Midterm Review

Machine Learning 10-601B

Seyoung Kim

Basics on Probability

- Conditional probability
- Independence, conditional independence
- Bayes’ rule, prior, likelihood, posterior probability
- Chain rule

Probability Estimation

- Density estimation P(Y,X) P(Y|X) and its relation to classification
- For any given probability model, one can estimate the model parameters θ given data D with maximum likelihood or MAP estimations

- MLE: Choose θ that maximizes the probability of observed data:

$$\begin{aligned}\widehat{\theta} &= \arg\max_{\theta} \quad P(\mathcal{D} \mid \theta) \\ &= \arg\max_{\theta} \quad \ln P(\mathcal{D} \mid \theta)\end{aligned}$$

$$\boxed{\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = 0}$$

- MAP: use most likely parameter:

$$\widehat{\theta} = \arg\max_{\theta} P(\theta \mid \mathcal{D})$$

Naïve Bayes Classifier

- Training and testing based on Bayes rule

- Conditional independence in Naïve Bayes classifier

$$P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)$$

  i.e., that X_i and X_j are conditionally independent given Y, for all $i \neq j$

  - Why is it important?

- Naïve Bayes
  - Training using MLE, MAP estimates
  - How to handle discrete and continuous (Gaussian) input features

Naïve Bayes Algorithm – discrete X_i

- Train Naïve Bayes (given data for X and Y)

  for each[*] value y_k

  estimate $\pi_k \equiv P(Y = y_k)$

  for each[*] value x_{ij} of each attribute X_i

  estimate $\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k)$

- Classify (X^{new})

$$Y^{new} \leftarrow \arg\max_{y_k} \; P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

$$Y^{new} \leftarrow \arg\max_{y_k} \; \pi_k \prod_i \theta_{ijk}$$

[*] probabilities must sum to 1, so need estimate only n-1 of these...

Logistic Regression

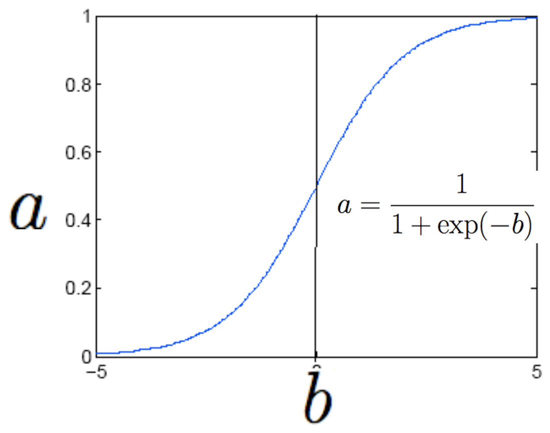
$$P(Y=1|X=<X_1,...X_n>) = \frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$P(Y=0|X=<X_1,...X_n>) = \frac{exp(w_0+\sum_i w_i X_i)}{1+exp(w_0+\sum_i w_i X_i)}$$

implies

$$\frac{P(Y=0|X)}{P(Y=1|X)} = exp(w_0+\sum_i w_i X_i)$$



implies

$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0+\sum_i w_i X_i$$

linear classification rule!

Training Logistic Regression: Maximizing Conditional Log Likelihood

$$P(Y=0|X,W) = \frac{1}{1+exp(w_0 + \sum_i w_i X_i)}$$

$$P(Y=1|X,W) = \frac{exp(w_0 + \sum_i w_i X_i)}{1+exp(w_0 + \sum_i w_i X_i)}$$

$$\begin{aligned} l(W) &\equiv \ln \prod_l P(Y^l|X^l,W) \\ &= \sum_l Y^l(w_0 + \sum_i^n w_i X_i^l) - \ln(1 + exp(w_0 + \sum_i^n w_i X_i^l)) \end{aligned}$$

Good news: *l(W)* is concave function of *W*
Bad news: no closed-form solution to maximize *l(W)*
Gradient descent!

Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

Generative classifiers (e.g., Naïve Bayes)

- Assume some functional form for P(X|Y), P(X) (i.e., P̀(X,Y))
- Estimate parameters of P(X|Y), P(X) directly from training data
- Use Bayes rule to calculate $P(Y|X= x_i)$

> - Find **θ =** argmax $_{\mathbf{w}}$ $\mathbf{\Pi_i}$ $Pr(y_i,x_i|$**θ)**
> - Different assumptions about *generative process* for the data: Pr(X,Y), priors on **θ,…**

Discriminative classifiers (e.g., Logistic regression)

- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

> - Find **θ =** argmax $_{\mathbf{w}}$ $\mathbf{\Pi_i}$ $Pr(y_i|x_i,$**θ)**
> - Different assumptions about conditional probability: Pr(Y|X), priors on θ, …

Generative vs. Discriminative Classifiers

Training classifiers involves estimating f: X → Y, or P(Y|X)

Generative classifiers (e.g., Naïve Bayes)

- Assume some functional form for P(X|Y), P(X) (i.e., P(X,Y))
- Estimate parameters of P(X|Y), P(X) directly from training data
- Use Bayes rule to calculate P(Y|X= x_i)

s for the

What are the advantages/ disadvantages of the two types of classifiers?

