L11: Pattern recognition principles

Bayesian decision theory

Statistical classifiers

Dimensionality reduction

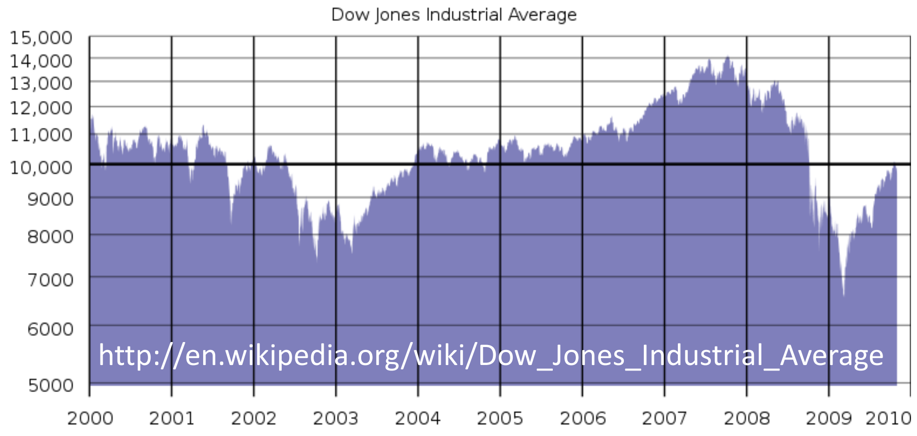
Clustering

This lecture is partly based on [Huang, Acero and Hon, 2001, ch. 4]

Bayesian decision theory

Bayes decision rule

- Consider the problem of making predictions for the stock market
  - The goal is to predict whether the Dow Jones Industrial Average index will go UP, DOWN, or remain UNCHANGED
- Assume the only available information is the prior probability $P(\omega_i)$ for each of the three outcomes, which we denote by $\{\omega_1, \omega_2, \omega_3\}$
  - If we are to make predictions based only on the prior, the most sensitive decision would be to choose the class with largest prior
  - However, this decision is unreasonable since it always makes the same prediction but we know that the DJIA does fluctuate up and down



- To improve our predictions, we may make additional observations x, i.e., interest rates, unemployment rates, etc.
  - These observations can be characterized by their likelihoods $p(x|\omega_i)$, which tells how likely we are to observe x for each possible outcome
  - With this information, and using Bayes' rule we obtain

$$P(\omega_i|x) = \frac{p(x|\omega_i)P(\omega_i)}{p(x)} = \frac{p(x|\omega_i)P(\omega_i)}{\sum_{k=1}^{3} p(x|\omega_k)P(\omega_k)}$$

- An intuitive decision rule would be to choose class ω_k with greatest posterior

$$k = \arg\max_i P(\omega_i|x\ ) = \arg\max_i p(x|\omega_i\ )P(\omega_i\ )$$

- As we will see next, this rule yields the minimum decision error (on average)

Minimum-error-rate decision rules

- Let $\Omega = \{\omega_1 \dots \omega_s\}$ be a set of s possible categories to be predicted, and let $\Delta = \{\delta_1 \dots \delta_t\}$ be the finite set of t possible decisions
- Let $l(\delta_i|\omega_j)$ be the loss incurred for making decision δ_i when the true class is ω_j (in our market example, this loss would be financial)
- The expected loss associated with making δ_i given observation x is

$$R(\delta_i|x) = \sum_{j=1}^{s} l(\delta_i|\omega_j)P(\omega_j|x)$$

  - This expression is known as a conditional risk
- We then seek a decision rule $\delta(x)$ (a mapping from observations x into $\Delta = \{\delta_1 \dots \delta_t\}$) that minimizes the overall risk

$$R = \int_{-\infty}^{\infty} R(\delta(x)|x)p(x)dx$$

- which can be minimized by minimizing its integrand $\forall x$, that is, by always choosing the class with lowest conditional risk

$$k = \arg\min_i R(\delta_i|x) = \arg\min_i \sum_{j=1}^{s} l(\delta_i|\omega_j)P(\omega_j|x)$$

- Assume that all errors are equally costly, which we can model with a symmetric zero-one loss function

$$l(\delta_i|\omega_j) = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases}$$

  - Can you think of situations where this loss function may not be advisable?

- Plugging this expression into that of the conditional risk yields

$$R(\delta_i|x) = \sum_{j=1}^{s} l(\delta_i|\omega_j) P(\omega_j|x) =$$

$$= \sum_{j \neq \mathrm{i}} P(\omega_j|x) = 1 - P(\omega_i|x)$$

- In other words, in order to minimize classification rate, we should always choose the class with largest posterior

Statistical classifiers

What happens when the likelihood functions are unknown?

