Review of Intro to ML

Jan 04, 2022

Logistics

- Final Exam (Jan 10th)
- 1 A4 cheat sheet, handwritten
- Cover all topics taught in class

CHAPTER 3:

Bayesian Decision Theory

Classification

Observation → measurable input

- Credit scoring: Inputs are income and savings.

  Output is low-risk vs high-risk
- Input: $\boldsymbol{x} = [x_1, x_2]^T$,Output: C $\in$ {0,1}
- Prediction:

Bernoulli rv
Condition on
x1, x2

$$\text{choose}\begin{cases} C = 1 \text{ if } P(C=1|x_1,x_2) \ > \ 0.5 \\ C = 0 \text{ otherwise} \end{cases}$$

or

$$\text{choose}\begin{cases} C = 1 \text{ if } P(C=1|x_1,x_2) \ > P(C=0|x_1,x_2) \\ C = 0 \text{ otherwise} \end{cases}$$

ERROR? 1 – max (P(C=1)|x1,x2), P(C=0|x1, x2))

Maximum A-posterior decision rule (find the class with highest probability)

Guarantee the error is smallest as measured in terms or probability

Bayes' Rule

What is C (high risk/low risk) given the two inputs

prior *likelihood*

posterior

$$P(C \mid \mathbf{x}) = \frac{P(C)\,p(\mathbf{x} \mid C)}{p(\mathbf{x})}$$

evidence

Total probability theorem

$$P(C=0) + P(C=1) = 1$$

$$p(\mathbf{x}) = p(\mathbf{x} \mid C=1)P(C=1) + p(\mathbf{x} \mid C=0)P(C=0)$$

$$p(C=0 \mid \mathbf{x}) + P(C=1 \mid \mathbf{x}) = 1$$

Bayes' Rule: K>2 Classes

$$P(C_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid C_i)P(C_i)}{p(\mathbf{x})}$$

$$= \frac{p(\mathbf{x} \mid C_i)P(C_i)}{\sum_{k=1}^{K} p(\mathbf{x} \mid C_k)P(C_k)}$$

$P(C_i) \geq 0$ and $\sum_{i=1}^{K} P(C_i) = 1$

choose C_i if $P(C_i \mid \mathbf{x}) = \max_k P(C_k \mid \mathbf{x})$

Make a 'classification' decision by looking at the posterior distribution

Losses and Risks

- Actions: α_i
  - This concept is important in cases when 'loss' associated to each action may not be the same
  - Finance, medical, …so on
- Loss of α_i when the state is C_k : λ_{ik}
- **Expected risk** (Duda and Hart, 1973)

Loss of making a decision to assign input to class i, when the true class is k

Minimizing expected risk by picking that action i

$$R(\alpha_i \mid \mathbf{x}) = \sum_{k=1}^{K} \lambda_{ik} P(C_k \mid \mathbf{x})$$

$$\text{choose } \alpha_i \text{ if } R(\alpha_i \mid \mathbf{x}) = \min_k R(\alpha_k \mid \mathbf{x})$$

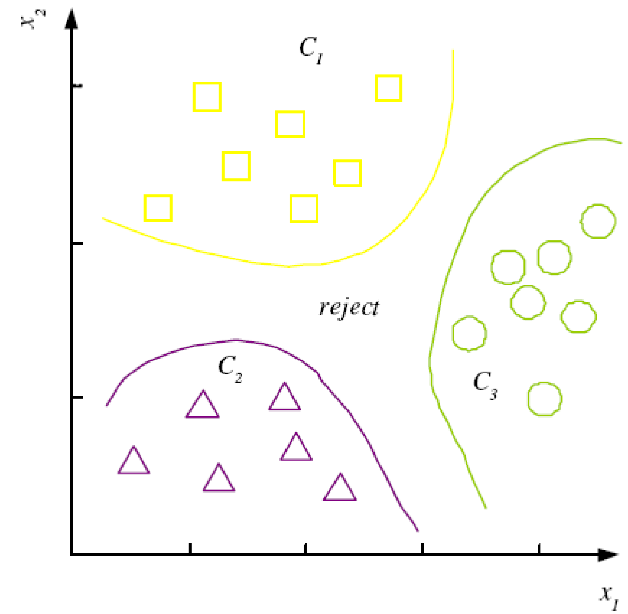
Discriminant Functions

Classification rule: pick one such function that maximizes

choose C_i if $\boxed{g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})}$

$$g_i(\mathbf{x}) = \begin{cases} -R(\alpha_i \mid \mathbf{x}) \\ P(C_i \mid \mathbf{x}) \\ p(\mathbf{x} \mid C_i) P(C_i) \end{cases}$$

Three different discriminant functions



K decision regions $\mathcal{R}_1, \ldots, \mathcal{R}_K$

$$\mathcal{R}_i = \{\mathbf{x} \mid g_i(\mathbf{x}) = \max_k g_k(\mathbf{x})\}$$

Divides feature space into K region
For those inputs x, find the function that give the largest value (use that function to carve out a region

K=2 Classes

- Dichotomizer (K=2) vs Polychotomizer (K>2)
- $g(\boldsymbol{x}) = g_1(\boldsymbol{x}) - g_2(\boldsymbol{x})$

$$\text{choose } \begin{cases} C_1 \text{ if } g(\mathbf{x}) > 0 \\ C_2 \text{ otherwise} \end{cases}$$

- *Log odds:*

A discriminant function

$$\log \frac{P(C_1 \mid \mathbf{x})}{P(C_2 \mid \mathbf{x})}$$

Needs only 1 value to make a decision

CHAPTER 4:

Parametric Methods

Statistics based ML method

Why do we need this?

Using Bayes Decision Theory for Classification

$$P(C_i \mid \mathbf{x}) = \frac{\boxed{p(\mathbf{x} \mid C_i)} P(C_i)}{p(\mathbf{x})}$$

$$= \frac{p(\mathbf{x} \mid C_i) P(C_i)}{\sum_{k=1}^{K} p(\mathbf{x} \mid C_k) P(C_k)}$$

The distribution function of P(x) when X is coming from a class C_i

Need this probability to be used as discriminant function

Collect data
Use training set
Learn the discriminant function (essentially estimate the parameters)

$P(C_i) \geq 0$ and $\sum_{i=1}^{K} P(C_i) = 1$

choose C_i if $P(C_i \mid \mathbf{x}) = \max_k P(C_k \mid \mathbf{x})$

Maximum Likelihood Estimator

- Likelihood of θ given the sample $\mathcal{X}$

$$l(\vartheta|\mathcal{X}) = p(\mathcal{X}|\vartheta) = \prod_t p(x^t|\vartheta)$$

Estimate parameter of the model using X with this criterion

Joint factors to product (assume iid samples)

- Log likelihood

$$\mathcal{L}(\vartheta|\mathcal{X}) = \log l(\vartheta|\mathcal{X}) = \sum_t \log p(x^t|\vartheta)$$

- Maximum likelihood estimator (MLE)

$$\vartheta^* = \operatorname{argmax}_\vartheta \mathcal{L}(\vartheta|\mathcal{X})$$

Parametric Classification

-once done (learning), you also obtain the actual 'discriminant' function that can be used for classification

$$g_i(x) = p(x \mid C_i)P(C_i)$$

or

$$g_i(x) = \log p(x \mid C_i) + \log P(C_i)$$

This is your discriminant function for classification

$$\boxed{\begin{aligned} p(x \mid C_i) &= \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(x-\mu_i)^2}{2\sigma_i^{\,2}}\right] \\ g_i(x) &= -\frac{1}{2}\log 2\pi - \log \sigma_i - \frac{(x-\mu_i)^2}{2\sigma_i^{\,2}} + \log P(C_i) \end{aligned}}$$

The actual implementation when using Gaussian distribution

An example of learning for two class problem

- Given the sample $\mathcal{X} = \{x^t, r^t\}_{t=1}^N$

$$x \in \Re \qquad r_i^t = \begin{cases} 1 \text{ if } x^t \in C_i \\ 0 \text{ if } x^t \in C_j, j \neq i \end{cases}$$

- ML estimates are

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad m_i = \frac{\sum_t x^t r_i^t}{\sum_t r_i^t} \quad s_i^2 = \frac{\sum_t \left(x^t - m_i\right)^2 r_i^t}{\sum_t r_i^t}$$

- Discriminant

Once done, use the following function on test set

$$g_i(x) = -\frac{1}{2}\log 2\pi - \log s_i - \frac{\left(x - m_i\right)^2}{2 s_i^2} + \log \hat{P}(C_i)$$

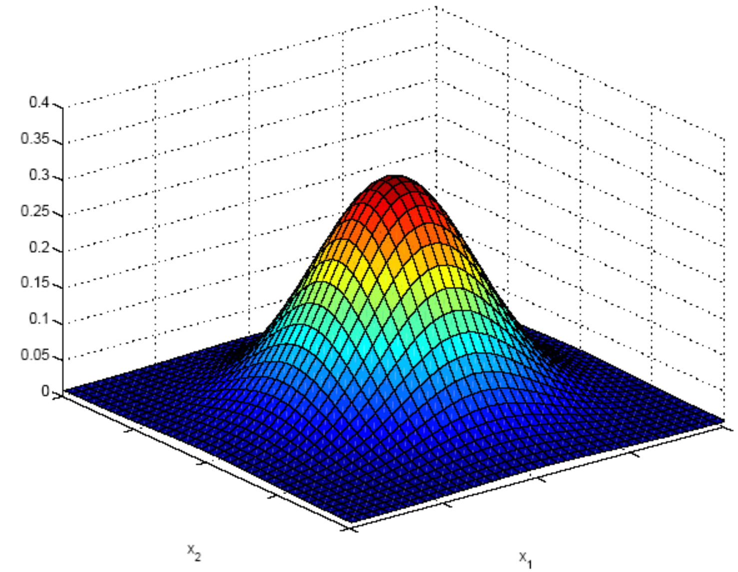
Input x (test sample) for each i, look at the class label i, for the one that is max, pick that class i