Discriminati

- Assume some functional form for P(Y|X)
- Estimate parameters of P(Y|X) directly from training data

- Find $\boldsymbol{\theta} = \text{argmax}_{\mathbf{w}} \boldsymbol{\Pi}_{\mathbf{i}} \Pr(y_i|x_i,\boldsymbol{\theta})$
- Different assumptions about conditional probability: Pr(Y|X), priors on θ, …

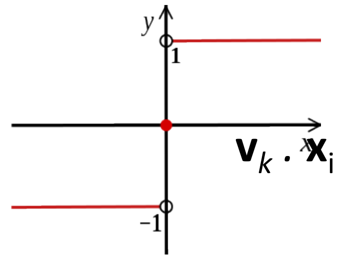
Perceptron

- Another linear but non-probabilistic classifier
- Margin-based learning

[Rosenblatt, 1957]

$$\begin{tikzcd} A \arrow[r, shift left, "instance\ \mathbf{x}_i"] \arrow[r, bend right=60, "y_i"'] & B \arrow[l, shift left, "\hat{y}_i"] \end{tikzcd}$$

Compute: $\hat{y}_i = \text{sign}(\mathbf{v}_k \,.\, \mathbf{x}_i)$



If mistake: $\mathbf{v}_{k+1} = \mathbf{v}_k + y_i \, \mathbf{x}_i$

- What happens if data are separable or non-separable?
- Voted perceptron

Linear Regression

- Learn P(Y|X) when Y is continuous

$$y = f(x) + \epsilon \qquad \text{where} \qquad \epsilon \sim N(0, \sigma)$$

$$f(x) = w_0 + \sum w_i x_i$$

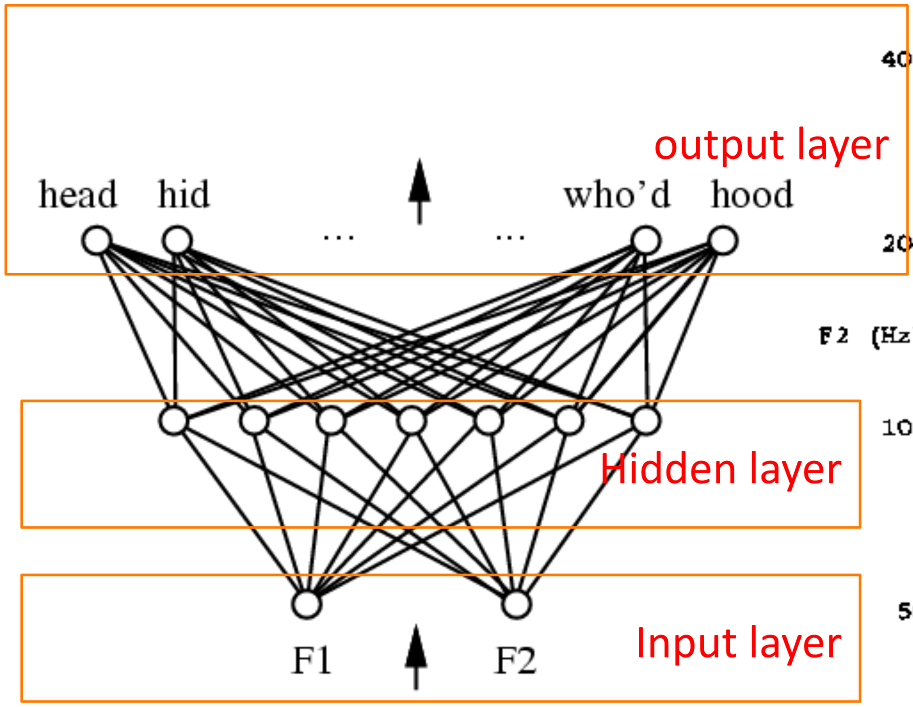
$$p(y|x) = N(w_0 + \sum_i w_i x_i, \sigma)$$

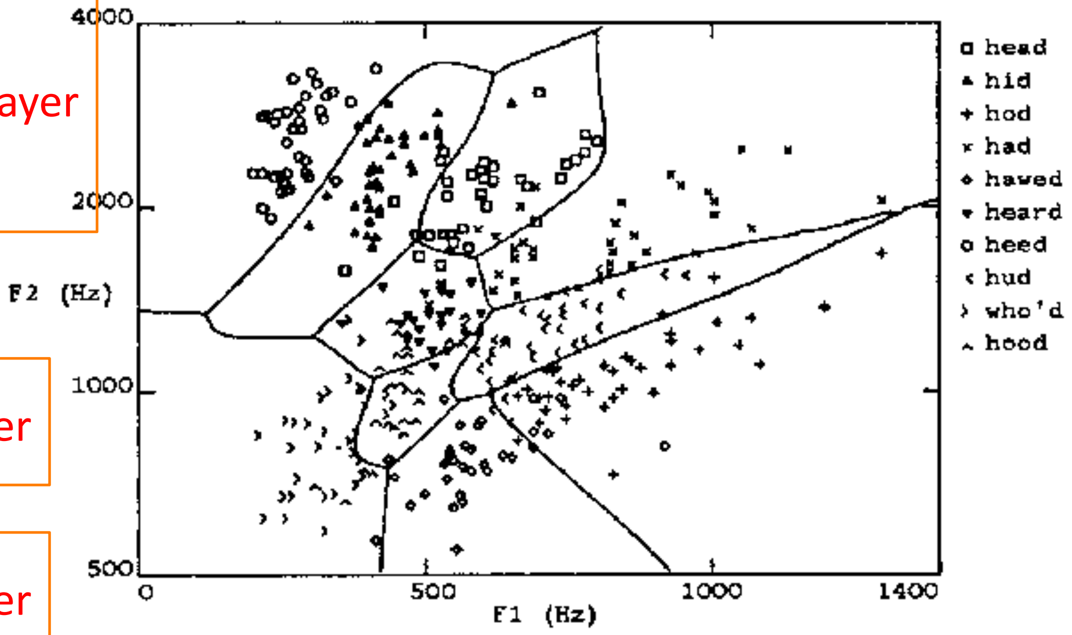
- MLE: How to find one, what it is, computational challenges