- The Bayes/MAP decision rule assumes that $p(x|\omega_i)$ are known, which is unlikely in most practical applications
- In this case, we have several options
  - We can attempt to estimate $p(x|\omega_i)$ from data, by means of density estimation techniques
    - This gives rise to methods such as Naïve Bayes and nearest-neighbors
  - We can assume $p(x|\omega_i)$ follows a particular distribution (i.e. Normal) and estimate its parameters (see previous lecture)
    - This gives rise to methods such as Quadratic classifiers
  - Ignore the underlying distribution, and attempt to separate the data geometrically
    - This gives rise to methods such as perceptrons or support vector machines
- Due to time constraints, here we will review two simple techniques
  - Quadratic classifiers
  - K nearest neighbors

Quadratic or Gaussian classifiers

- A Quadratic classifier results when we assume that the likelihood function follows a Gaussian distribution

$$p(x|\omega_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}$$

- From which the posterior can be estimated as

$$P(\omega_i|x) \propto P(\omega_i)\frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)}$$

- Taking logs, we obtain

$$\log P(\omega_i|x) = -\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i) + \log P(\omega_i) - \frac{1}{2}\log|\Sigma_i| - \frac{d}{2}\log 2\pi$$

- Denoting $d_i(x) = \log P(\omega_i|x)$, we then obtain the rule

$$k = \arg\max_i d_i(x)$$

- In other words, assign example x with the class with largest $d_i(x)$
  - The function $d_i(x)$ is known as a discriminant function

– To simplify the classification rule, we can eliminate terms in the discriminant function that are independent of the class

$$d_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) + \log P(\omega_i) - \frac{1}{2}\log|\Sigma_i|$$

– Assuming equiprobable classes, we can then drop $\log P(\omega_i)$

$$d_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i) - \frac{1}{2}\log|\Sigma_i|$$

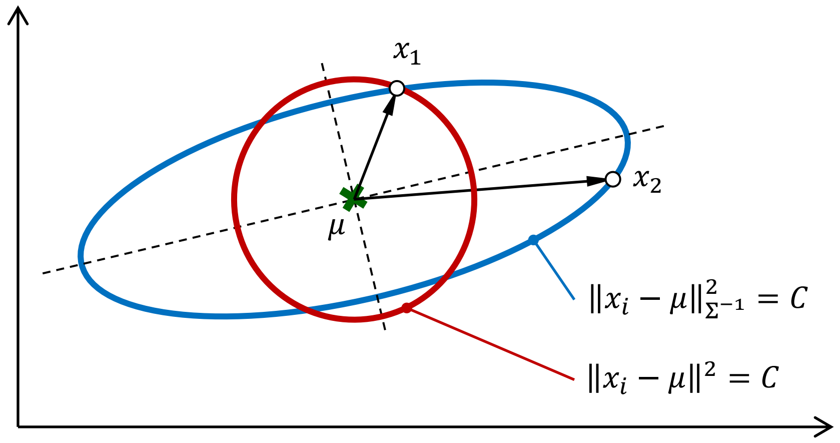
– By making additional assumptions, we can further simplify $d_i(x)$

– **Case 1**: assume $\Sigma_i = I\sigma^2$
  - In this case, the discriminant function reduces to the Euclidean distance

$$d_i(x) = -\frac{1}{2\sigma^2}(x-\mu_i)^2$$

  - This rule is known as the minimum-distance nearest-mean classifier
  - It can be shown that the resulting decision boundary is linear

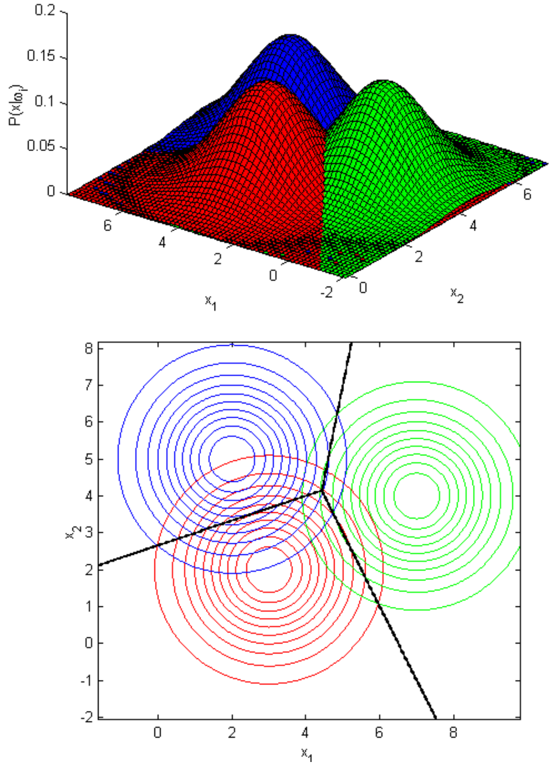
- **Case 2**: assume $\Sigma_i = \Sigma$
  - In this case, the discriminant function reduces to
$$d_i(x) = -\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)$$
  - Which is also a nearest-mean classifier, but in this case uses the *Mahalanobis distance* instead of the Euclidean distance
  - It can be shown that the resulting decision boundary is linear

