Intro to ML

November 1st, 2021

CHAPTER 7:

Clustering

Expectation-Maximization (EM)

- Log likelihood of a mixture model

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

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

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, $L_c(\Phi \mid X,Z)$, in terms of ***x*** and ***z***
- Incomplete likelihood, $L(\Phi \mid X)$, 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

$\mathbf{z}^t$ is indicator variable (z_1^t, z_2^t….z_k^t) z_i^t=1
when x^t = belong to cluster G_i
z multinomial distribution
Prior distribution z_i is π_i

$$P(\mathbf{z}^t) = \prod_{i=1}^{k} \pi_i^{z_i^t} \tag{7.7}$$

The likelihood of an observation $\mathbf{x}^t$ is equal to its probability specified by the component that generated it:

$$p(\mathbf{x}^t|\mathbf{z}^t) = \prod_{i=1}^{k} p_i(\mathbf{x}^t)^{z_i^t} \tag{7.8}$$

$p_i(\mathbf{x}^t)$ is shorthand for $p(\mathbf{x}^t|G_i)$. The joint density is

$$p(\mathbf{x}^t, \mathbf{z}^t) = P(\mathbf{z}^t)p(\mathbf{x}^t|\mathbf{z}^t)$$

and the complete data likelihood of the iid sample $\mathcal{X}$ is

$$\begin{aligned}
\mathcal{L}_c(\Phi|\mathcal{X}, \mathcal{Z}) &= \log \prod_t p(\mathbf{x}^t, \mathbf{z}^t|\Phi) \\
&= \sum_t \log p(\mathbf{x}^t, \mathbf{z}^t|\Phi) \\
&= \sum_t \log P(\mathbf{z}^t|\Phi) + \log p(\mathbf{x}^t|\mathbf{z}^t, \Phi) \\
&= \sum_t \sum_i z_i^t [\log \pi_i + \log p_i(\mathbf{x}^t|\Phi)]
\end{aligned}$$

$t \quad i$

E-step: We define

$$\begin{aligned}
\mathcal{Q}(\Phi|\Phi^l) &\equiv E\left[\log P(X,Z)|\mathcal{X},\Phi^l\right] \\
&= E\left[\mathcal{L}_c(\Phi|\mathcal{X},\mathcal{Z})|\mathcal{X},\Phi^l\right] \\
&= \sum_t \sum_i E[z_i^t|\mathcal{X},\Phi^l][\log \pi_i + \log p_i(\mathbf{x}^t|\Phi)]
\end{aligned}$$

$E[X] = \sum xp(x)$

where

$$\begin{aligned}
E[z_i^t|\mathcal{X},\Phi^l] &= E[z_i^t|\mathbf{x}^t,\Phi^l] \quad \mathbf{x}^t \text{ are iid} \\
&= P(z_i^t = 1|\mathbf{x}^t,\Phi^l) \quad z_i^t \text{ is a 0/1 random variable} \\
&= \frac{p(\mathbf{x}^t|z_i^t = 1,\Phi^l)P(z_i^t = 1|\Phi^l)}{p(\mathbf{x}^t|\Phi^l)} \quad \text{Bayes' rule} \\
&= \frac{p_i(\mathbf{x}^t|\Phi^l)\pi_i^l}{\sum_j p_j(\mathbf{x}^t|\Phi^l)\pi_j^l} \\
&= \frac{p(\mathbf{x}^t|\mathcal{G}_i,\Phi^l)P(\mathcal{G}_i)}{\sum_j p(\mathbf{x}^t|\mathcal{G}_j,\Phi^l)P(\mathcal{G}_j)} \\
&= P(\mathcal{G}_i|\mathbf{x}^t,\Phi^l) \equiv h_i^t
\end{aligned} \tag{7.9}$$

Under current model parameter

M-step: We maximize $\mathcal{Q}$ to get the next set of parameter values Φ^{l+1}:

$$\Phi^{l+1} = \arg\max_{\Phi} \mathcal{Q}(\Phi|\Phi^l)$$

which is

$$\begin{aligned}\mathcal{Q}(\Phi|\Phi^l) &= \sum_t \sum_i h_i^t [\log \pi_i + \log p_i(\boldsymbol{x}^t|\Phi)] \\ &= \sum_t \sum_i h_i^t \log \pi_i + \sum_t \sum_i h_i^t \log p_i(\boldsymbol{x}^t|\Phi)\end{aligned}$$

The second term is independent of π_i and using the constraint that $\sum_i \pi_i = 1$ as the Lagrangian, we solve for

$$\nabla_{\pi_i} \sum_t \sum_i h_i^t \log \pi_i - \lambda \left(\sum_i \pi_i - 1 \right) = 0$$

and get

$$\pi_i^{l+1} = \frac{\sum_t h_i^t}{N}$$

which is analogous to the calculation of priors in equation 7.2.