- MAP estimate: ridge regression, lasso, regularization

Multilayer Networks of Sigmoid Units

$$unit\ output = \frac{1}{1 + exp(w_0 + \sum_i w_i x_i)}$$



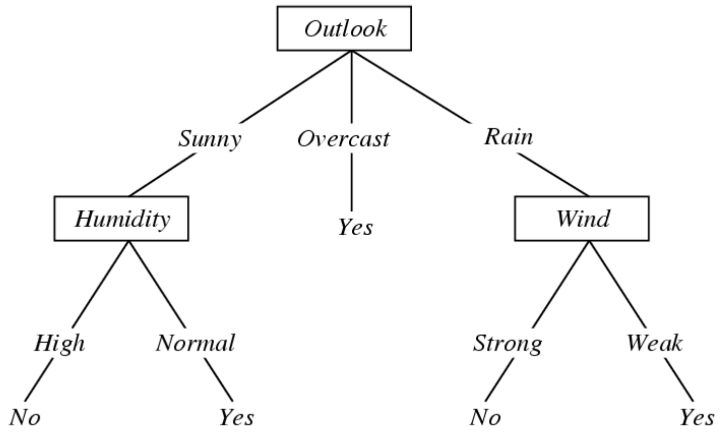


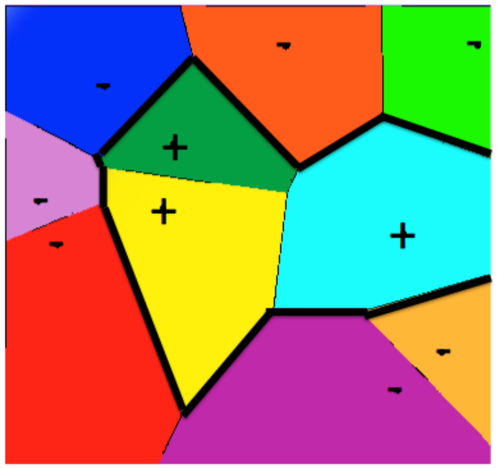
- non-linear decision boundary
- M(C)LE training
  - Non-convex optimization
  - back-propagation
  - Avoiding overfitting with early stopping

Other non-linear classifiers

- Decision tree learning
  - What decision trees are
  - How to learn them
    - information gain
  - How to prune them to avoid overfitting
- Nearest neighbor learning
  - No training!
  - At test time, for each test sample, find k-nearest neighbors and classify as frequent labels among neighbors

f: <Outlook, Humidity, Wind, Temp> → PlayTennis?



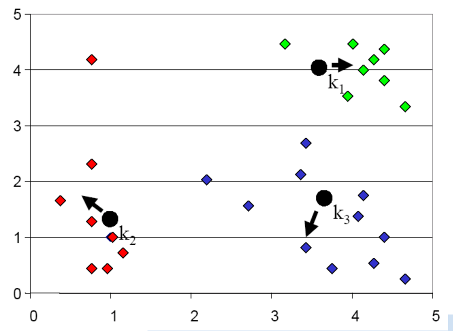


Unsupervised Learning

- Clustering
  - Hierarchical clustering
  - K-means clustering: non-probabilistic approach
  - Mixture model: probabilistic interpretation of K-means

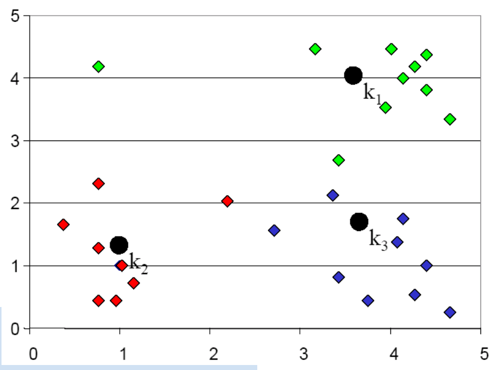
- Dimensionality reduction
  - Why useful?
  - Principal component analysis

