Intro to ML

November 15th, 2021

Logistics

- Midterm (Nov 22)
- Everything up to kernel machines
- The subsections covered in each chapter can be on the exam
- I will design the midterm will also reference hw
- 2 hrs exam
- 1 hand-written A4 cheat sheet allowed (both side ok)

CHAPTER 14:

Kernel Machines

Kernel Machines

- Discriminant-based: No need to estimate densities first
- Define the discriminant in terms of support vectors
  - Support vectors: subset of training instances
- The use of kernel functions, application-specific measures of similarity
- Convex optimization problems with a unique solution

Optimal Separating Hyperplane

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_t \text{ where } r^t = \begin{cases} +1 & \text{if } \mathbf{x}^t \in C_1 \\ -1 & \text{if } \mathbf{x}^t \in C_2 \end{cases}$$

find $\mathbf{w}$ and w_0 such that

$$\mathbf{w}^T\mathbf{x}^t + w_0 \geq +1 \text{ for } r^t = +1$$

$$\mathbf{w}^T\mathbf{x}^t + w_0 \leq +1 \text{ for } r^t = -1$$

which can be rewritten as

$$r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq \boxed{+1}$$

Not simply >0
With some distance

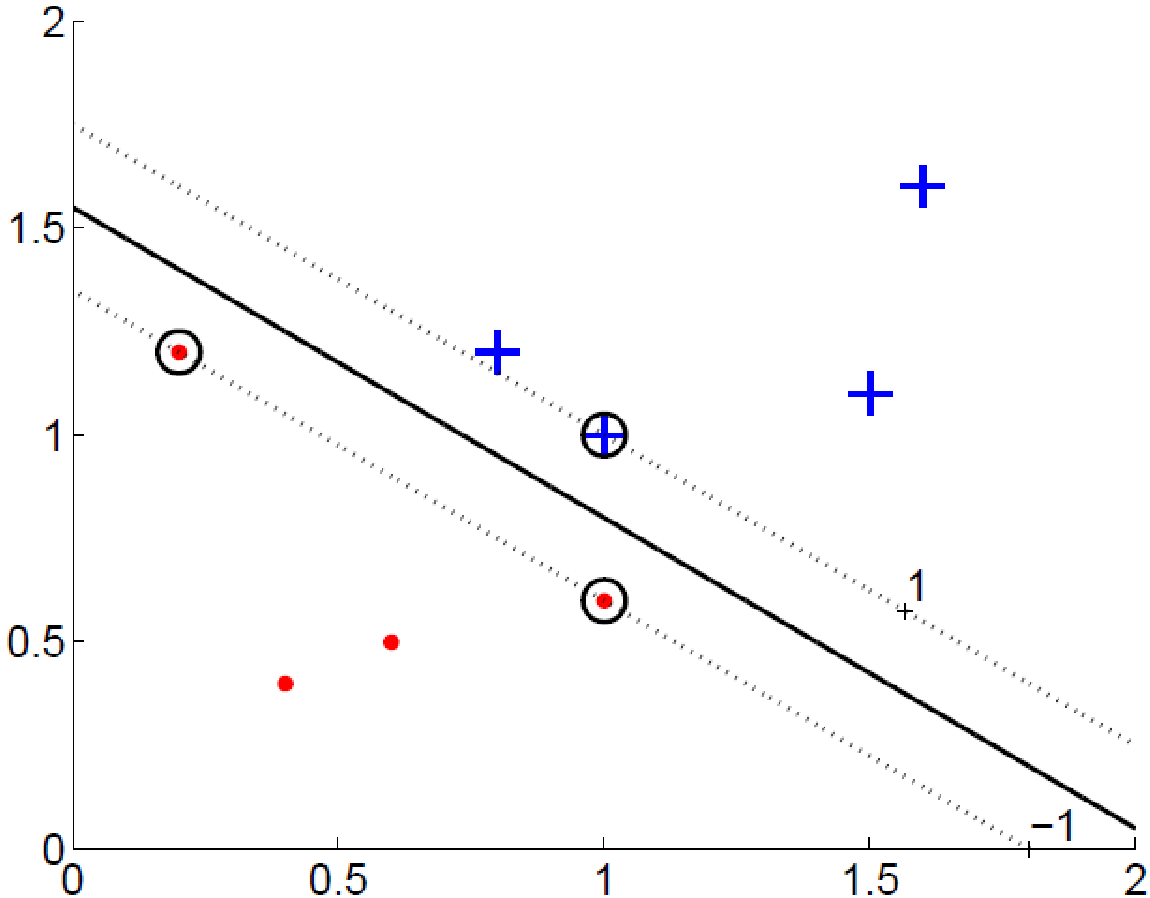
(Cortes and Vapnik, 1995; Vapnik, 1995)

Margin

- Distance from the discriminant to the closest instances on either side
- Distance of x to the hyperplane is $$\frac{\left|\mathbf{w}^T\mathbf{x}^t + w_0\right|}{\|\mathbf{w}\|}$$
- We require $$\frac{r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right)}{\|\mathbf{w}\|} \geq \rho, \forall t$$ Like to maximize this distance
- For a unique sol'n, fix $\rho\|w\|=1$, and to max margin equal minimize w

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1, \forall t$$

Margin



$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ subject to } r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) \geq +1, \forall t$$