Similarly, the first term of equation 7.10 is independent of the components and can be dropped while estimating the parameters of the components. We solve for

$$\nabla_{\Phi} \sum_t \sum_i h_i^t \log p_i(\boldsymbol{x}^t|\Phi) = 0$$

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)

Complete steps for GMM

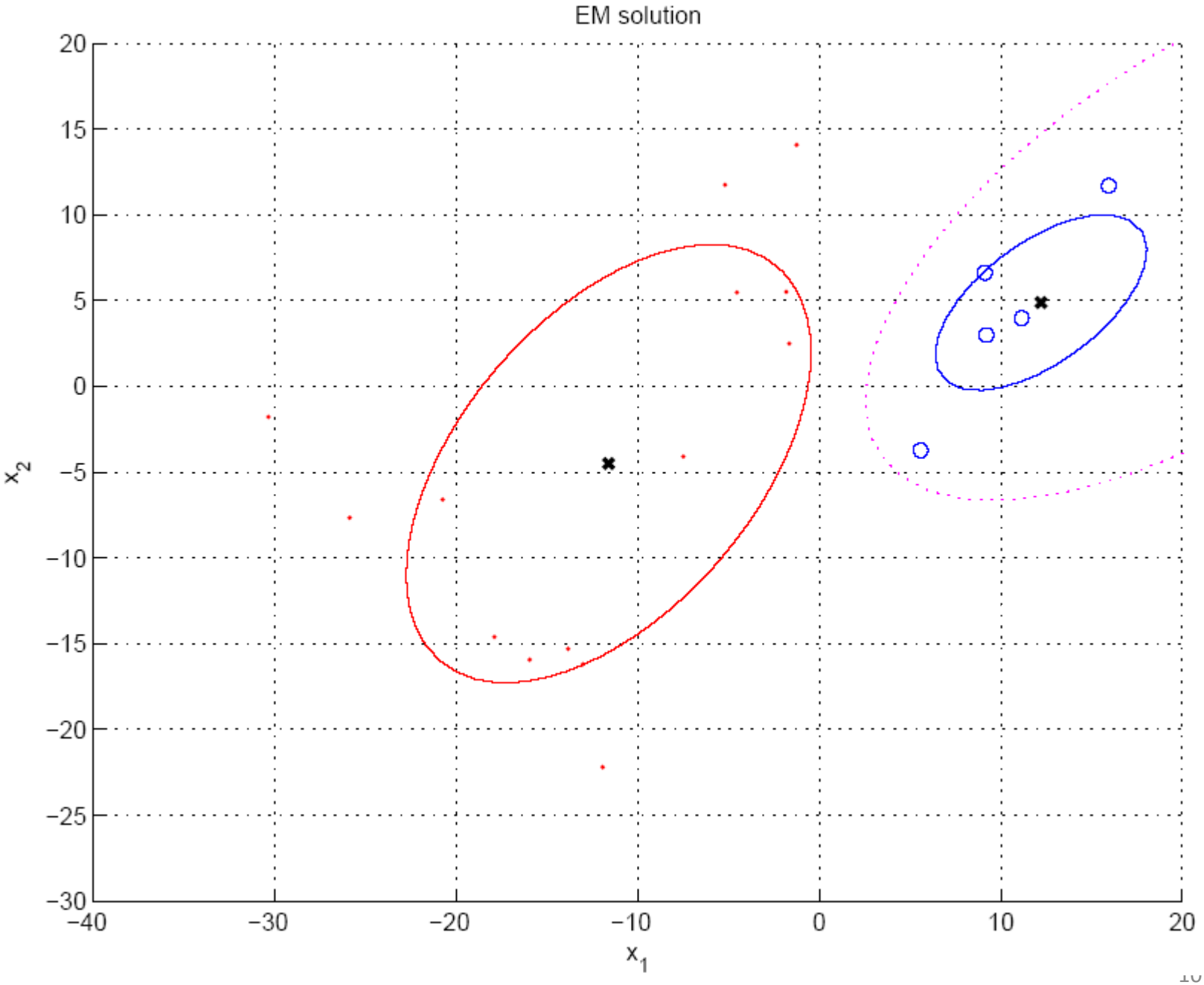
- If assume Gaussian component (each mixture is a Gaussian distribution)