K-Means Clustering Algorithm



Find the cluster means

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

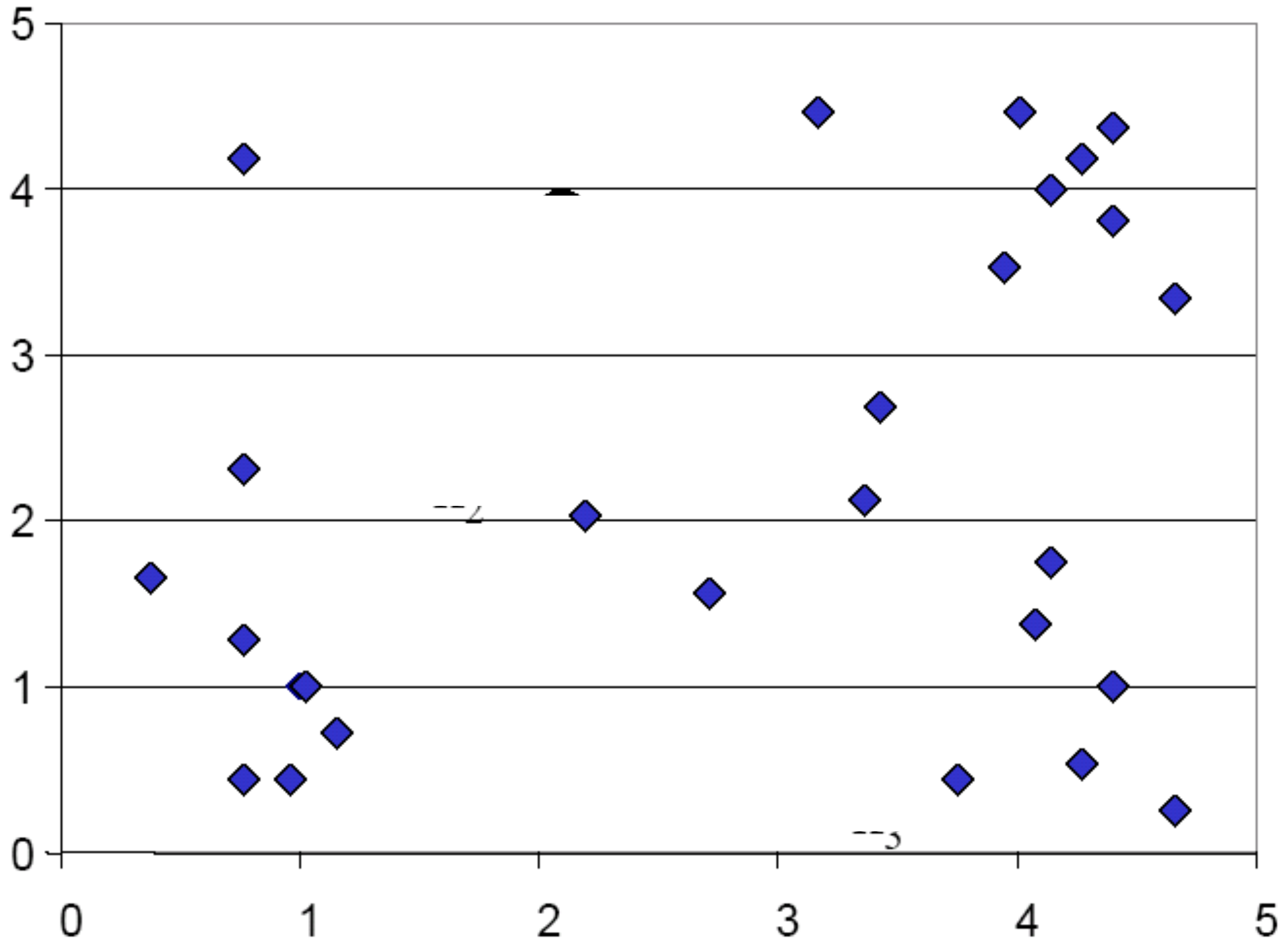


Re-assign samples x_i's to clusters

$$\arg\max_k \| x_i - \mu_k \|_2^2$$

Iterate until convergence

Mixture Model: Probability Model for Data P(X)?



Mixture Model

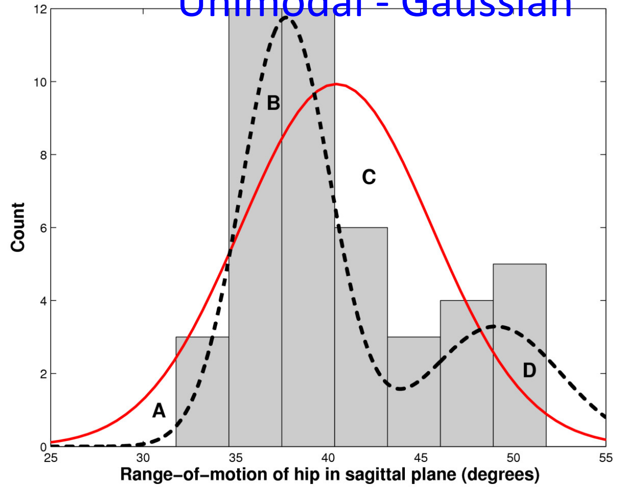
- A density model **p**(**x**) may be multi-modal

$$p(x_n) = \sum_k p(x_n, z_n = k) p(z_n = k)$$
$$= \sum_k N(x, | \mu_k, \Sigma_k) \pi_k$$

mixture proportion　　mixture component

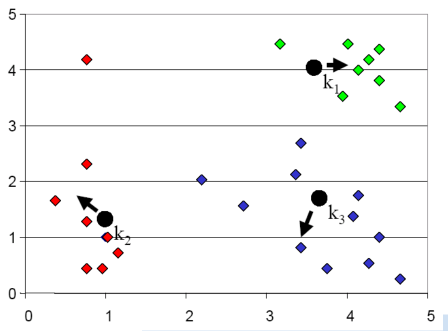
Multi-model: how do we model this?