Multivariate Data

- Multiple measurements (sensors)
- d inputs/features/attributes: d-variate
- N instances/observations/examples

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

Multivariate Normal Distribution



$$\mathbf{x} \sim \mathcal{N}_d(\boldsymbol{\mu}, \Sigma)$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

Multivariate Normal Distribution

Use of inverse variance

- Larger variance adds less distance
- Correlated variable contribute less

- Mahalanobis distance: $(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})$

  measures the distance from $\boldsymbol{x}$ to $\boldsymbol{\mu}$ in terms of Σ (normalizes for difference in variances and correlations)

- Bivariate: $d = 2$

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}$$

$$\boxed{\begin{gathered} p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(z_1^2 - 2\rho z_1 z_2 + z_2^2\right)\right] \\ z_i = (x_i - \mu_i)/\sigma_i \end{gathered}}$$

Parametric Classification

- If $p\ (\boldsymbol{x} \mid C_i) \sim N\ (\boldsymbol{\mu}_i, \Sigma_i)$

$$\boxed{p(\mathbf{x} \mid C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T \Sigma_i^{-1} (\mathbf{x}-\boldsymbol{\mu}_i)\right]}$$

- Discriminant functions

$$\boxed{\begin{aligned} g_i(\mathbf{x}) &= \log p(\mathbf{x} \mid C_i) + \log P(C_i) \\ &= -\frac{d}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_i| - \frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i) + \log P(C_i) \end{aligned}}$$

Estimation of Parameters

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N}$$

$$\mathbf{m}_i = \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T}{\sum_t r_i^t}$$

$$g_i(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}(\mathbf{x}-\mathbf{m}_i)^T \mathbf{S}_i^{-1}(\mathbf{x}-\mathbf{m}_i) + \log \hat{P}(C_i)$$

Different $\mathbf{S}_i$

Quadratic form

Quadratic discriminant

$$g_i(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{S}_i| - \frac{1}{2}\left(\mathbf{x}^T\mathbf{S}_i^{-1}\mathbf{x} - 2\mathbf{x}^T\mathbf{S}_i^{-1}\mathbf{m}_i + \mathbf{m}_i^T\mathbf{S}_i^{-1}\mathbf{m}_i\right) + \log\hat{P}(C_i)$$

$$\boxed{= \mathbf{x}^T\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^T\mathbf{x} + w_{i0}}$$

where

$$\mathbf{W}_i = -\frac{1}{2}\mathbf{S}_i^{-1}$$

$$\mathbf{w}_i = \mathbf{S}_i^{-1}\mathbf{m}_i$$

$$w_{i0} = -\frac{1}{2}\mathbf{m}_i^T\mathbf{S}_i^{-1}\mathbf{m}_i - \frac{1}{2}\log|\mathbf{S}_i| + \log\hat{P}(C_i)$$

Common Covariance Matrix S

- Shared common sample covariance **S** for all class

$$\mathbf{S} = \sum_i \hat{P}(C_i)\mathbf{S}_i$$

- Discriminant reduces to

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x} - \boldsymbol{m}_i)^T \mathbf{S}^{-1}(\boldsymbol{x} - \boldsymbol{m}_i) + \log \hat{P}(C_i)$$

which is a <u>linear discriminant</u>

$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

where

$$\mathbf{w}_i = \mathbf{S}^{-1}\mathbf{m}_i \quad w_{i0} = -\frac{1}{2}\mathbf{m}_i^T \mathbf{S}^{-1}\mathbf{m}_i + \log \hat{P}(C_i)$$

Diagonal **S**

- When x_j $j = 1,..d$, are independent, $\sum$ is diagonal
  $p(x|C_i) = \prod_j p(x_j | C_i)$ (Naive Bayes' assumption)

$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{\boxed{x_j^t - m_{ij}}}{s_j}\right)^2 + \log \hat{P}(C_i)$$

Classify based on weighted Euclidean distance (in s_j units) to the nearest mean

Diagonal **S**, equal variances

- Nearest mean classifier: Classify based on Euclidean distance to the nearest mean

$$\boxed{\begin{aligned} g_i(\mathbf{x}) &= -\frac{\|\mathbf{x}-\mathbf{m}_i\|^2}{2s^2} + \log \hat{P}(C_i) \\ &= -\frac{1}{2s^2}\sum_{j=1}^{d}\left(x_j^t - m_{ij}\right)^2 + \log \hat{P}(C_i) \end{aligned}}$$

- Each mean can be considered a prototype or template and this is template matching

CHAPTER 6:

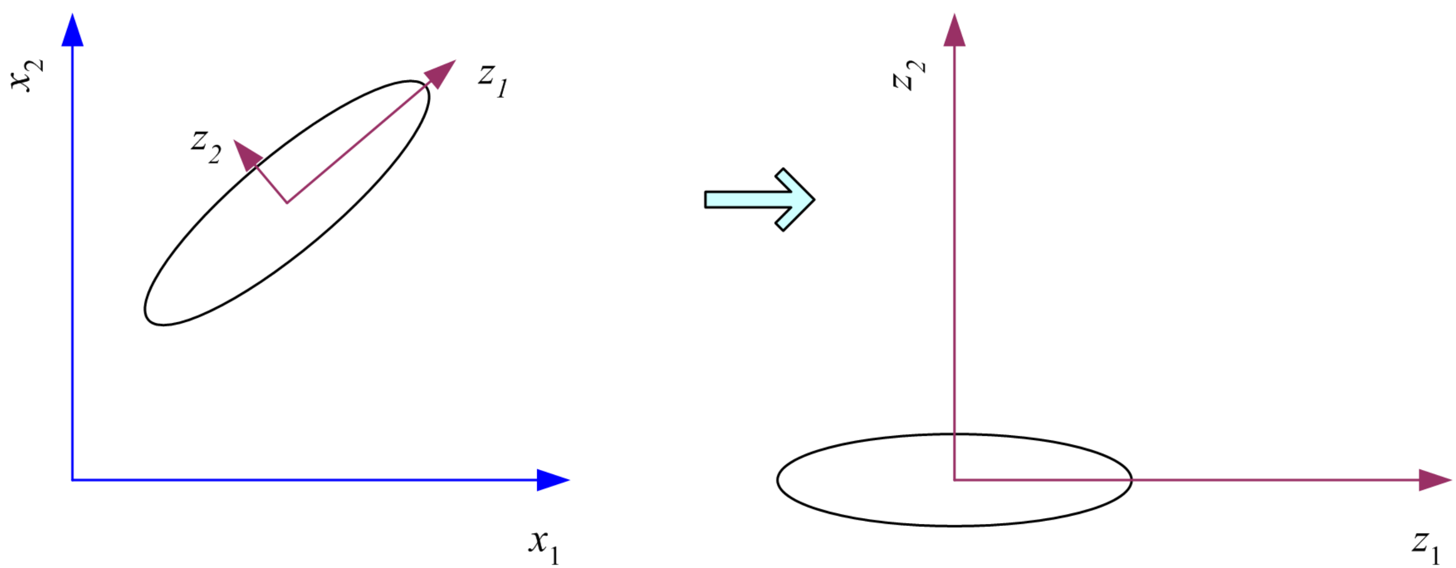
Dimensionality Reduction

What PCA does

$$z = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{m})$$

where the columns of $\mathbf{W}$ are the eigenvectors of $\sum$ and $\boldsymbol{m}$ is sample mean