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

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

Soft assignment of a sample to a class

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


EM solution
20
15
10
5
0
−5
−10
−15
−20
−25
−30
−40
−30
−20
−10
0
10
20
x2
x1

After Clustering

- Dimensionality reduction methods **find correlations between features and group(unsupervised, supervised) features**
- Clustering methods find **similarities between instances and group instances**
- Allows knowledge extraction through

  number of clusters -> diverse group

  prior probabilities -> natural clustering population

  cluster parameters, i.e., center, range of features.

  Example: CRM, customer segmentation

Clustering as Preprocessing

- Estimated group labels h_j (soft) or b_j (hard) may be seen as the dimensions of a new k dimensional space, where we can then learn our discriminant or regressor.
- Local representation (only one b_j is 1, all others are 0; only few h_j are nonzero) vs

  Distributed representation (After PCA; all z_j are nonzero)

Mixture of Mixtures

- In classification, the input comes from a mixture of classes (supervised).
- If each class is also a mixture, e.g., of Gaussians, (unsupervised), we have a mixture of mixtures:

$$p(\mathbf{x}|C_i) = \sum_{j=1}^{k_i} p(\mathbf{x}|G_{ij}) P(G_{ij})$$

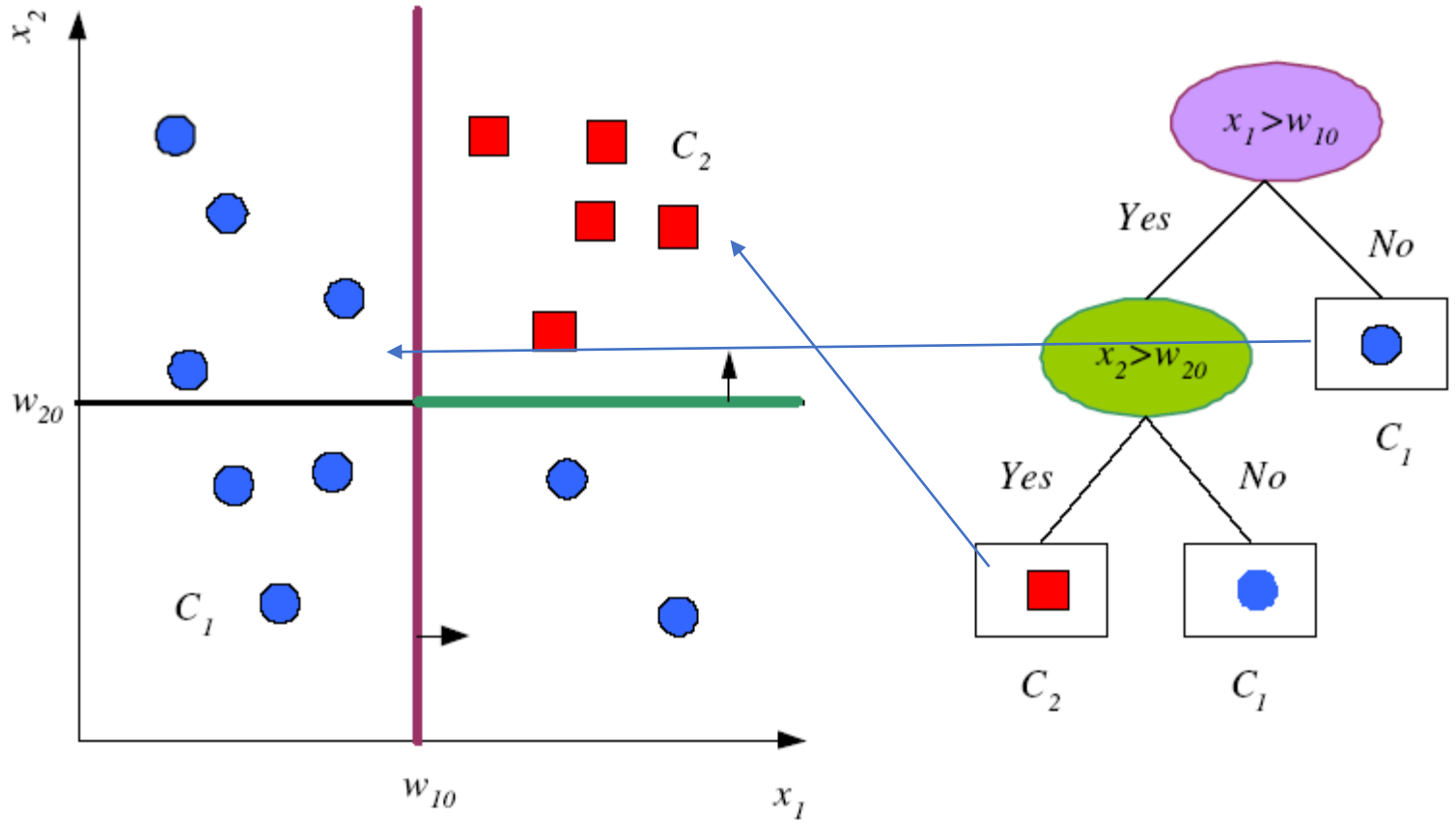
$$p(\mathbf{x}) = \sum_{i=1}^{K} p(\mathbf{x}|C_i) P(C_i)$$