Unimodal - Gaussian

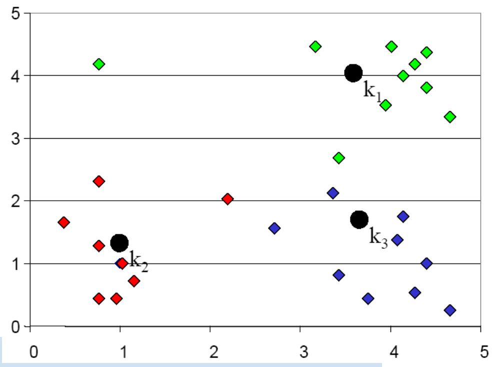


- EM algorithm
  - Maximize expected complete data log-likelihood (Expectation is taken with respect P(Y|X, θ)
  - E-step and M-step
  - Issues in convergence

EM Algorithm



Maximization (M)-step:
- Find mixture parameters



Expectation (E)-step:
-Re-assign samples x_i's to clusters
- Impute the unobserved values z_i

Iterate until convergence

Computing the Components

- Projection of vector **x** onto an axis (dimension) **u** is $\mathbf{u}^T\mathbf{x}$
- Assume **X** is a normalized *nxp* data matrix for *n* samples and *p* features. Direction of greatest variability is that in which the average square of the projection is greatest:

Maximize **(1/n)** $\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u}$

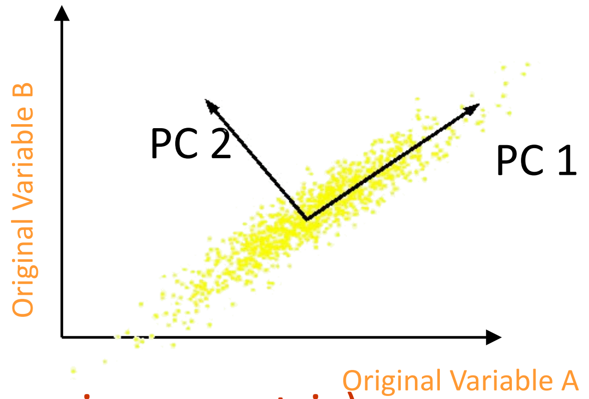
s.t $\mathbf{u}^T\mathbf{u} = 1$

Construct Langrangian (1/n) $\mathbf{u}^T\mathbf{X}^T\mathbf{X}\mathbf{u} - \lambda\mathbf{u}^T\mathbf{u}$

Vector of partial derivatives set to zero

$1/n\ \mathbf{X}^T\mathbf{X}\mathbf{u} - \lambda\mathbf{u} = 0$

or equivalently $\mathbf{S}\mathbf{u} - \lambda\mathbf{u} = 0$ (S =$\mathbf{1/n}\ \mathbf{X}^T\mathbf{X}$: covariance matrix)



As $\mathbf{u} \neq \mathbf{0}$ then **u** must be an eigenvector of S with eigenvalue λ

- λ is the principal eigenvalue of the **covariance matrix S**
- The eigenvalue denotes the amount of variability captured along that dimension

- How many principal components?

Semi-supervised Learning

- Improving naïve Bayes classifier with unlabeled data

- EM algorithm
  - Why EM?
  - How to derive it?

Bayesian Networks <u>Definition</u>

A Bayes network represents the joint probability distribution over a collection of random variables

A Bayes network is a directed acyclic graph and a set of conditional probability distributions (CPD's)

- Each node denotes a random variable
- Edges denote dependencies
- For each node X_i its CPD defines $P(X_i \mid Pa(X_i))$
- The joint distribution over all variables is defined to be

$$P(X_1 \ldots X_n) = \prod_i P(X_i | Pa(X_i))$$

Pa(X) = immediate parents of X in the graph

Constructing a Bayesian network

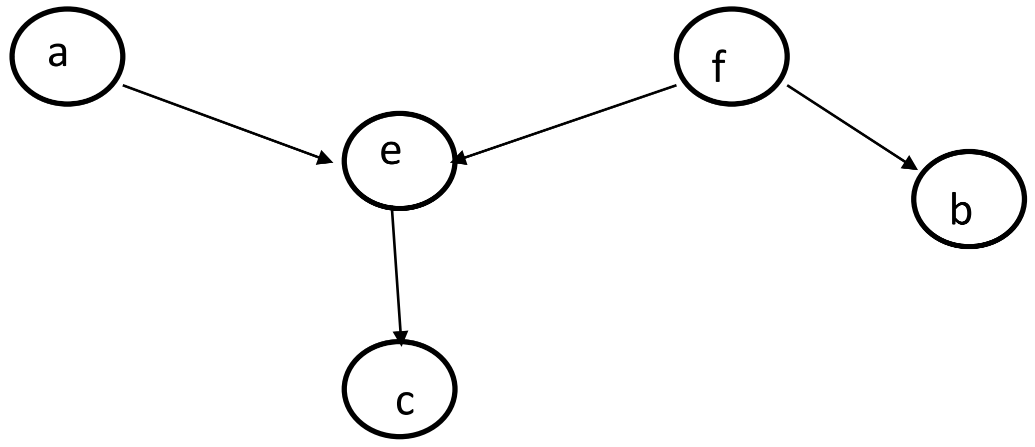
- How do we go about constructing a network for a specific problem?
- Step 1: Identify the random variables
- Step 2: Determine the conditional dependencies
- Step 3: Populate the CPTs

Can reduce the number of parameters! Why?