Centers the data at the origin and rotates the axes

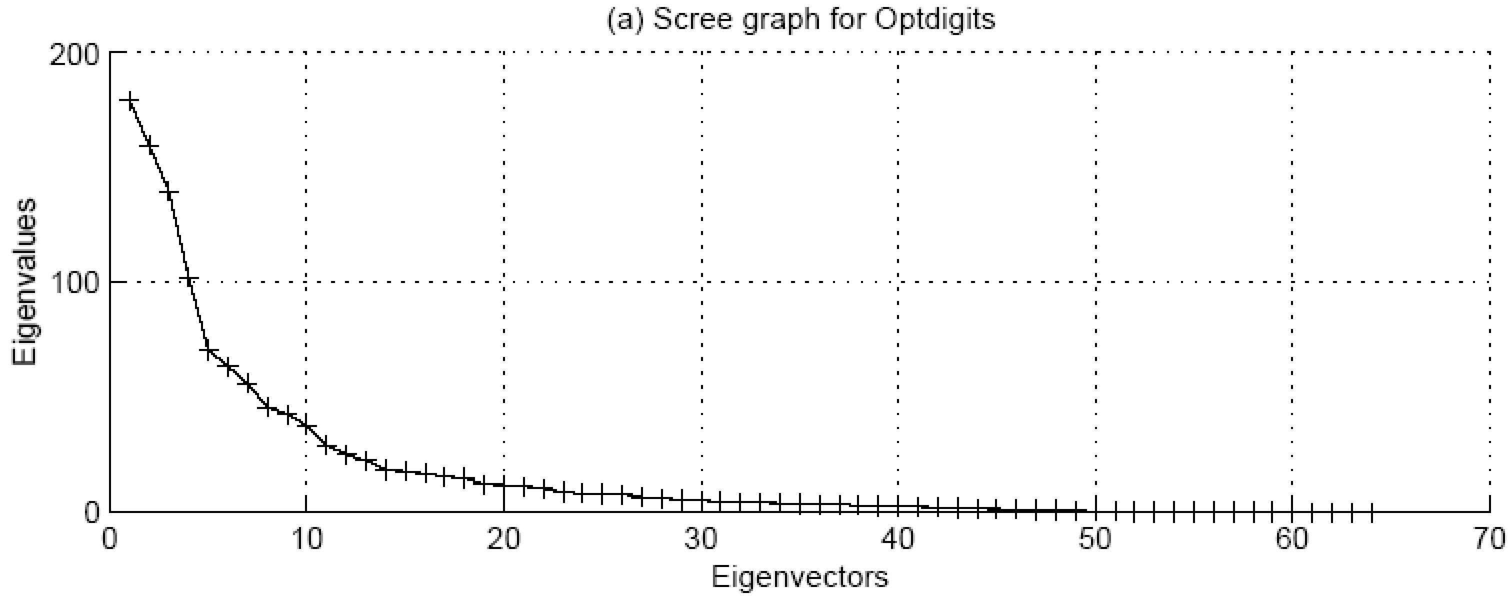


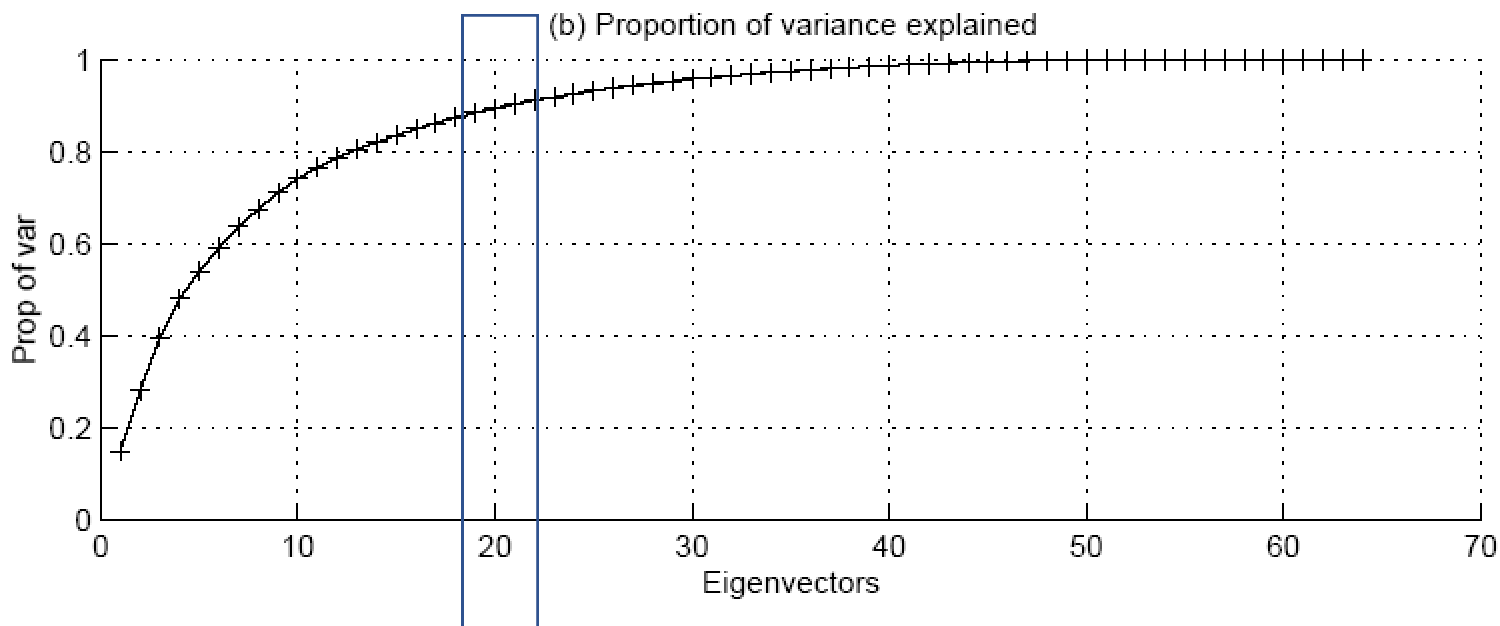
How to choose k ?

- Proportion of Variance (PoV) explained

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

  when λ_i (eigenvalues) are sorted in descending order
- Typically, stop at PoV>0.9
- Scree graph plots of PoV vs *k*, stop at "elbow"
  - *adding another eigenfactor does not add much variances*


(a) Scree graph for Optdigits
Eigenvalues
200
100
0
0
10
20
30
40
50
60
70
Eigenvectors


(b) Proportion of variance explained
Prop of var
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
Eigenvectors

After PCA

- From d-dimension to k-dimension
  - K<d, parametric discriminant function rely on fewer parameters (due to lesser features)
  - Since these features are projected to be uncorrelated (orthogonal is you assume Normal distribution)
    - The multivariate Gaussian’s covariance matrix can now be assumed to be diagonal

- PCA is an unsupervised method of dimension reduction -> does not require the knowledge of label, just the data

CHAPTER 7:

Clustering

Classes vs. Clusters

- Supervised: $X = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_t$
- Classes C_i $i=1,\ldots,K$

$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x} \mid C_i) P(C_i)$$

where $p(\boldsymbol{x}|C_i) \sim N(\boldsymbol{\mu}_i, \Sigma_i)$

- $\Phi = \{P(C_i), \boldsymbol{\mu}_i, \Sigma_i\}^K_{i=1}$

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad \mathbf{m}_i = \frac{\sum_t r_i^t \mathbf{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\mathbf{x}^t - \mathbf{m}_i)(\mathbf{x}^t - \mathbf{m}_i)^T}{\sum_t r_i^t}$$

- Unsupervised : $X = \{\boldsymbol{x}^t\}_t$
- Clusters G_i $i=1,\ldots,k$

$$p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid G_i) P(G_i)$$

where $p(\boldsymbol{x}|G_i) \sim N(\boldsymbol{\mu}_i, \Sigma_i)$

- $\Phi = \{P(G_i), \boldsymbol{\mu}_i, \Sigma_i\}^k_{i=1}$

Labels $\boldsymbol{r}^t_i$?

Through clustering

Mixture model

$$p(\mathbf{x}) = \sum_{i=1}^{k} p(\mathbf{x} \mid G_i) P(G_i)$$

where G_i the components/groups/clusters,

$P(G_i)$ mixture proportions (priors),

$p(\boldsymbol{x} \mid G_i)$ component densities

If known, the data sample can find it's associated 'components/ groups /cluster'

Expectation-Maximization (EM)

- Log likelihood of a mixture model

$$\mathcal{L}(\Phi \mid \mathcal{X}) = \log \prod_t p(\mathbf{x}^t \mid \Phi)$$

$$= \sum_t \log \sum_{i=1}^{k} p(\mathbf{x}^t \mid G_i) P(G_i)$$

Unknown, (we don't know which cluster the sample belongs to)

No analytical solution when learning this model

EM Algorithm, core concept

- Assume there exist hidden variables z, which when known, make optimization much simpler
- Complete likelihood, $L_C(\Phi|X,Z)$, in terms of ***x*** and ***z***
- Incomplete likelihood, $L(\Phi|X)$, in terms of ***x***

E- and M-steps

Model parameter

Iterate the two steps

1. E-step: Estimate *z* given X and current Φ
2. M-step: Find new Φ' given *z*, X, and old Φ.

$$\text{E-step}: \mathcal{Q}\left(\Phi \mid \Phi^{l}\right) = E\left[\mathcal{L}_C(\Phi \mid \mathcal{X}, \mathcal{Z}) \mid \mathcal{X}, \Phi^{l}\right]$$

$$\text{M-step}: \Phi^{l+1} = \arg\max_{\Phi} \mathcal{Q}\left(\Phi \mid \Phi^{l}\right)$$

An increase in Q function increases incomplete likelihood

$$\mathcal{L}\left(\Phi^{l+1} \mid \mathcal{X}\right) \geq \mathcal{L}\left(\Phi^{l} \mid \mathcal{X}\right)$$

There is proof beyond this class

Complete steps for GMM

- If assume Gaussian component (each mixture is a Gaussian distribution)

$$P(\mathcal{G}_i) = \frac{\sum_t h_i^t}{N} \qquad \mathbf{m}_i^{l+1} = \frac{\sum_t h_i^t \mathbf{x}^t}{\sum_t h_i^t}$$

$$\mathbf{S}_i^{l+1} = \frac{\sum_t h_i^t (\mathbf{x}^t - \mathbf{m}_i^{l+1})(\mathbf{x}^t - \mathbf{m}_i^{l+1})^T}{\sum_t h_i^t}$$

Soft assignment of a sample to a class

$$h_i^t = \frac{\pi_i |\mathbf{S}_i|^{-1/2} \exp[-(1/2)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T \mathbf{S}_i^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_i)]}{\sum_j \pi_j |\mathbf{S}_j|^{-1/2} \exp[-(1/2)(\boldsymbol{x}^t - \boldsymbol{m}_j)^T \mathbf{S}_j^{-1} (\boldsymbol{x}^t - \boldsymbol{m}_j)]}$$