Use LaGrange multiplier

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t \left[r^t\left(\mathbf{w}^T\mathbf{x}^t + w_0\right) - 1\right]$$

$$\boxed{= \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{t=1}^{N} \alpha^t r^t \left(\mathbf{w}^T\mathbf{x}^t + w_0\right) + \sum_{t=1}^{N} \alpha^t}$$

karush kuhn tucker condition

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{t=1}^{N} \alpha^t r^t \mathbf{x}^t$$

$$\frac{\partial L_p}{\partial w_0} = 0 \Rightarrow \sum_{t=1}^{N} \alpha^t r^t = 0$$

The reason to rewrite this

- Original complexity depends on d
- Turn the solution into complexity depends on N

$$L_d = \frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right) - \mathbf{w}^T\sum_t \alpha^t r^t \mathbf{x}^t - w_0\sum_t \alpha^t r^t + \sum_t \alpha^t$$

$$= -\frac{1}{2}\left(\mathbf{w}^T\mathbf{w}\right) + \sum_t \alpha^t$$

$$= -\frac{1}{2}\sum_t\sum_s \alpha^t\alpha^s r^t r^s \left(\mathbf{x}^t\right)^T \mathbf{x}^s + \sum_t \alpha^t$$

solve for alpha

subject to $\sum_t \alpha^t r^t = 0$ and $\alpha^t \geq 0, \forall t$

Pick any support vector and solve for w
Average over support vector

Most α^t are 0 and only a small number have $\alpha^t > 0$;

they are the support vectors they satisfy $r^t(w^T x^t + w_0) = 1$

These are support vector machines, it only cares those on the boundaries not within the decision regions

testing

- $g(x)=w^T x + w_0$
- Choose the results according to sign

Soft Margin Hyperplane

- Not linearly separable

$$r^t\left(\mathbf{w}^T x^t + w_0\right) \geq 1 - \xi^t$$

Slack variable

$$0 < \xi^t < 1$$

Error in the margin

$$\xi^t \geq 1$$

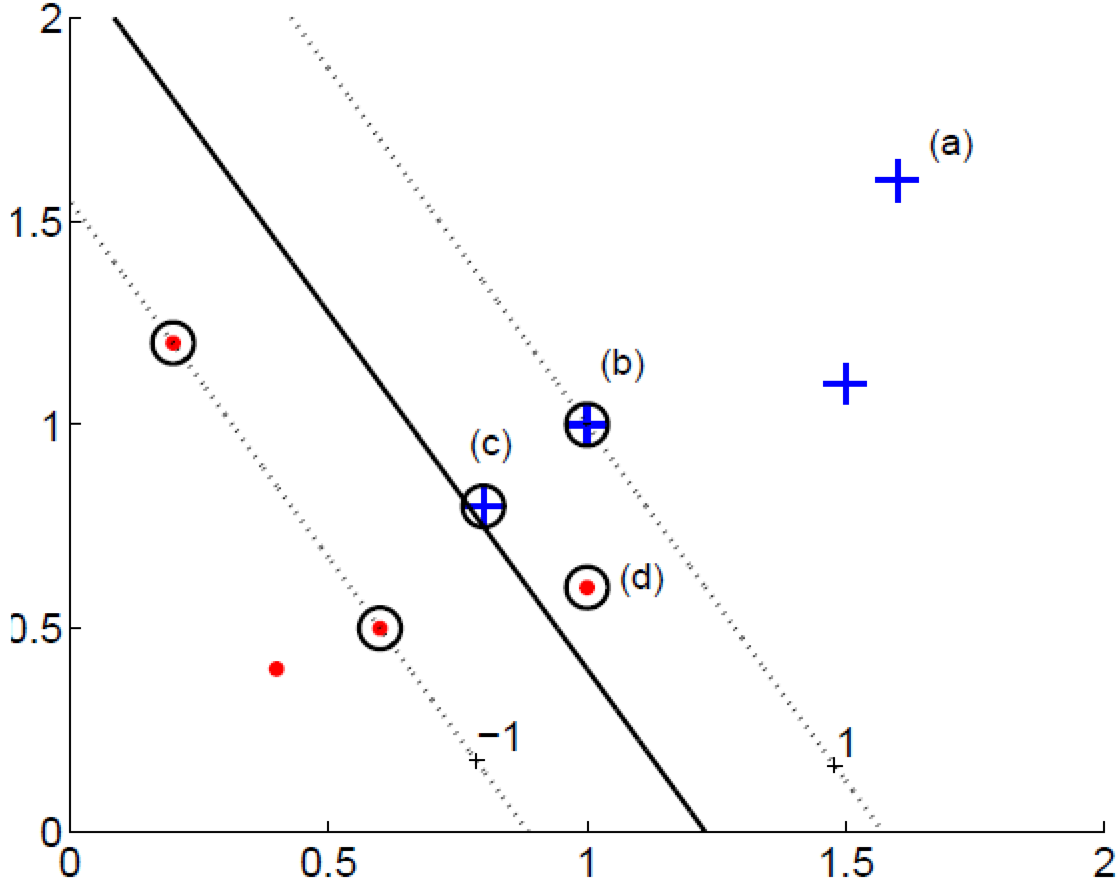
misclassified

- Soft error $\sum_t \xi^t$

- New primal is

Lagrange to ensure positivity of error

$$L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t \xi^t - \sum_t \alpha^t\left[r^t\left(\mathbf{w}^T x^t + w_0\right) - 1 + \xi^t\right] - \sum_t \mu^t \xi^t$$