Can be learned from observation data!

Conditional independencies in Bayesian networks
- Markov blanket: All parent, children and co-parents of children
- D-separation: v-structure, non v-structure

Markov Blanket, D-separation

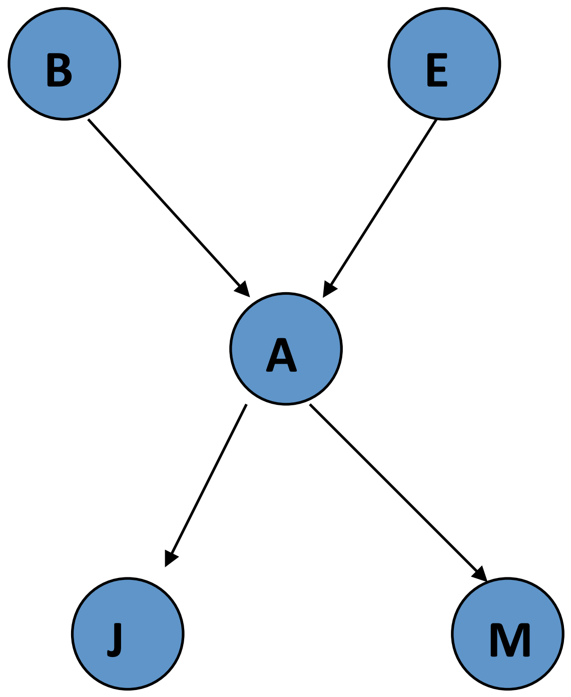


$$a \perp b \mid c\,?$$

$$a \perp b \mid f\,?$$

Inference/Learning in Bayesian Networks

- Once the network is constructed, we can use algorithms for inferring the values of unobserved variables.
- For example, in our previous network the only observed variables are the phone call and the radio announcement. However, what we are really interested in is whether there was a burglary or not.
- How can we determine that?
  1. Enumeration
  2. Variable elimination
  3. Stochastic inference
- Learning parameters given network structure: MLE, MAP



B – Did a burglary occur?

E – Did an earthquake occur?

A – Did the alarm go off?

M – Mary calls

J – John calls

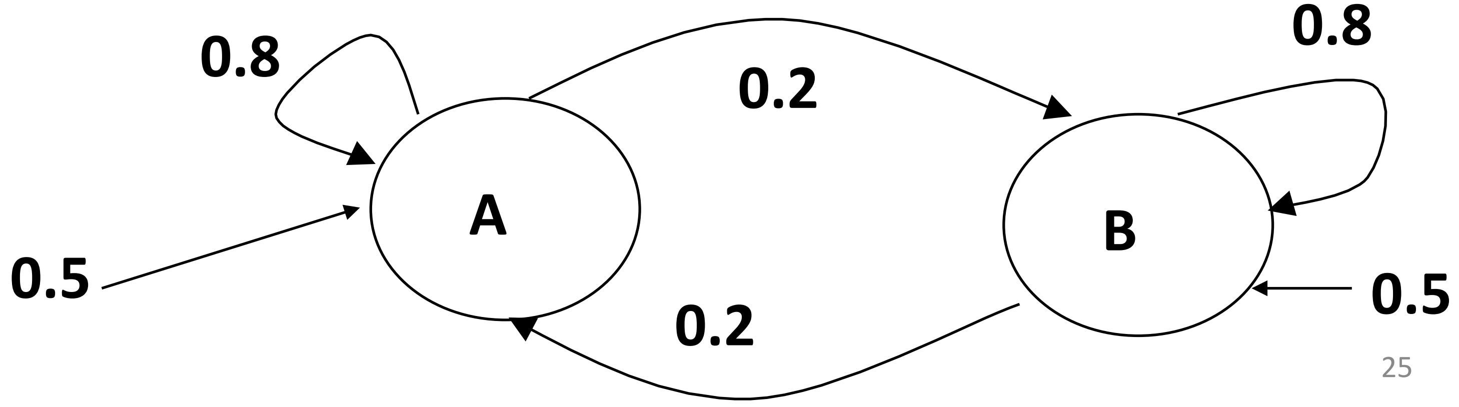
A Hidden Markov model

- A set of states $\{s_1 \ldots s_n\}$

  - In each time point we are in exactly one of these states denoted by q_t
- A set of possible outputs Σ

  - At time *t* we emit a symbol $\sigma \in \Sigma$

- Π_i, the probability that we *start* at state s_i
- A transition probability model, $P(q_t = s_i \mid q_{t-1} = s_j)$
- An emission probability model, $p(o_t = \sigma \mid s_i)$