Practice implementation of GMM

- EM is initialized by k-means -> so you get initial parameter

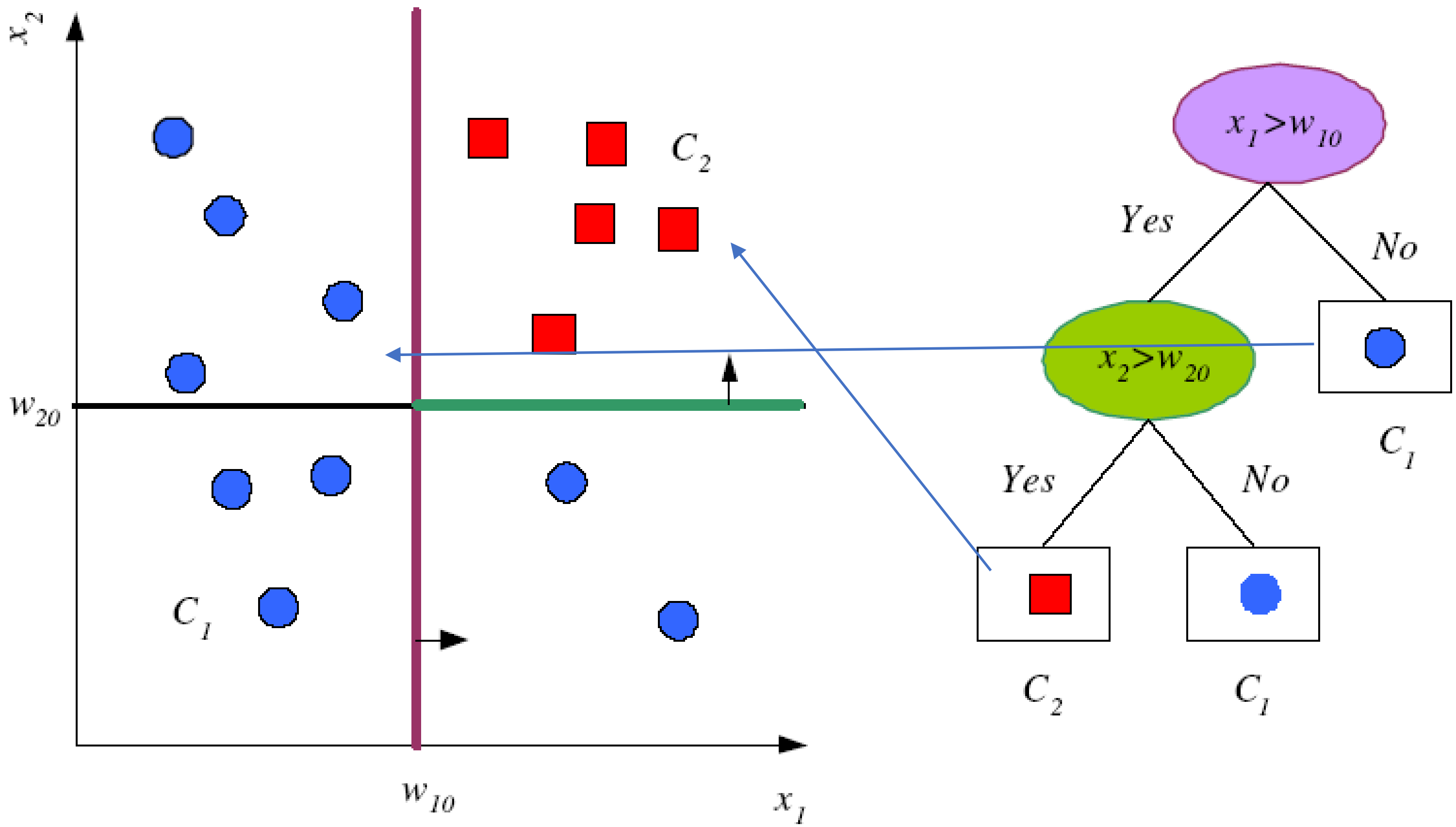
- Once done k-mean, use m_i and samples associated with each cluster as to estimate the initial parameters used for mixture of Gaussian distributions

- Once done-learning, the GMM model can be used for ‘clustering’ of samples x^t (compute h_i^t)

CHAPTER 9:

Decision Trees

Tree Uses Nodes and Leaves



Properties

- Non-parametric method
- Interpretability
  - Can be thought of as an implementation of various IF-THEN rules
  - Easy to understand what is going on in the decision making

- Each node implements a test function $f_m(x)$ with discrete outcomes labeling the branches
  - Training incidences travel through the tree/branches until reaching leaves

A tree: Divide and Conquer Strategy

- Internal decision nodes
  - Univariate: Uses a single attribute, x_i
    - Numeric x_i : Binary split : $x_i > w_m$
    - Discrete x_i : n-way split for n possible values
  - Multivariate: Uses all attributes, $\boldsymbol{x}$
- Leaves
  - Classification: Class labels, or proportions
  - Regression: Numeric; r average, or local fit
- Learning is **greedy**; find the best split from the root and work its way down recursively (Breiman et al, 1984; Quinlan, 1986, 1993)

Classification Trees (ID3,CART,C4.5)

- At node m, N_m instances reach m, N^i_m belong to C_i

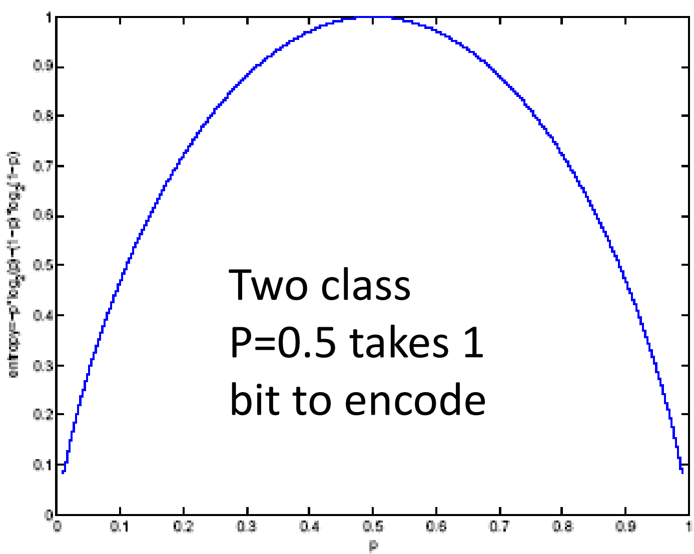
$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

If split is pure, there is no need to split anymore

- Node m is pure if p^i_m is 0 or 1
- Measure of impurity is entropy

$$\mathcal{I}_m = -\sum_{i=1}^{K} p^i_m \log_2 p^i_m$$

- Entropy: # of bits need to code



Other measures of two class?

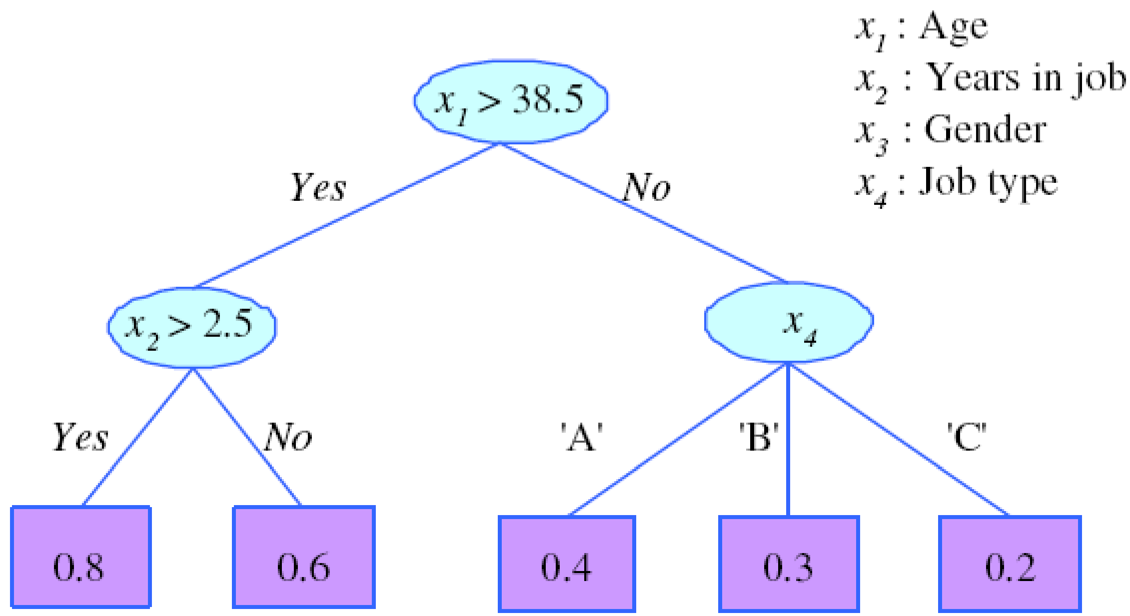
Properties:

- $\Phi(½, ½) >= \Phi(p, 1\text{-}p)$ for any p in [0,1].
- $\Phi(0,1)= \Phi(1,0) = 0$
- $\Phi(p, 1\text{-}p)$ is increasing in p on [0, ½] and decreasing in p on [1/2, 1]

- Gini index (Breiman et al. 1984)
  - $\Phi(p, 1\text{-}p) = 2p(1\text{-}p)$
- Misclassification error
  - $\Phi(p, 1\text{-}p) = 1\text{-} \max(p, 1\text{-}p)$

Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

CHAPTER 10:

Linear Discrimination

Likelihood- vs. Discriminant-based Classification

- Likelihood-based: Assume a model for $p(\boldsymbol{x}|C_i)$, use Bayes' rule to calculate $P(C_i|\boldsymbol{x})$

  $$g_i(\boldsymbol{x}) = \log P(C_i|\boldsymbol{x})$$

Just any form of equations

- Discriminant-based: Assume a **model** for $g_i(\boldsymbol{x}|\Phi_i)$; not density estimation
- Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries
- Inductive bias come from your assumption of boundary not the density itself
- Knowing **how to separate** is more important (easier?) than knowing the underlying data distribution

