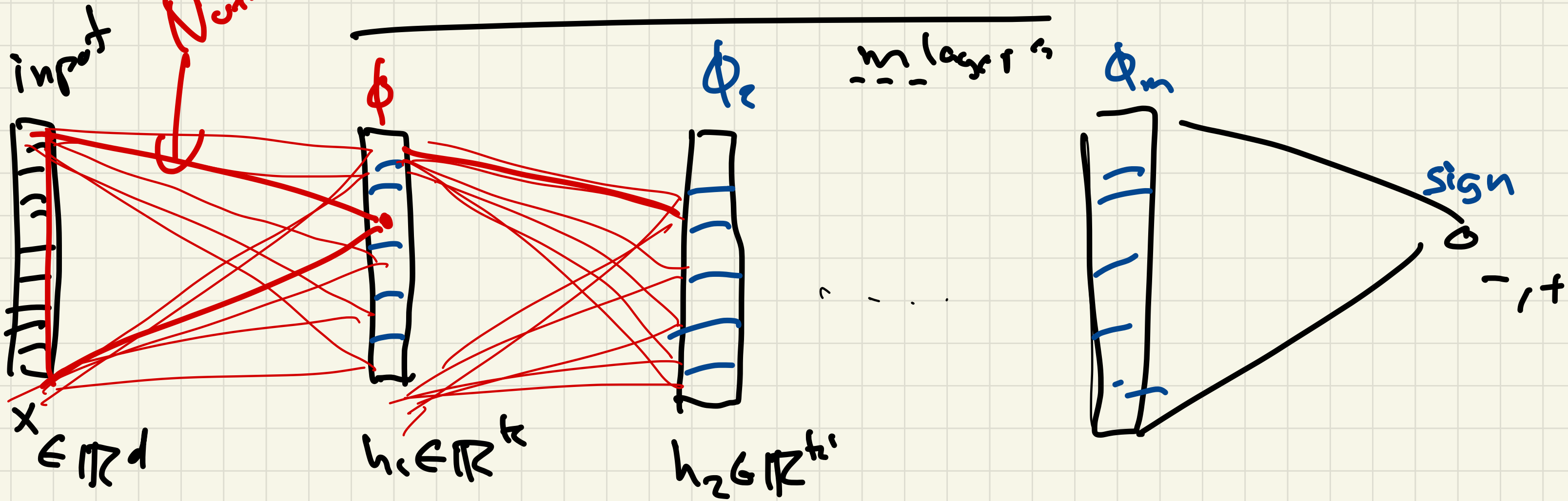
Activation Function

$\phi : \mathbb{R} \to [0,1]$ or ... $[-1,+1]$, or $[0,\infty)$

sigmoid $\phi(y) = \dfrac{1}{1+e^{-y}} \in [0,1]$

ReLu $\phi(y) = \max(0,y) \in [0,\infty)$

Nueral Network



Nueron, α_j, ϕ_j

$$h_1(j) = \phi_1(\langle \alpha_j, x \rangle)$$

Parameters

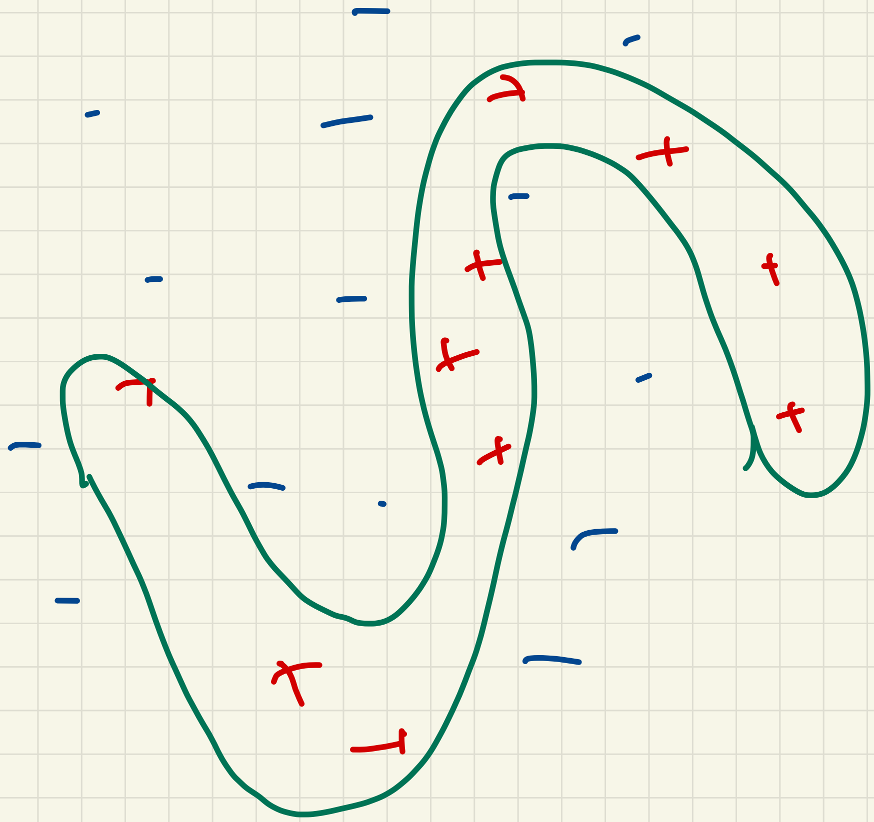
$\alpha_1, \dots, \alpha_k$ layer 1

$\alpha_1 \dots \alpha_{k_2}$ layer 2

$\vdots$

$\alpha_1 \dots \alpha_{k_m}$ layer m

$\approx m \cdot k^2$ parameters.

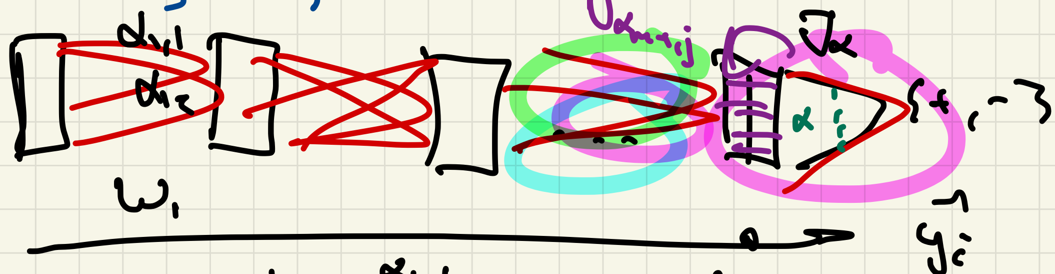


Train (Deep) NN w/ GD.

Can efficient compute gradient.

$\approx k^2 m$ parameters

by layers from last backwards.



Backpropagation

$w_1 = \alpha_{1,1}, \alpha_{1,2}, \alpha_{1,\dots k}$

$(y_i - \hat{y}_i)^2$

$$g(x_i) = W^m \phi^m (W^{m-1} \phi^{m-1} (\dots \phi^1 (W^1 x) \dots))$$

Decision Trees

$x \in \mathbb{R}^d$ or $\{A, \dots, C\}$