C is a tunable parameter

Trade off between margin maximization and error minimization

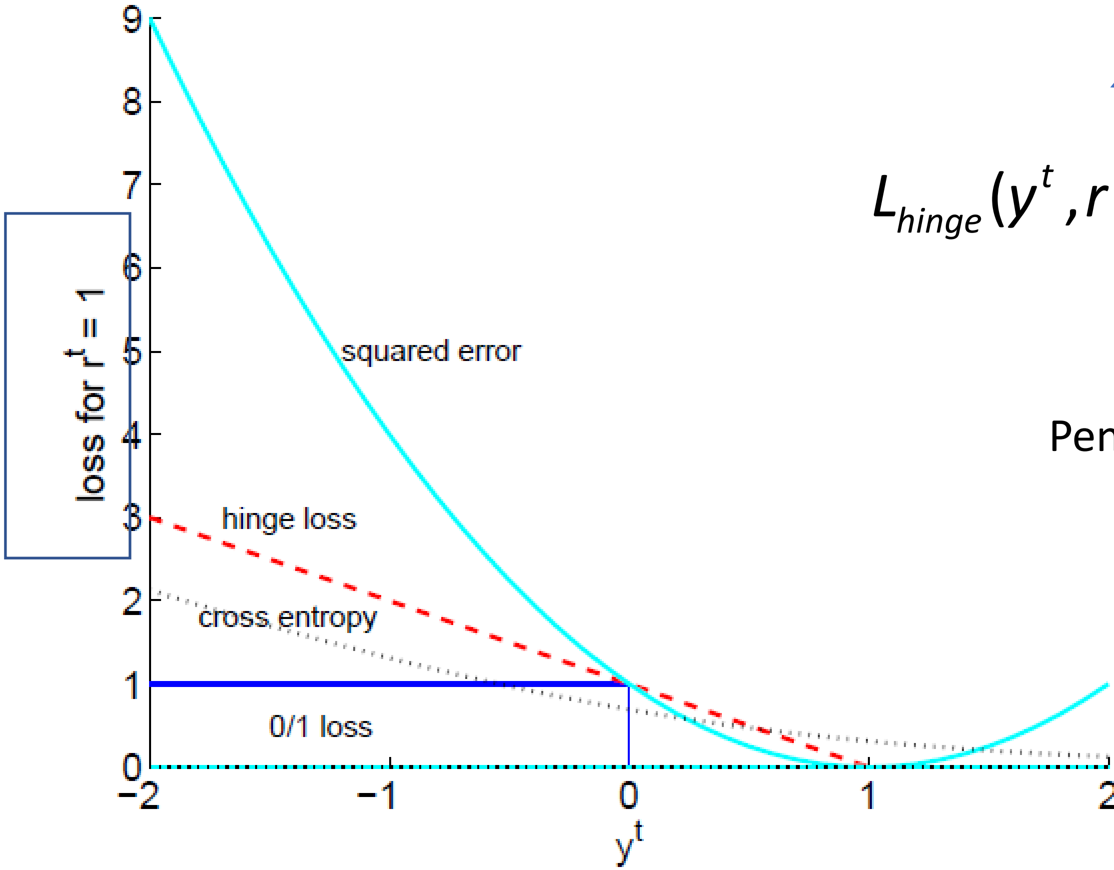
Too large -> high penalty for error -> may overfit

Too small -> not enough penalty for error -> may underfit

Hinge Loss

$$r^t\left(\mathbf{w}^T x^t + w_0\right) \geq \boxed{1 - \xi^t}$$

$$L_{hinge}(y^t, r^t) = \begin{cases} 0 & \text{if } y^t r^t \geq 1 \\ 1 - y^t r^t & \text{otherwise} \end{cases}$$



Penalizes instance in the margin

Kernel Trick

- Preprocess input ***x*** by basis functions

$$\boldsymbol{z} = \boldsymbol{\varphi}(\boldsymbol{x}) \qquad g(\boldsymbol{z}) = \boldsymbol{w}^T \boldsymbol{z}$$

$$g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\varphi}(\boldsymbol{x})$$

- The SVM solution

$$\mathbf{w} = \sum_t \alpha^t r^t \mathbf{z}^t = \sum_t \alpha^t r^t \boldsymbol{\varphi}(\mathbf{x}^t)$$

$$g(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) = \sum_t \alpha^t r^t \boxed{\boldsymbol{\varphi}(\mathbf{x}^t)^T \boldsymbol{\varphi}(\mathbf{x})}$$