- e.g., Using GMM as distribution for a class label

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

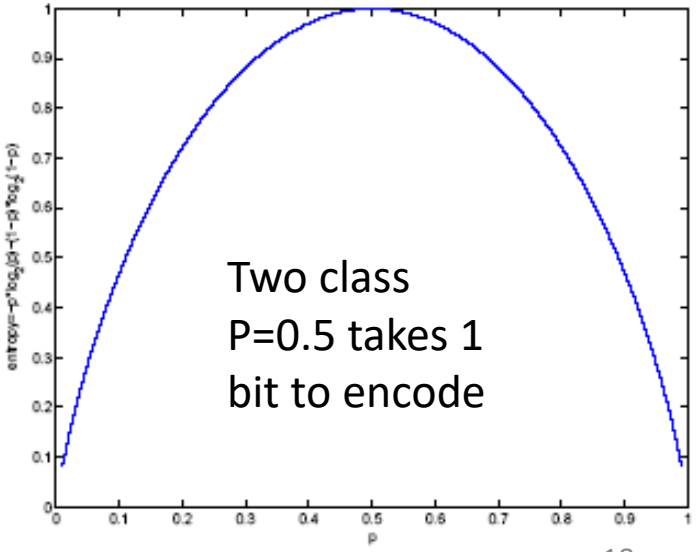
$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p^i_m = \frac{N^i_m}{N_m}$$

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:

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

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

Best Split

- If node *m* is pure, generate a leaf and stop, otherwise split and continue recursively
- Impurity after split at node m: N_{mj} of N_m take branch *j*. N^i_{mj} belong to C_i

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p^i_{mj} = \frac{N^i_{mj}}{N_{mj}} \qquad \mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p^i_{mj} \log_2 p^i_{mj}$$