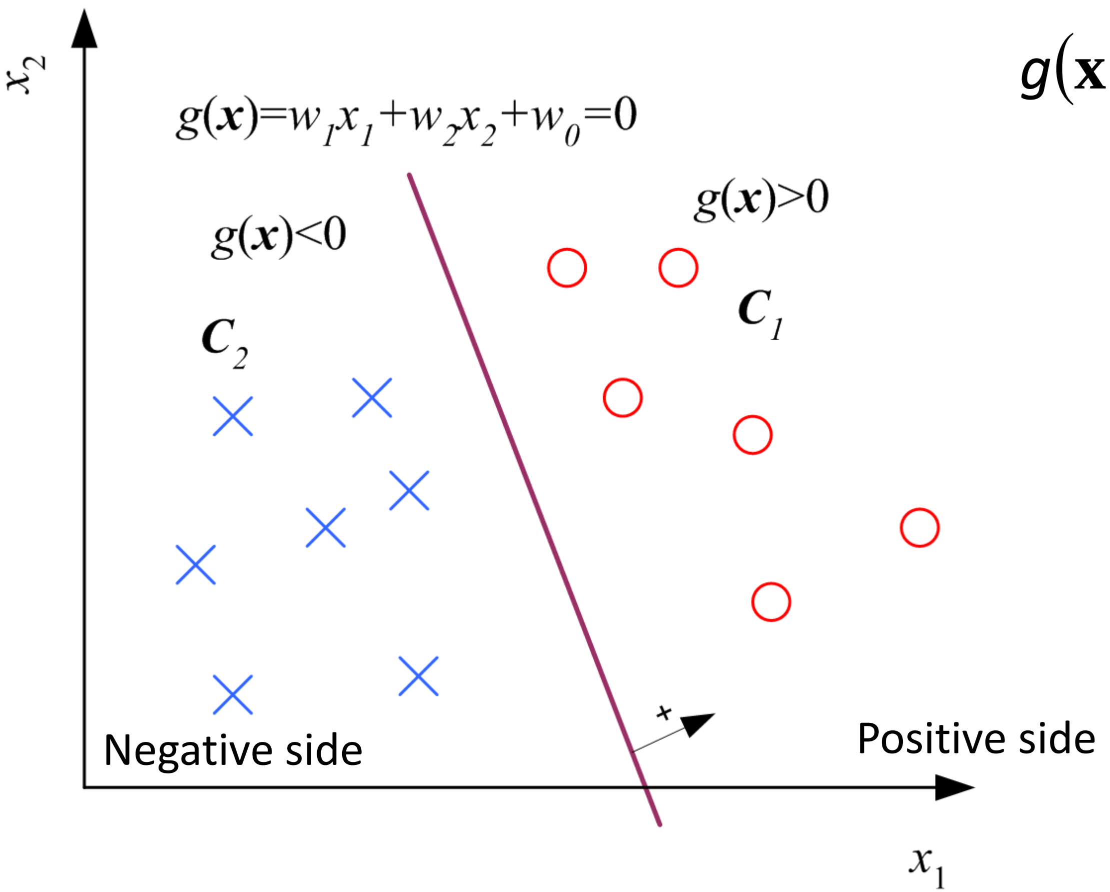
Linear Discriminant

- Linear discriminant function (assuming a linear separation):

$$g_i(\mathbf{x}|\mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0} = \sum_{j=1}^{d} w_{ij} x_j + w_{i0}$$

- Advantages:
  - Simple: O(d) space/computation
  - Knowledge extraction: Weighted sum of attributes; positive/negative weights, magnitudes
  - Optimal when $p(\boldsymbol{x}|C_i)$ are <u>Gaussian with shared cov matrix</u>; <u>useful</u> when classes are (almost) linearly separable

Two Classes



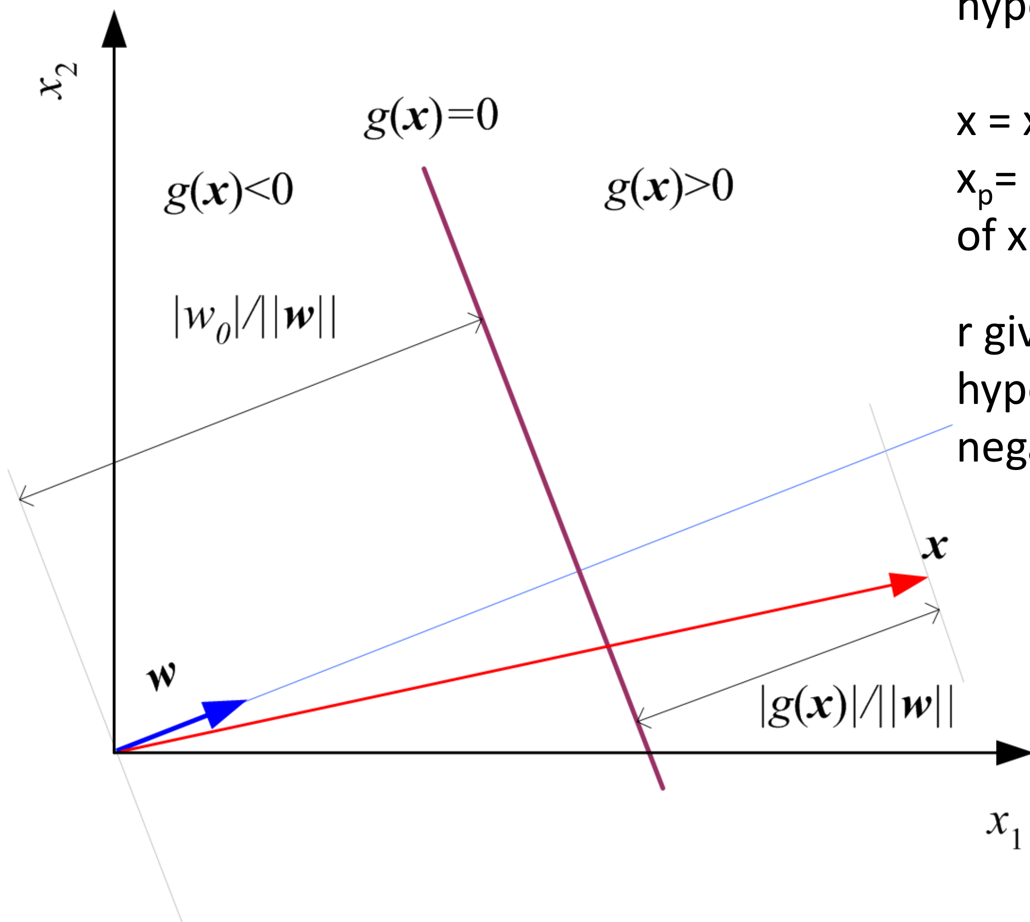
$$
\begin{aligned}
g(\mathbf{x}) &= g_1(\mathbf{x}) - g_2(\mathbf{x}) \\
&= \left(\mathbf{w}_1^T\mathbf{x} + w_{10}\right) - \left(\mathbf{w}_2^T\mathbf{x} + w_{20}\right) \\
&= \left(\mathbf{w}_1 - \mathbf{w}_2\right)^T\mathbf{x} + \left(w_{10} - w_{20}\right) \\
&= \mathbf{w}^T\mathbf{x} + w_0
\end{aligned}
$$

Threshold

Weight vector

$$\text{choose } \begin{cases} C_1 & \text{if } g(\mathbf{x}) > 0 \\ C_2 & \text{otherwise} \end{cases}$$

Geometry

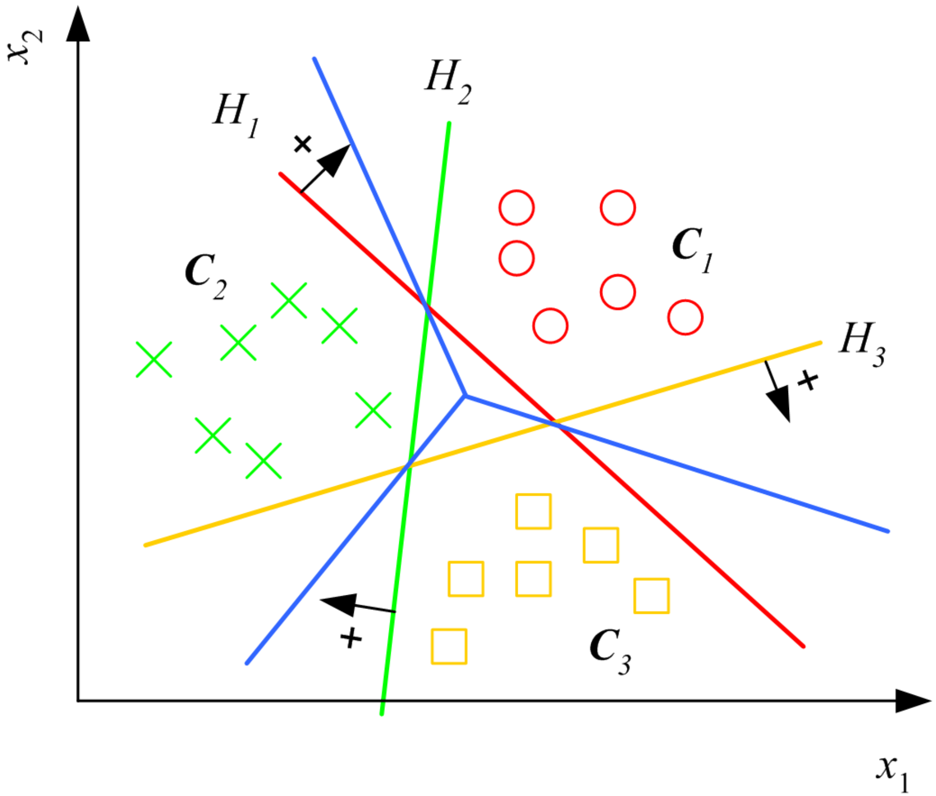


For x1 x2 on decision plane
$w^T(x1-x2) = 0$
w is normal to any vector lying on the decision hyperplane