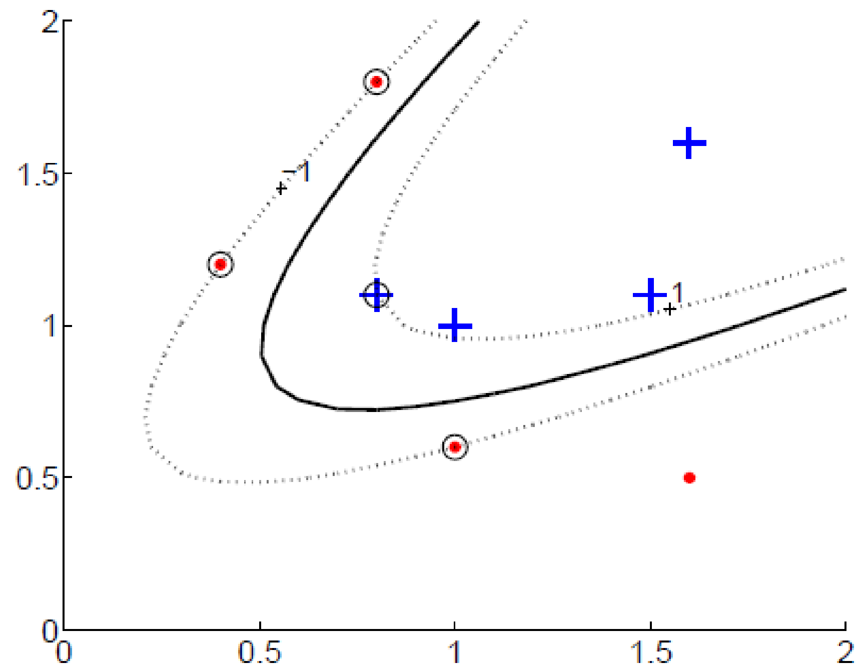
$$g(\mathbf{x}) = \sum_t \alpha^t r^t \boxed{K(\mathbf{x}^t, \mathbf{x})}$$

Inner product in z space replace by a kernel machine in x

Polynomial Kernels

Polynomials of degree q:

$$K\left(\mathbf{x}^t,\mathbf{x}\right)=\left(\mathbf{x}^T\mathbf{x}^t+1\right)^q$$



$$K(\mathbf{x},\mathbf{y})=\left(\mathbf{x}^T\mathbf{y}+1\right)^2$$
$$=\left(x_1y_1+x_2y_2+1\right)^2$$
$$\boxed{=1+2x_1y_1+2x_2y_2+2x_1x_2y_1y_2+x_1^2y_1^2+x_2^2y_2^2}$$
$$\phi(\mathbf{x})=\left[1,\sqrt{2}x_1,\sqrt{2}x_2,\sqrt{2}x_1x_2,x_1^2,x_2^2\right]^T$$

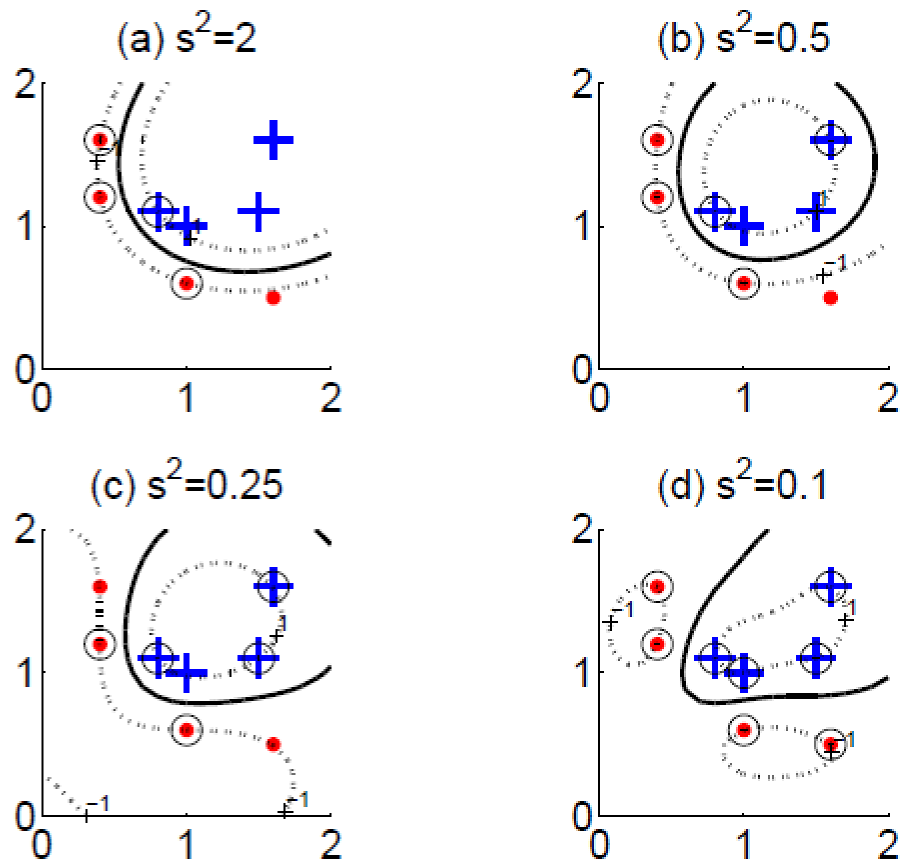
inner product of this basis function

RBF (Gaussian) Kernel

Radial-basis functions:

$$K\left(\mathbf{x}^t,\mathbf{x}\right)=\exp\left[-\frac{\left\|\mathbf{x}^t-\mathbf{x}\right\|^2}{2s^2}\right]$$

Think about this in terms of similarity measures



Defining Kernels

- Kernel “engineering”
- Defining good measures of similarity
- String kernels, graph kernels, image kernels, ...
- Kernel can be ‘designed’