Total impurity at node m

- Find the variable and split that min impurity (among all variables -- and split positions for numeric variables)

```
GenerateTree(X)
    If NodeEntropy(X)< θ_I /* eq. 9.3
        Create leaf labelled by majority class in X
        Return
    i ← SplitAttribute(X)
    For each branch of x_i
        Find X_i falling in branch
        GenerateTree(X_i)
SplitAttribute(X)
    MinEnt← MAX
    For all attributes i = 1,...,d
        If x_i is discrete with n values
            Split X into X_1,...,X_n by x_i
            e ← SplitEntropy(X_1,...,X_n) /* eq. 9.8 */
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split X into X_1, X_2 on x_i
                e←SplitEntropy(X_1, X_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

Some caveat

- Splitting favors attributes with many values (easy to find one that decreases the impurity the most)
  - This create a ‘complex tree’
  - Unreasonable : say take training sample index t as an attribute -> always result in 0 impurity but that’s not right
- Noise issue, may need a very large tree until it reaches purity
  - Define a threshold to say ‘pure enough’

Regression Trees

- Error at node m:

$$b_m(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_m : \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

$$E_m = \frac{1}{N_m} \sum_t \left(r^t - g_m\right)^2 b_m\left(\mathbf{x}^t\right) \qquad g_m = \frac{\sum_t b_m\left(\mathbf{x}^t\right) r^t}{\sum_t b_m\left(\mathbf{x}^t\right)}$$

Mean square error

- After splitting:

$$b_{mj}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_{mj} : \mathbf{x} \text{ reaches node } m \text{ and branch } j \\ 0 & \text{otherwise} \end{cases}$$

$$E'_m = \frac{1}{N_m} \sum_j \sum_t \left(r^t - g_{mj}\right)^2 b_{mj}\left(\mathbf{x}^t\right) \qquad g_{mj} = \frac{\sum_t b_{mj}\left(\mathbf{x}^t\right) r^t}{\sum_t b_{mj}\left(\mathbf{x}^t\right)}$$

Other notes

- Worst possible error

$E_m = \max_j \max_t |r^t - g_{mj}| b_{mj}(x^t)$

- This measure guarantees that the error for any instance is never larger than a given threshold