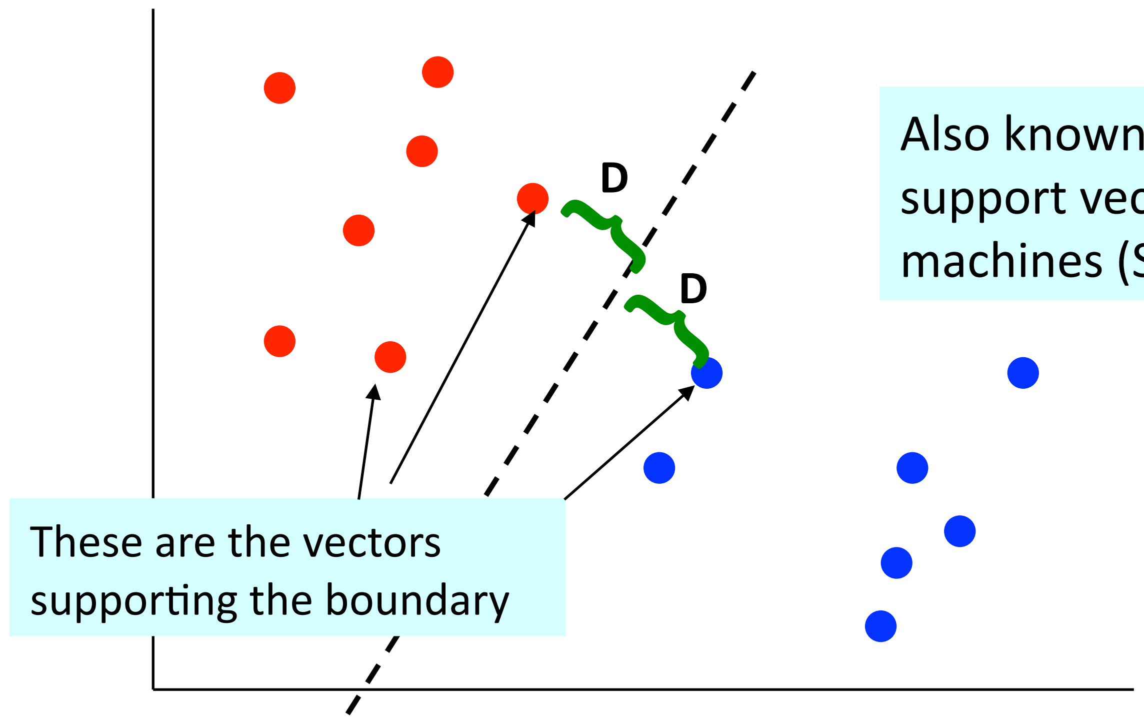
Inference/Learning in HMMs

- Computing P(Q) and $P(q_t = s_i)$
  - If we cannot look at observations
- Computing P(Q | O) and $P(q_t = s_i | O)$
  - Forward-backward algorithm
- Computing $\text{argmax}_Q P(Q | O)$
  - When we care about the entire path
  - Viterbi algorithm

- Learning with EM algorithm
  - Why?
  - Inference as a subroutine in E step

Support Vector Machine as Max Margin Classifiers

- Instead of fitting all points, focus on boundary points
- Learn a boundary that leads to the largest margin from points on both sides



Also known as linear support vector machines (SVMs)

These are the vectors supporting the boundary

Optimization Problem for Support Vector Machine

Two optimization problems: For the separable and non separable cases

$$\min_w \frac{w^T w}{2}$$

For all x in class + 1

$w^T x + b \geq 1$

For all x in class - 1

$w^T x + b \leq -1$

$$\min_w \frac{w^T w}{2} + \sum_{i=1}^{n} C\varepsilon_i$$

For all x_i in class + 1

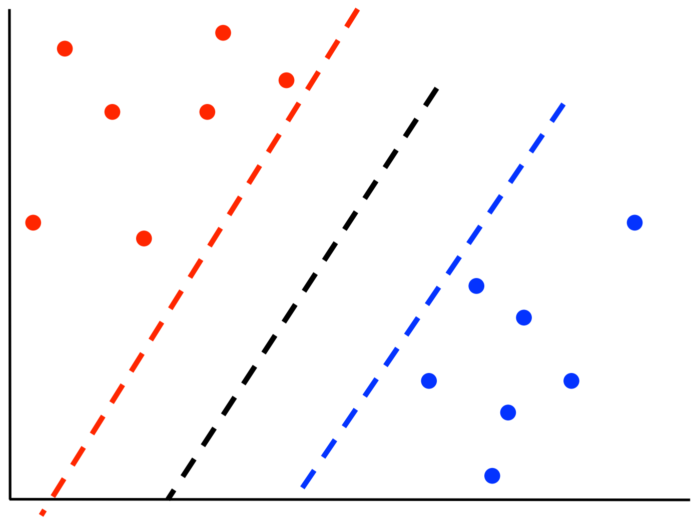
$w^T x + b \geq 1 - \varepsilon_i$

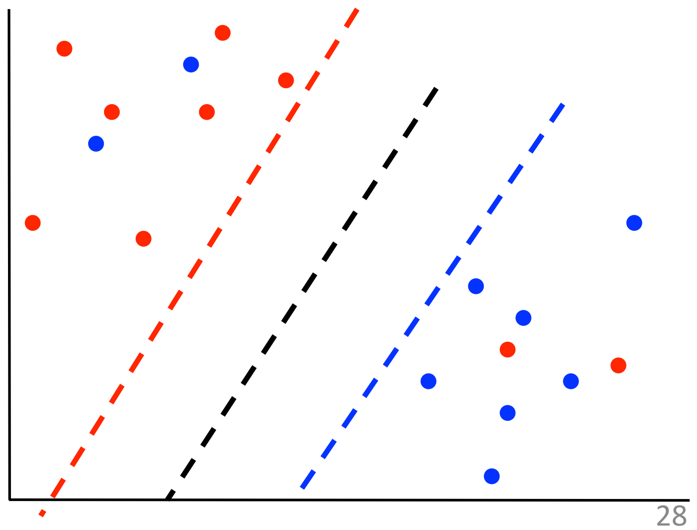
For all x_i in class - 1

$w^T x + b \leq -1 + \varepsilon_i$

For all i

$\varepsilon_l \geq 0$





Dual SVM for linearly separable case

Our dual target function:

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2}\sum_{\mathrm{i,j}} \alpha_i \alpha_j \mathrm{y}_\mathrm{i} y_j \mathbf{x_i x_j}$$

$$\sum_i \alpha_i \mathrm{y}_\mathrm{i} = 0$$

$$\alpha_i \geq 0 \qquad \forall i$$

Dot product for all training samples

Dot product with training samples

To evaluate a new sample x_j we need to compute:

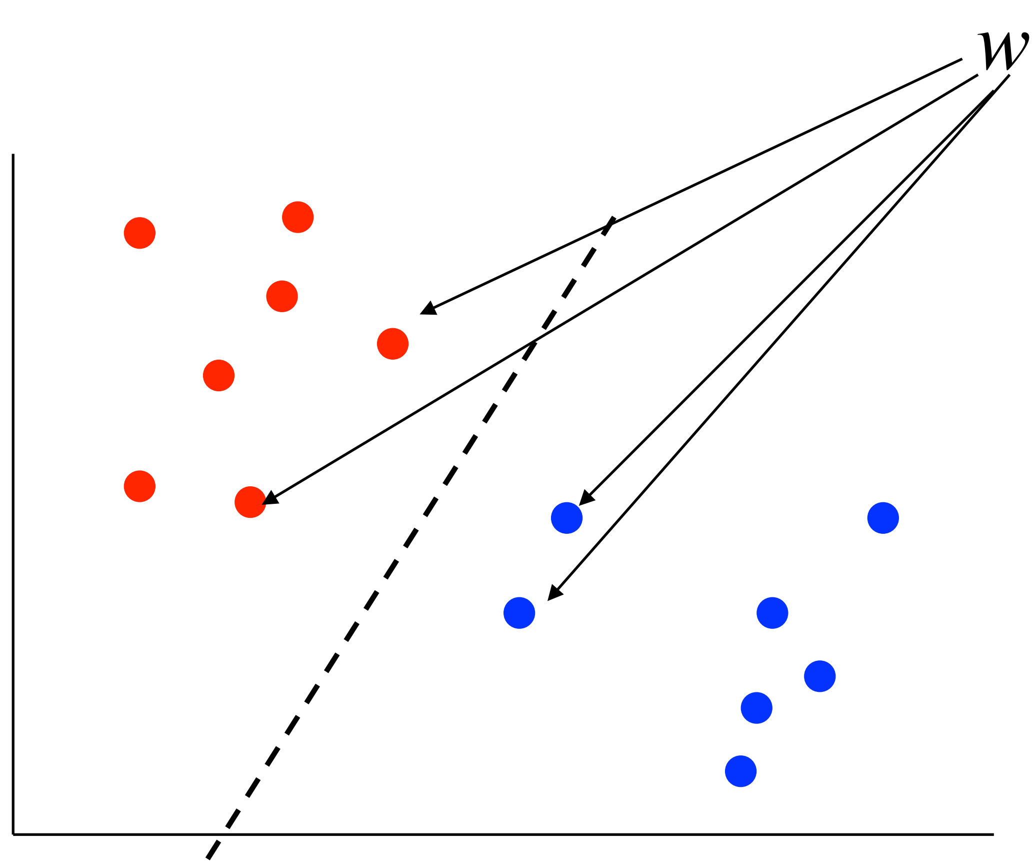
$$\mathrm{w}^\mathrm{T} x_j + b = \sum_\mathrm{i} \alpha_i \mathrm{y}_\mathrm{i} \mathbf{x_i x_j} + b$$

Is this too much computational work (for example when using transformation of the data)?

How to apply kernel trick?

Dual SVM - interpretation

$$w = \sum_i \alpha_i x_i y_i$$



For α's that are not 0 – support vectors!

Regularized Regression

- Recall linear regression: $$\mathbf{y} = \mathbf{X}^T \beta + \varepsilon$$

$$\beta^* = \arg\max_\beta (\mathbf{y} - \mathbf{X}^T \beta)^T (\mathbf{y} - \mathbf{X}^T \beta)$$

$$= \arg\max_\beta \| \mathbf{y} - \mathbf{X}^T \beta \|^2$$

- Regularized LR:
  - L2-regularized LR:

$$\beta^* = \arg\max_\beta \| \mathbf{y} - \mathbf{X}^T \beta \|^2 + \lambda \| \beta \|$$

where $$\| \beta \| = \sum_i \beta_i^2$$

  - L1-regularized LR:

$$\beta^* = \arg\max_\beta \| \mathbf{y} - \mathbf{X}^T \beta \|^2 + \lambda | \beta |$$

where $$| \beta | = \sum_i | \beta_i |$$

Performs a model selection directly