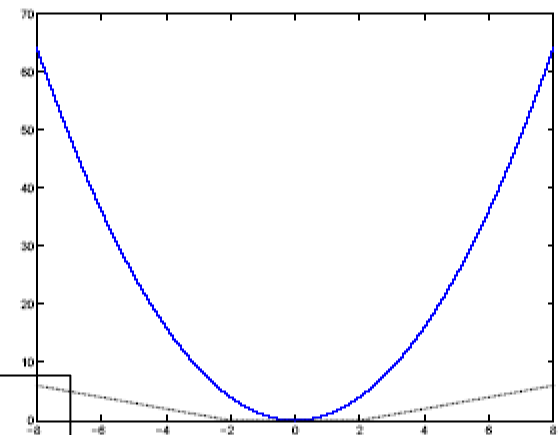
SVM for Regression

- Use a linear model (possibly kernelized)

$$f(\boldsymbol{x})=\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}+w_0$$

- Use the ϵ-sensitive error function

$$e_{\varepsilon}\left(r^t, f\left(\mathbf{x}^t\right)\right)=\begin{cases}0 & \text{if } \left|r^t - f\left(\mathbf{x}^t\right)\right| < \varepsilon \\ \left|r^t - f\left(\mathbf{x}^t\right)\right| - \varepsilon & \text{otherwise}\end{cases}$$



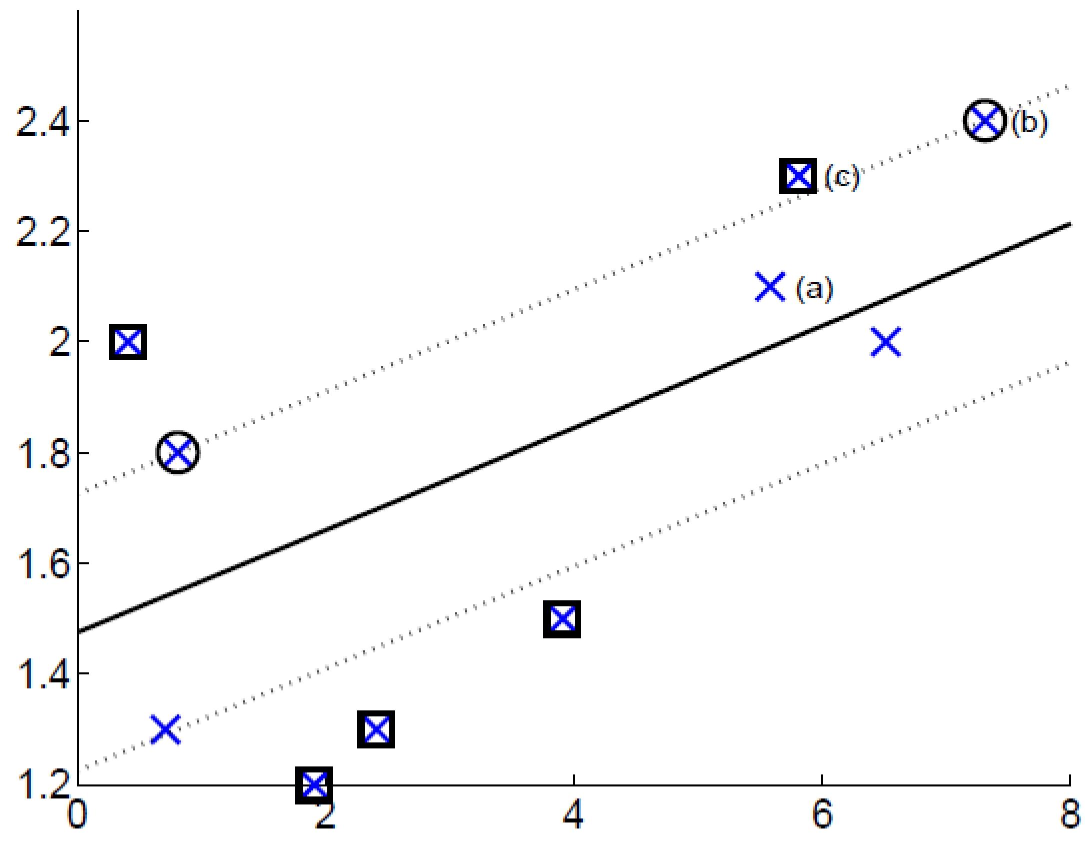
Tolerate an absolute difference

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_t \left(\xi_+^t + \xi_-^t\right)$$