- Define acceptable error -> decides the complexity of the tree

- Instead of using average at the leaf node, we could use the following

$$g_m(x)=w_m\text{^}tx+w_{m0}$$

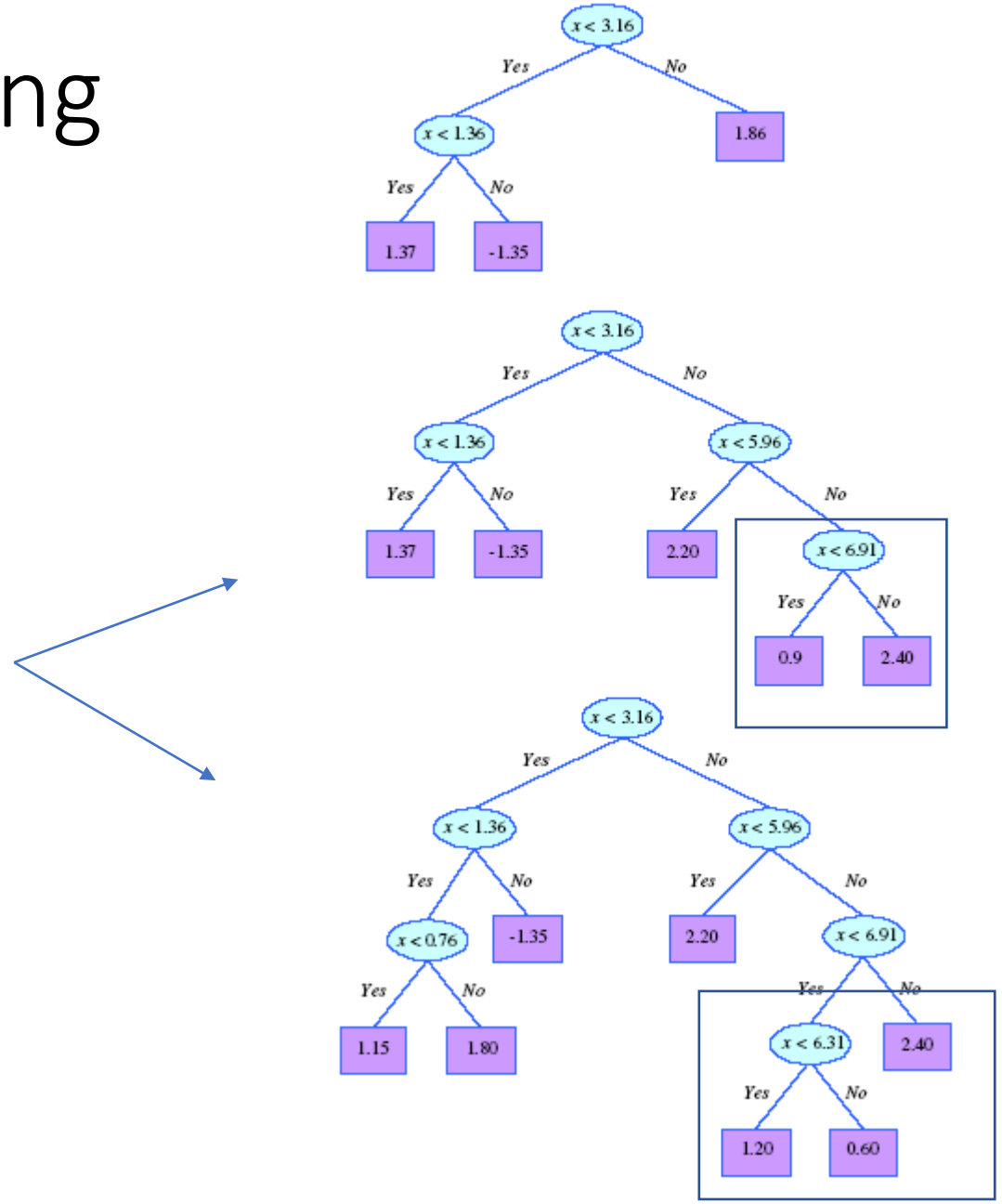
Pruning Trees

- Remove subtrees for better generalization (decrease variance)
  - Prepruning: Early stopping
    - Stop splitting when there are too few instances left (say 5% of training samples left)
  - Postpruning: Grow the whole tree then prune subtrees that overfit on the pruning set
- Prepruning is faster, postpruning is more accurate (requires a separate pruning set)

Post-pruning

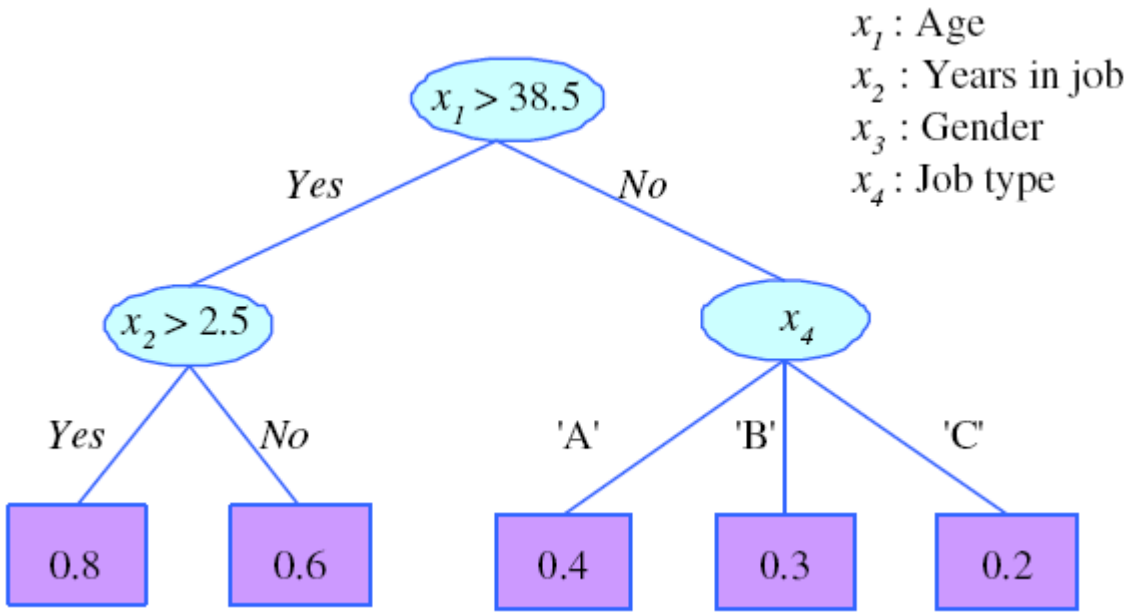
Take the separate pruning set, check whether the error in these two different trees are different

If not the complexity is not justified



Rule Extraction from Trees

C4.5Rules
(Quinlan, 1993)



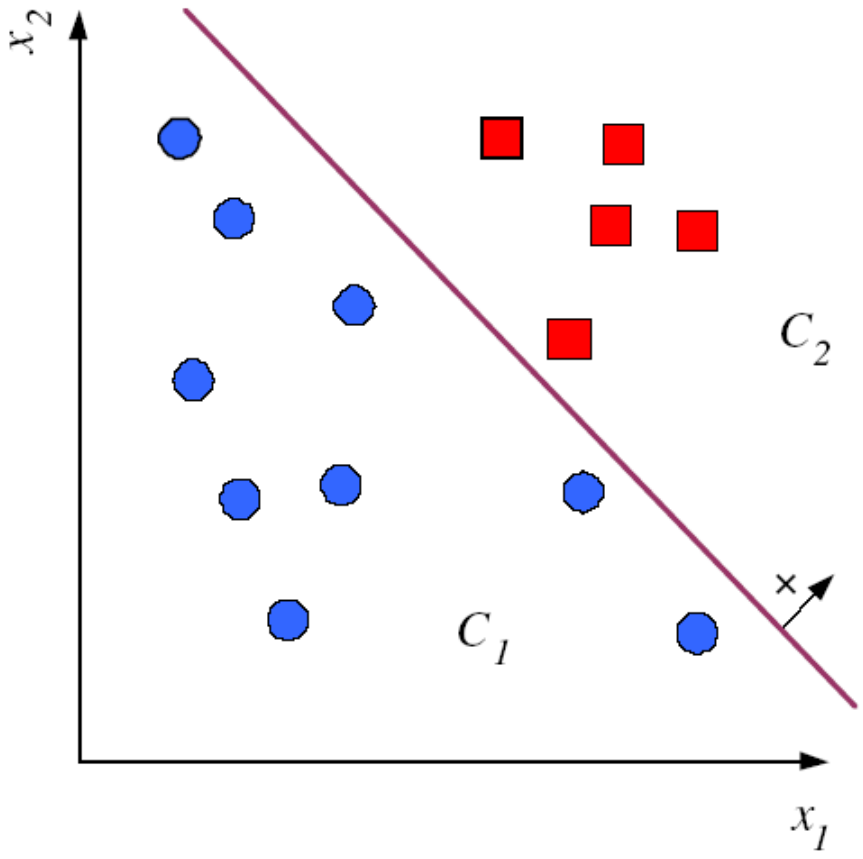
R1: IF (age>38.5) AND (years-in-job>2.5) THEN y =0.8
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN y =0.6
R3: IF (age≤38.5) AND (job-type='A') THEN y =0.4
R4: IF (age≤38.5) AND (job-type='B') THEN y =0.3