$x = x_p + r\, w/||w||$
x_p= is the normal projection of x onto hyperplane

r gives us distance from x to hyperplane (positive, negative)

Multiple Classes



$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

Classes are
linearly separable
if
g(i) positive for only one class
More realistic setting:

Choose C_i if

$$g_i(\mathbf{x}) = \max_{j=1}^{K} g_j(\mathbf{x})$$

From Discriminants to Posteriors

When $p\ (\boldsymbol{x} \mid C_i) \sim N\ (\boldsymbol{\mu}_i, \Sigma)$

$$g_i(\mathbf{x} \mid \mathbf{w}_i, w_{i0}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}$$

$$\mathbf{w}_i = \Sigma^{-1}\mu_i \quad w_{i0} = -\frac{1}{2}\mu_i^T \Sigma^{-1}\mu_i + \log P(C_i)$$

$$y \equiv P(C_1 \mid \mathbf{x}) \text{ and } P(C_2 \mid \mathbf{x}) = 1 - y$$

$$\text{choose } C_1 \text{ if} \begin{cases} y > 0.5 \\ y/(1-y) > 1 \\ \log[y/(1-y)] > 0 \end{cases} \text{ and } C_2 \text{ otherwise}$$

Log(y/(1-y))-> logit transform
Log odds of y

$$\text{logit}\left(P(C_1|\mathbf{x})\right) = \log\frac{P(C_1|\mathbf{x})}{1-P(C_1|\mathbf{x})} = \log\frac{P(C_1|\mathbf{x})}{P(C_2|\mathbf{x})}$$

$$= \log\frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \log\frac{P(C_1)}{P(C_2)}$$

$$= \log\frac{(2\pi)^{-d/2}|\Sigma|^{-1/2}\exp\left[-(1/2)(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)\right]}{(2\pi)^{-d/2}|\Sigma|^{-1/2}\exp\left[-(1/2)(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)\right]} + \log\frac{P(C_1)}{P(C_2)}$$

$$= \mathbf{w}^T\mathbf{x} + w_0$$

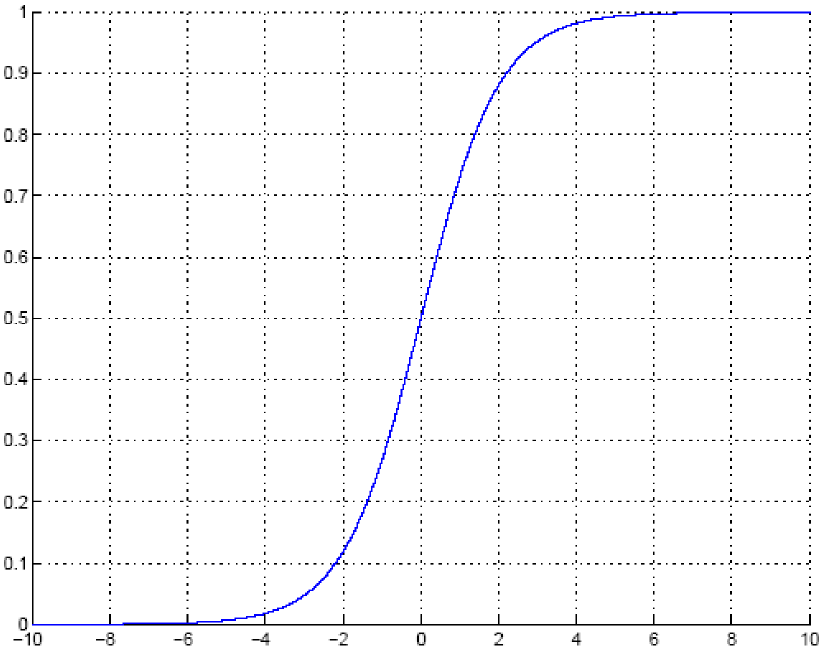
$$\text{where } \mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_2) \quad w_0 = -\frac{1}{2}(\mu_1+\mu_2)^T\Sigma^{-1}(\mu_1-\mu_2)$$

The inverse of logit ← Logistic function

$$\log\frac{P(C_1|\mathbf{x})}{1-P(C_1|\mathbf{x})} = \mathbf{w}^T\mathbf{x} + w_0$$

$$P(C_1|\mathbf{x}) = \text{sigmoid}(\mathbf{w}^T\mathbf{x} + w_0) = \frac{1}{1+\exp\left[-(\mathbf{w}^T\mathbf{x} + w_0)\right]}$$

Sigmoid (Logistic) Function



Calculate $g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ and choose C_1 if $g(\mathbf{x}) > 0$, or

Calculate $y = \text{sigmoid}\left(\mathbf{w}^T\mathbf{x} + w_0\right)$ and choose C_1 if $y > 0.5$

Sigmoid(0)=0.5, this function takes discriminant function to posterior probability

Logistic Discrimination (logistic regression)

Two classes: Assume log likelihood ratio (between 2 class) is linear

$$\log \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} = \mathbf{w}^T \mathbf{x} + w_0^o$$

$$\text{logit}(P(C_1|\mathbf{x})) = \log \frac{P(C_1|\mathbf{x})}{1-P(C_1|\mathbf{x})} = \log \frac{p(\mathbf{x}|C_1)}{p(\mathbf{x}|C_2)} + \log \frac{P(C_1)}{P(C_2)}$$

$$= \mathbf{w}^T \mathbf{x} + w_0$$

$$\text{where } w_0 = w_0^o + \log \frac{P(C_1)}{P(C_2)}$$

$$y = \hat{P}(C_1|\mathbf{x}) = \frac{1}{1+\exp\left[-\left(\mathbf{w}^T \mathbf{x} + w_0\right)\right]}$$

LR Training: Two Classes

Model label given x with probability y

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_t \qquad \boxed{r^t \mid \mathbf{x}^t \sim \text{Bernoulli}(y^t)}$$

Note the difference to likelihood method

$$y = P(C_1 \mid \mathbf{x}) = \frac{1}{1 + \exp\left[-\left(\mathbf{w}^T\mathbf{x} + w_0\right)\right]}$$

Maximize this function label/data condition likelihood based on data we have

$$l(\mathbf{w}, w_0 \mid \mathcal{X}) = \prod_t \left(y^t\right)^{\left(r^t\right)} \left(1 - y^t\right)^{\left(1 - r^t\right)}$$

$$E = -\log l$$

Minimize this

$$E(\mathbf{w}, w_0 \mid \mathcal{X}) = -\sum_t r^t \log y^t + \left(1 - r^t\right) \log\left(1 - y^t\right)$$

What is this? This is a function that we call ‘cross entropy’

Training: Gradient-Descent

$$E(\mathbf{w}, w_0 \mid \mathcal{X}) = -\sum_t r^t \log y^t + (1 - r^t) \log(1 - y^t)$$

If $y = \text{sigmoid}(a)$ $\quad \dfrac{dy}{da} = y(1 - y)$

$$\Delta w_j = -\eta \frac{\partial E}{\partial w_j} = \eta \sum_t \left(\frac{r^t}{y^t} - \frac{1 - r^t}{1 - y^t} \right) y^t (1 - y^t) x_j^t$$

$$\boxed{= \eta \sum_t (r^t - y^t) x_j^t, \; j = 1, \ldots, d}$$

$$\boxed{\Delta w_0 = -\eta \frac{\partial E}{\partial w_0} = \eta \sum_t (r^t - y^t)}$$

CHAPTER 14:

Kernel Machines

Kernel Machines

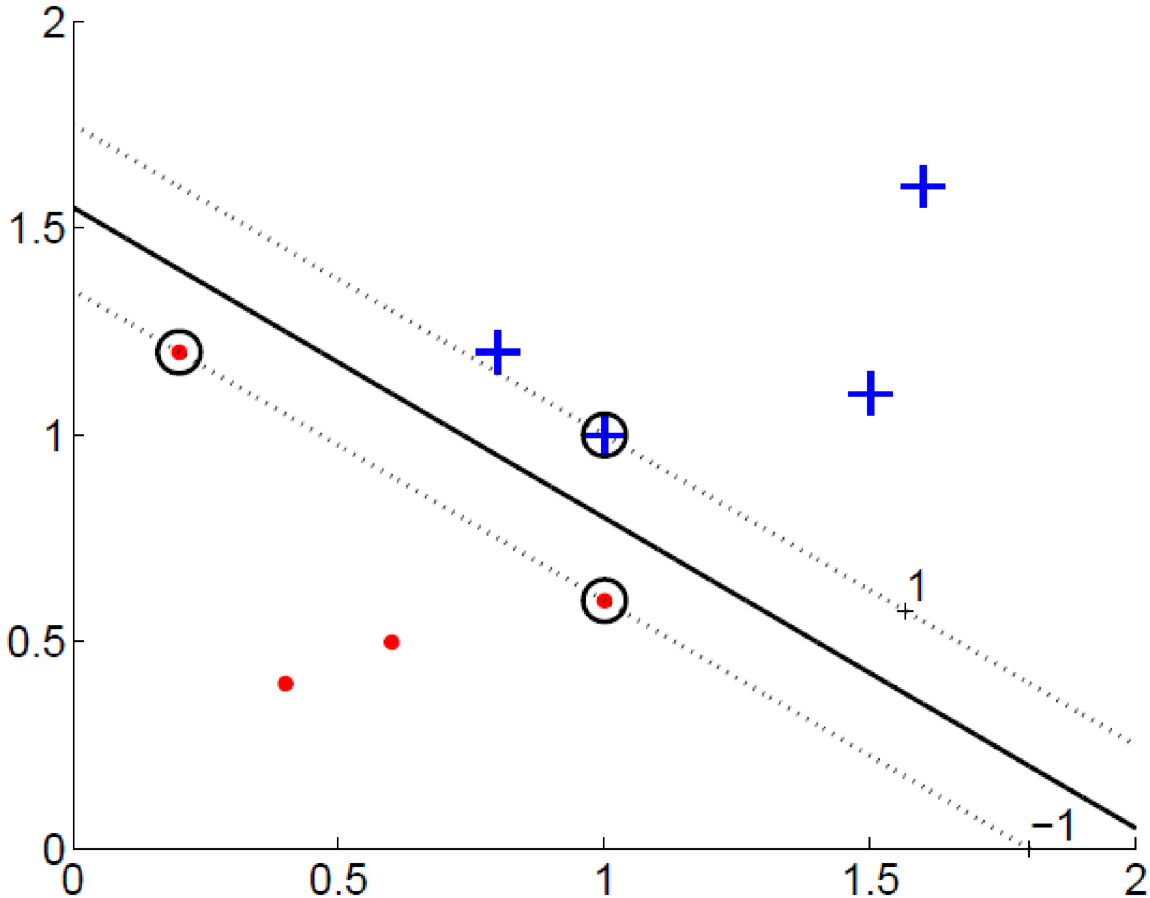
- Discriminant-based: No need to estimate densities first
- Define the discriminant in terms of support vectors
    - Support vectors: subset of training instances
- The use of kernel functions, application-specific measures of similarity
- Convex optimization problems with a unique solution

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

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

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

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

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

karush kuhn tucker condition
Maximum a^t
Such that
Lp gradient is 0
With respect to
ws

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

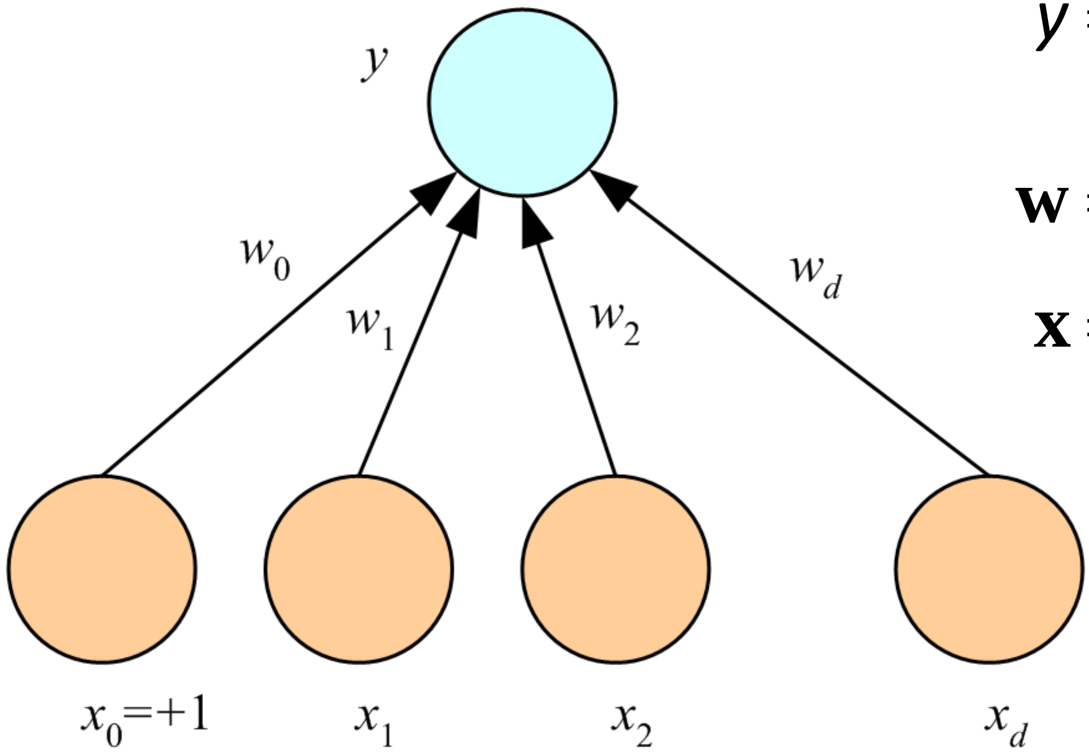
testing

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

CHAPTER 11:

Multilayer Perceptrons

Perceptron



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

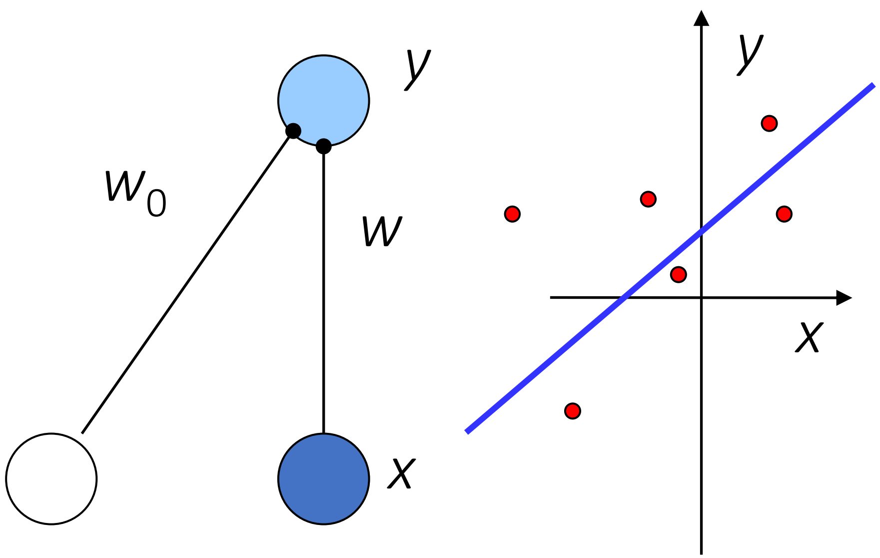
$$\mathbf{w} = \left[w_0, w_1, \ldots, w_d\right]^T$$

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

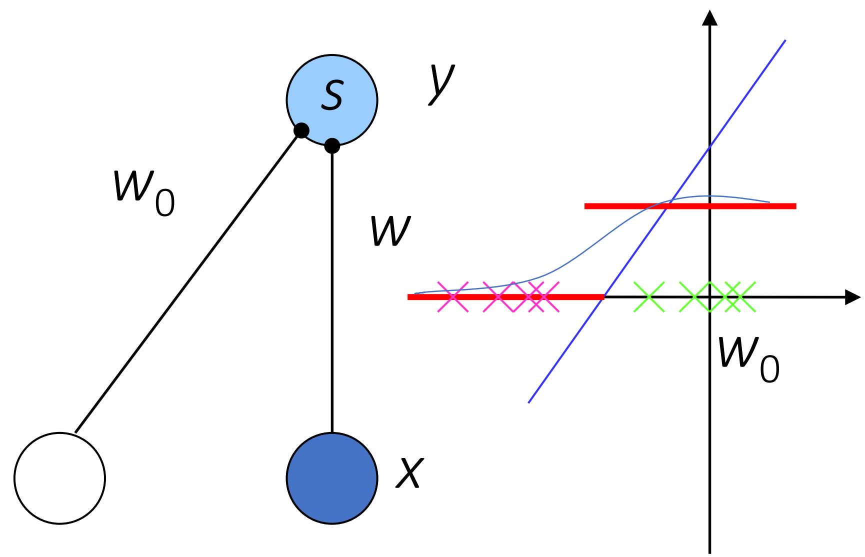
(Rosenblatt, 1962)