$$r^t - \left(\mathbf{w}^T\mathbf{x} + w_0\right) \leq \varepsilon + \xi_+^t$$

$$\left(\mathbf{w}^T\mathbf{x} + w_0\right) - r^t \leq \varepsilon + \xi_-^t$$

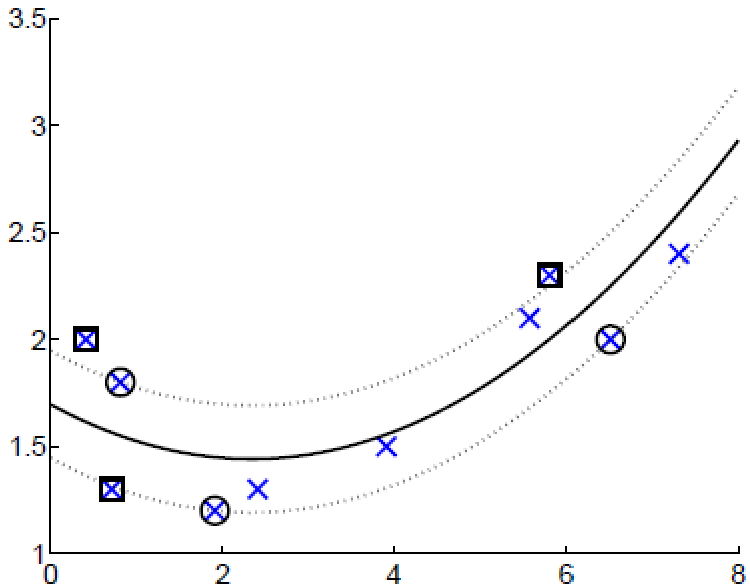
$$\xi_+^t, \xi_-^t \geq 0$$

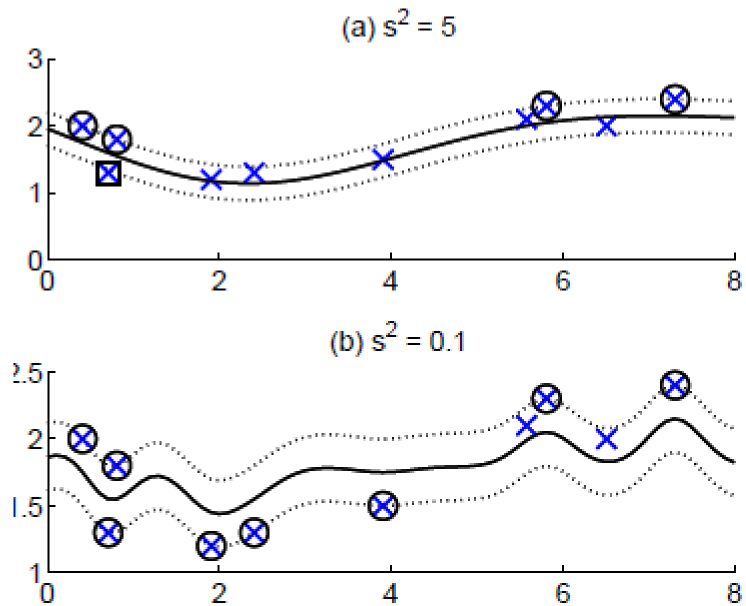


2.4
2.2
2
1.8
1.6
1.4
1.2
0
2
4
6
8
(a)
(b)
(c)

Kernel Regression

- Polynomial kernel
- Gaussian kernel

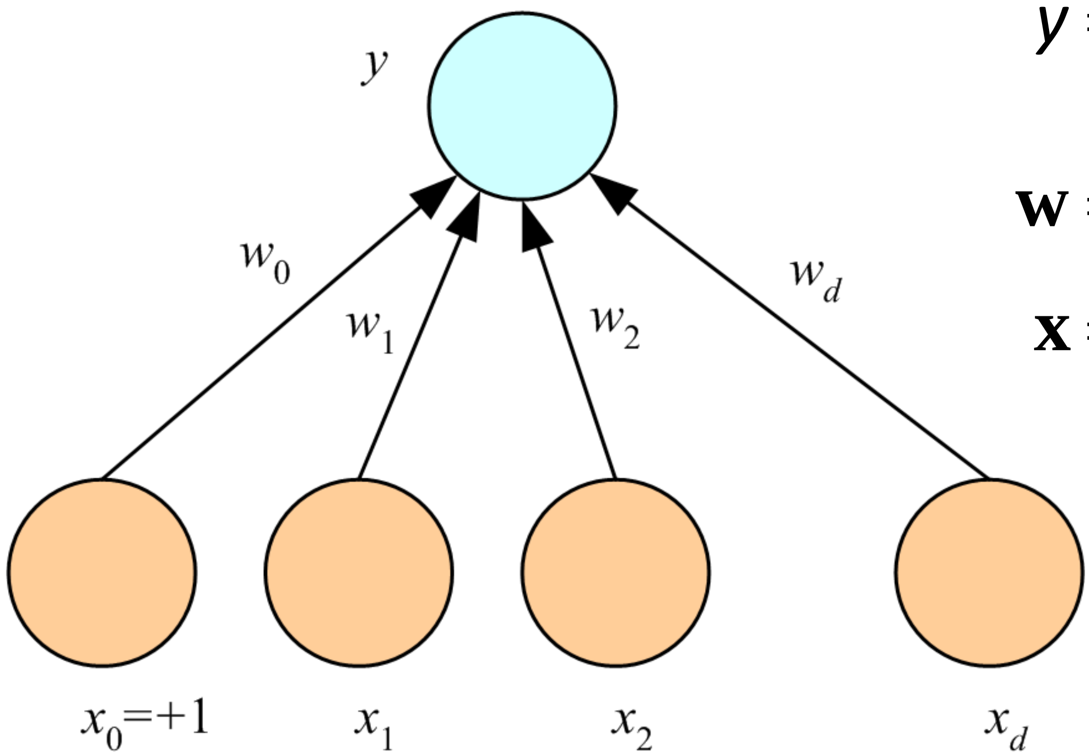




CHAPTER 11:

Multilayer Perceptrons

Perceptron



$$y = \sum_{j=1}^{d} w_j x_j + w_0 = \mathbf{w}^T \mathbf{x}$$

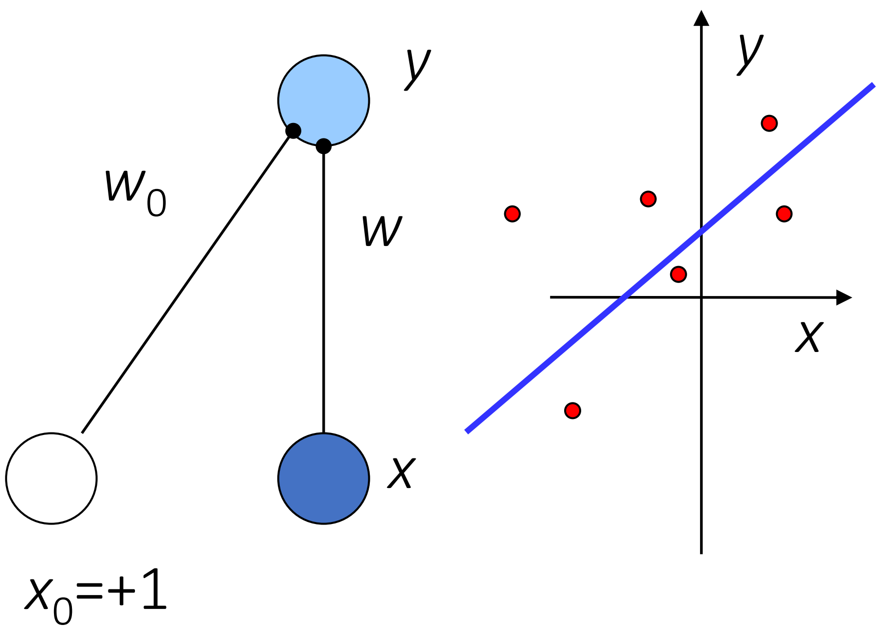
$$\mathbf{w} = [w_0, w_1, \ldots, w_d]^T$$

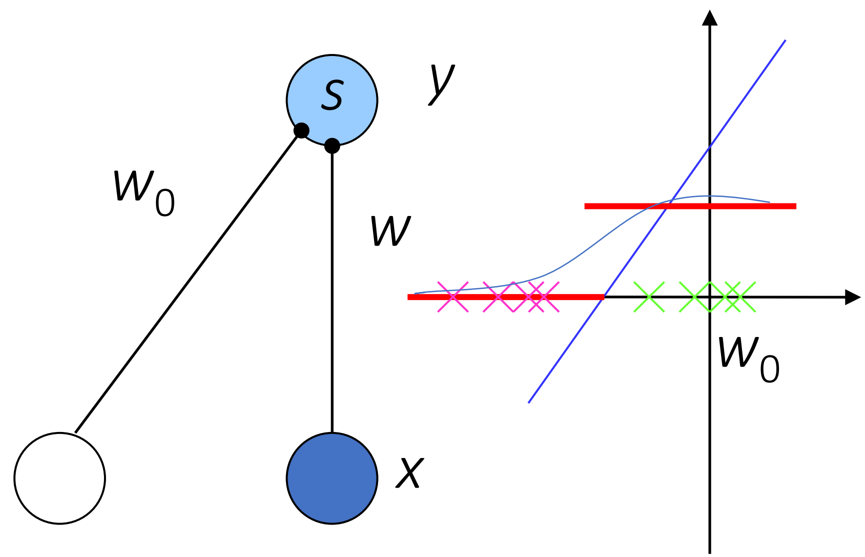
$$\mathbf{x} = [1, x_1, \ldots, x_d]^T$$

(Rosenblatt, 1962)

What a Perceptron Does

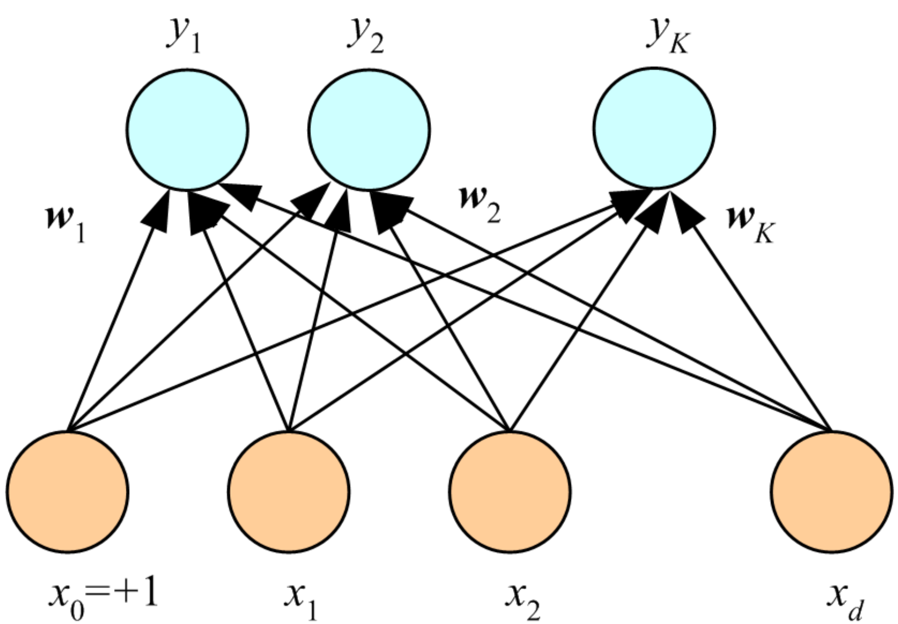
- Regression: $y=wx+w_0$
- Classification: $y=1(wx+w_0>0)$





$$y=\text{sigmoid}\left(o\right)=\frac{1}{1+\exp\left[-\mathbf{w}^{T}\mathbf{x}\right]}$$