R5: IF (age≤38.5) AND (job-type='C') THEN y =0.2

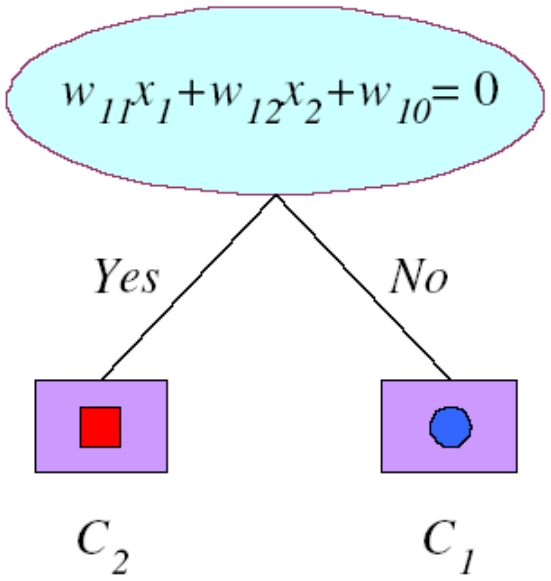
Multivariate Trees

Examples: if all inputs are numeric

The test function at node m for a binary split

fm(x) : w_m^t x + w_{m0} >0





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

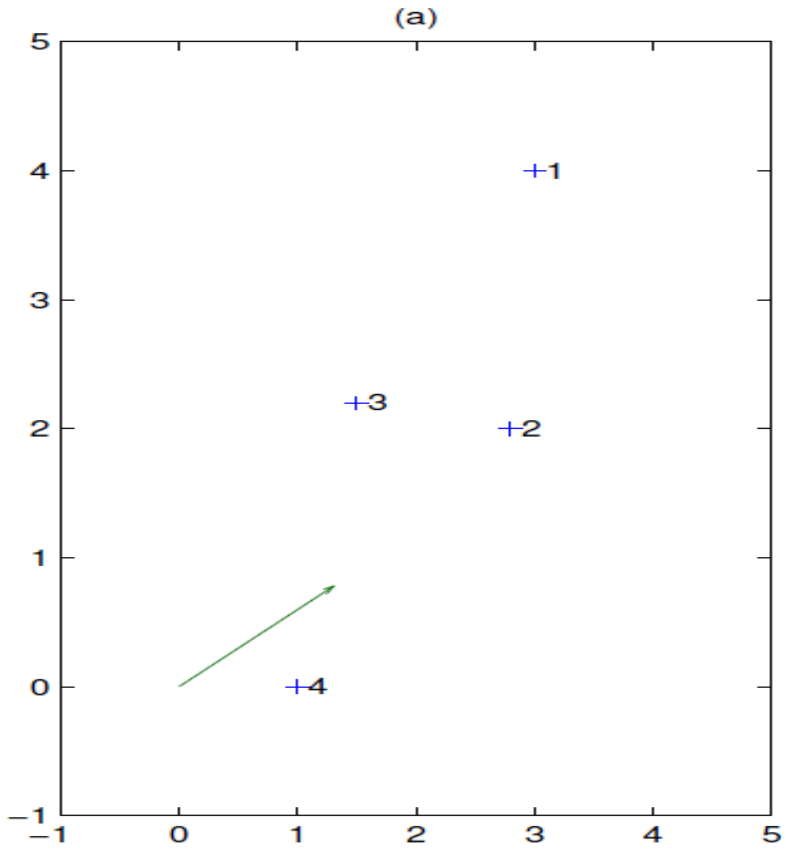
- Discriminant-based: Assume a **model** for $g_i(\boldsymbol{x}|\Phi_i)$; not density estimation
- Estimating the boundaries is enough; no need to accurately estimate the densities inside the boundaries
- Inductive bias come from your assumption of boundary not the density itself
- Knowing **how to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

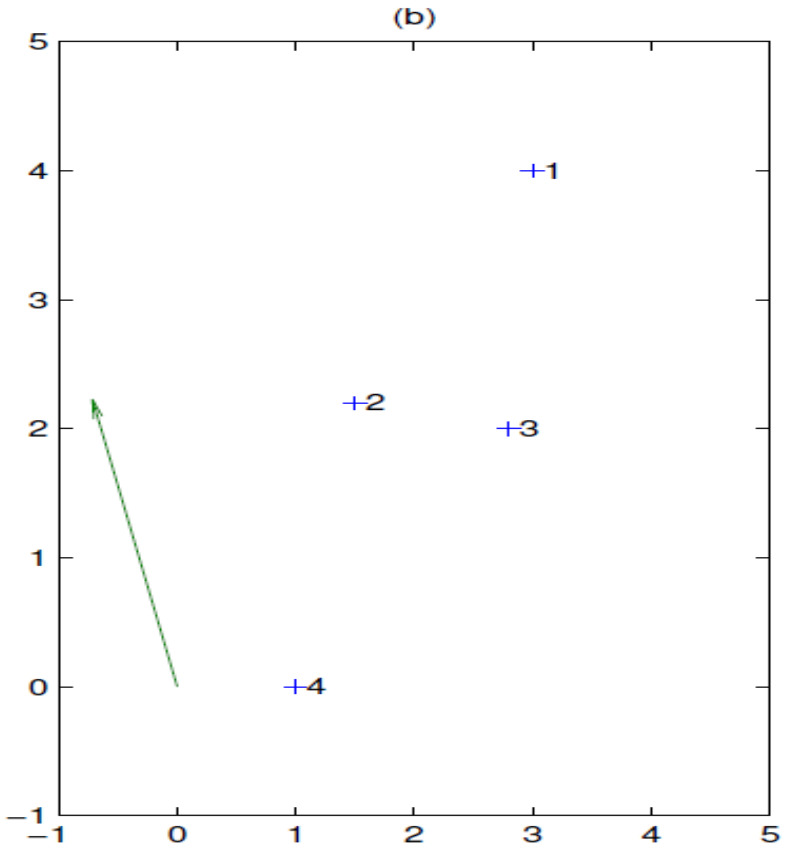
Linear Discriminant

- Linear discriminant function (assuming a linear separation):

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

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


(a)
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5
+1
+2
+3
+4


(b)
5
4
3
2
1
0
−1
−1
0
1
2
3
4
5
+1
+2
+3
+4