What a Perceptron Does

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

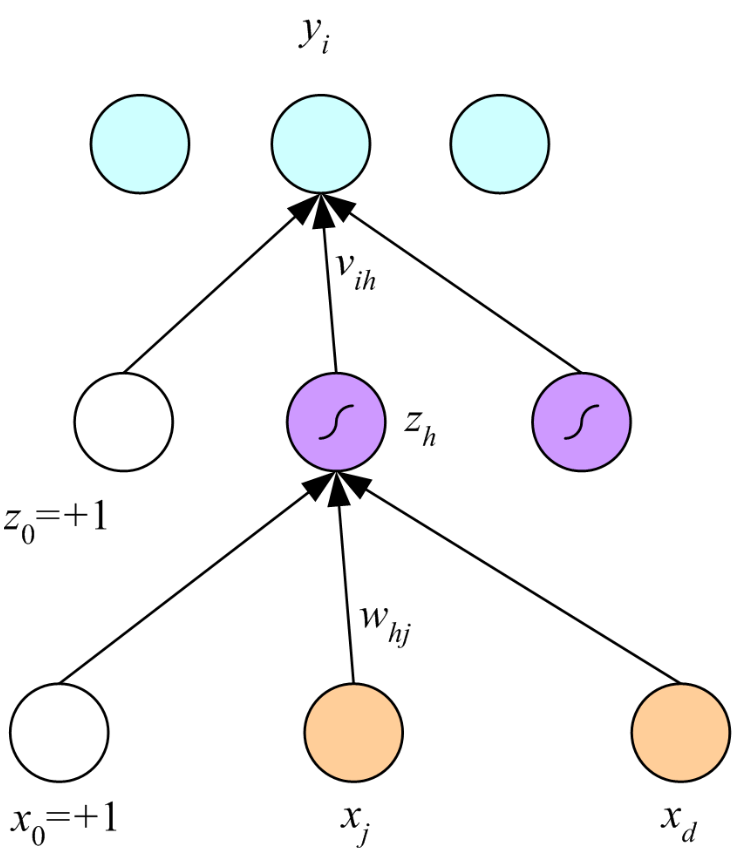


$x_0 = +1$



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

Multilayer Perceptrons



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

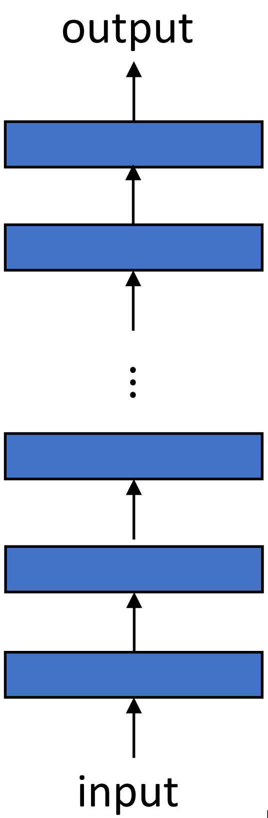
$$z_h = \text{sigmoid}\left(\mathbf{w}_h^T \mathbf{x}\right) = \frac{1}{1 + \exp\left[-\left(\sum_{j=1}^{d} w_{hj} x_j + w_{h0}\right)\right]}$$

(Rumelhart et al., 1986)

CHAPTER 12:

Deep Learning

Deep Neural Networks



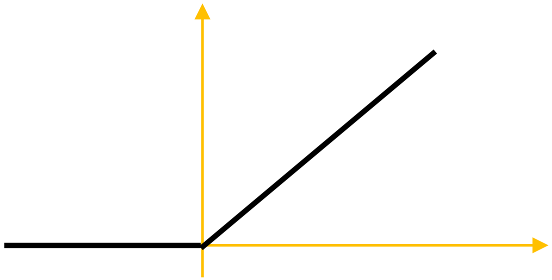
- Many hidden layers
    - Problematic in training: chain rule multiplication of derivatives (vanish of gradients or explosion)
- End-to-end training
- Learn increasingly abstract representations with minimal human contribution
    - Layers of abstraction in intuitive
    - Vision, speech, language, and so on
- Basis functions calculated from simpler basis functions

Representation learning is really the KEY behind Deep learning

Rectified Linear Unit (ReLU)

Left deriviation:
Relu'(a) = 1 if a>0, 0 otherwise

$$\text{ReLU}(a) = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases}$$



- Does not saturate for large *a*
- Leads to a sparse representation
  - Some of the nodes will results in 0
- No learning for *a*<0, be careful with initialization
  - Initialization should make sure all weights are positive
- Leaky ReLU: {a if a>0, αa otherwise, α is set 0.01)

Some derivative left

Improving Training Convergence

- Momentum. At each update, also add a fraction of the average of past gradients:

$$
\begin{aligned}
s_i^t &= \alpha s_i^{t-1} + (1-\alpha)\frac{\partial E^t}{\partial w_i} \\
\Delta w_i^t &= -\eta s_i^t
\end{aligned}
$$

Adaptive Learning Factor

- RMSprop. Make update inversely proportional to the sum of past gradients, so, update more when gradient is small and less where it is large.

$$\Delta w_i^t = -\frac{\eta}{\sqrt{r_i^t}} \frac{\partial E^t}{\partial w_i}$$

where r_i is the accumulated past gradient,

$$r_i^t = \rho r_i^{t-1} + (1-\rho) \left| \frac{\partial E^t}{\partial w_i} \right|^2$$

ADAM: Adaptive Learning Factor w/ Momentum

$$
\begin{aligned}
s_i^t &= \alpha s_i^{t-1} + (1-\alpha)\frac{\partial E^t}{\partial w_i} \\
r_i^t &= \rho r_i^{t-1} + (1-\rho)\left|\frac{\partial E^t}{\partial w_i}\right|^2
\end{aligned}
$$

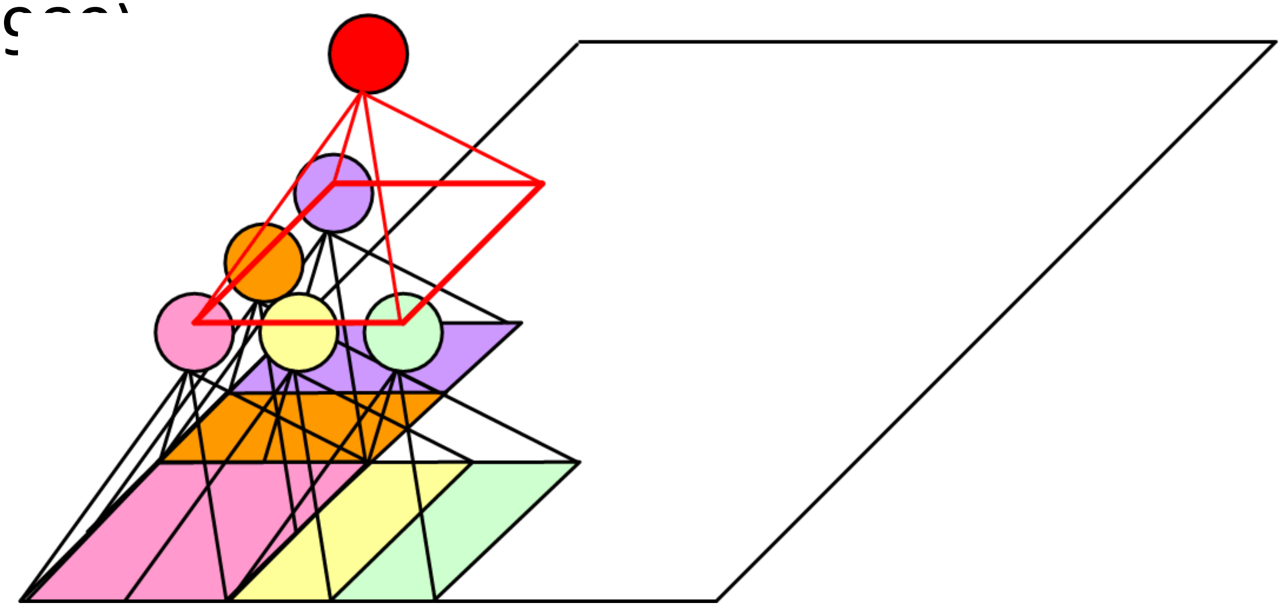
$$
\Delta w_i^t = -\eta \frac{\tilde{s}_i^t}{\sqrt{\tilde{r}_i^t}}
$$

Initially both *s* and *r* terms are 0, so we bias-correct them (both α and ρ are <1, so they get smaller as *t* gets large):

$$
\tilde{s}_i^t = \frac{s_i^t}{1-\alpha^t} \text{ and } \tilde{r}_i^t = \frac{r_i^t}{1-\rho^t}
$$

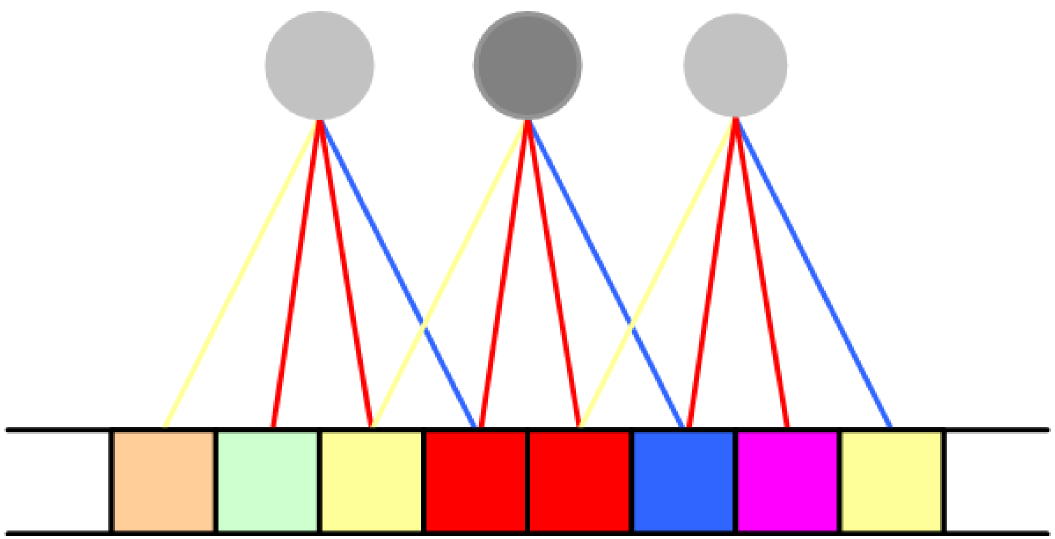
Regularization: Convolutions

- Each unit is connected to a small set of units in the preceding layer. In images, this corresponds to a local patch (LeCun et al 19[illegible])



Regularization: Weight Sharing

The same weights are used in different locations



1d example with 1x4 convolutions and weight sharing

Multiple Convolutional Layers