K Outputs



Regression:

$$y_i = \sum_{j=1}^{d} w_{ij} x_j + w_{i0} = \mathbf{w}_i^T \mathbf{x}$$

$$\mathbf{y} = \mathbf{W}\mathbf{x}$$

Classification:

$$o_i = \mathbf{w}_i^T \mathbf{x}$$

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

choose C_i

if $y_i = \max_k y_k$

Training a Perceptron: Regression

- Regression (Linear output):

$$E^t\left(\mathbf{w} \mid \mathbf{x}^t, r^t\right) = \frac{1}{2}\left(r^t - y^t\right)^2 = \frac{1}{2}\left[r^t - \left(\mathbf{w}^T \mathbf{x}^t\right)\right]^2$$

$$\Delta w_j^t = \eta\left(r^t - y^t\right)x_j^t$$

Classification

- Single sigmoid output

$$y^t = \text{sigmoid}\left(\mathbf{w}^T \mathbf{x}^t\right)$$
$$E^t\left(\mathbf{w} \mid \mathbf{x}^t, \mathbf{r}^t\right) = -r^t \log y^t - \left(1 - r^t\right) \log\left(1 - y^t\right)$$
$$\Delta w_j^t = \eta\left(r^t - y^t\right) x_j^t$$

- *K*>2 softmax outputs

$$y^t = \frac{\exp \mathbf{w}_i^T \mathbf{x}^t}{\sum_k \exp \mathbf{w}_k^T \mathbf{x}^t} \qquad E^t\left(\{\mathbf{w}_i\}_i \mid \mathbf{x}^t, \mathbf{r}^t\right) = -\sum_i r_i^t \log y_i^t$$
$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right) x_j^t$$

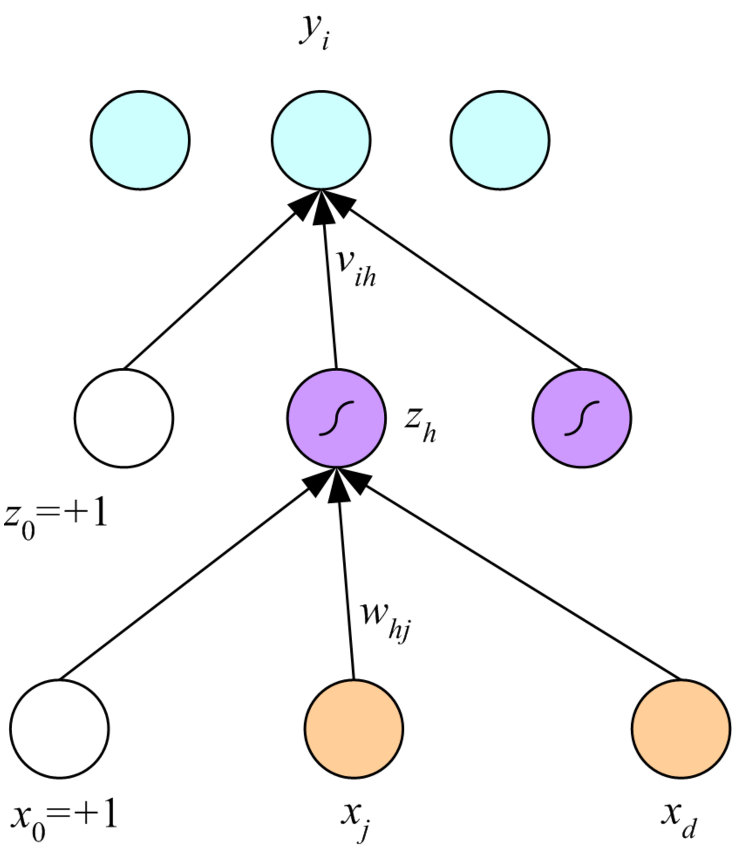
Training

- Online (instances seen one by one) vs batch (whole sample) learning:
  - No need to store the whole sample
  - Problem may change in time
  - Wear and degradation in system components
- Stochastic gradient-descent: Update after a single pattern
- Generic update rule (LMS rule):

$$\Delta w_{ij}^t = \eta\left(r_i^t - y_i^t\right)x_j^t$$

$$\text{Update} = \text{LearningFactor} \cdot \left(\text{DesiredOutput} - \text{ActualOutput}\right) \cdot \text{Input}$$

Multilayer Perceptrons



$$y_i = \mathbf{v}_i^T \mathbf{z} = \sum_{h=1}^{H} v_{ih} z_h + v_{i0}$$

$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right)$$

$$= \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

(Rumelhart et al., 1986)