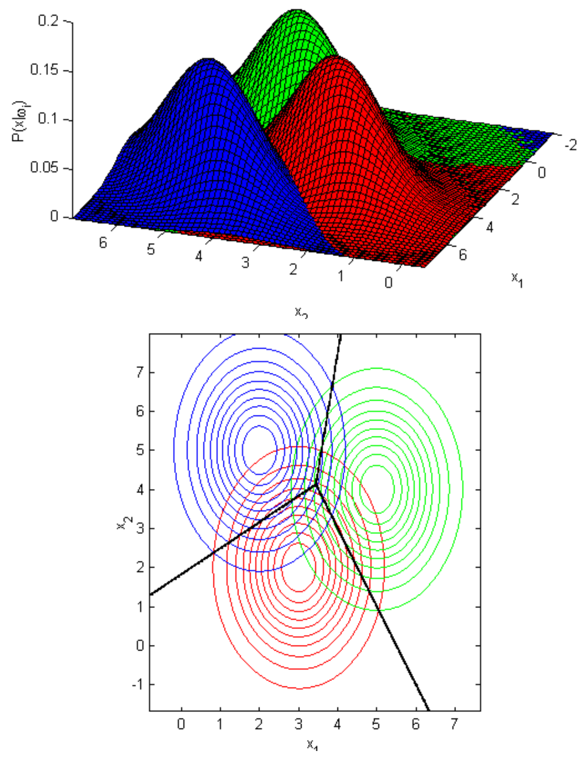


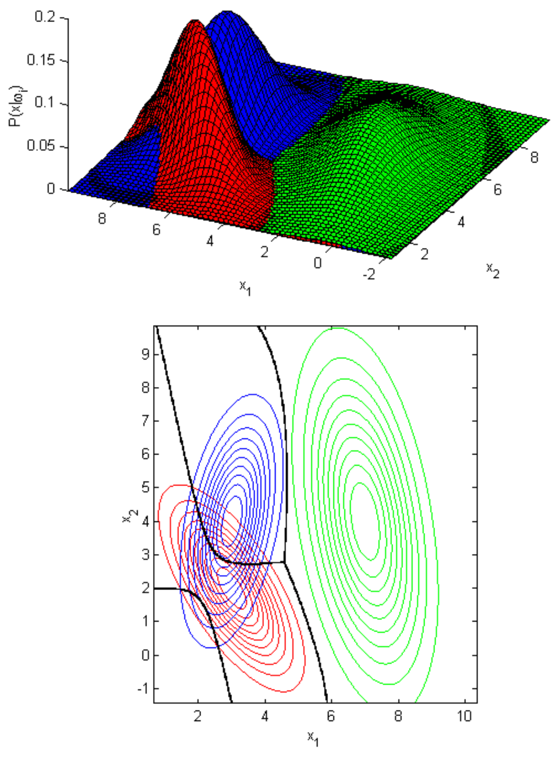
Case 1

Case 2

General case

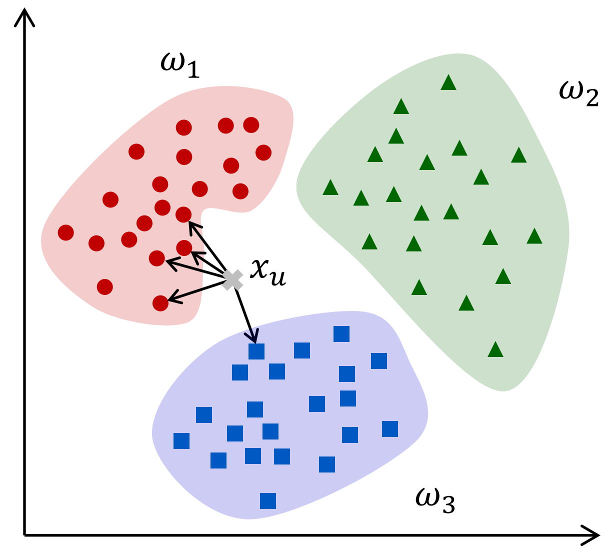






k nearest neighbors (kNN) classifier

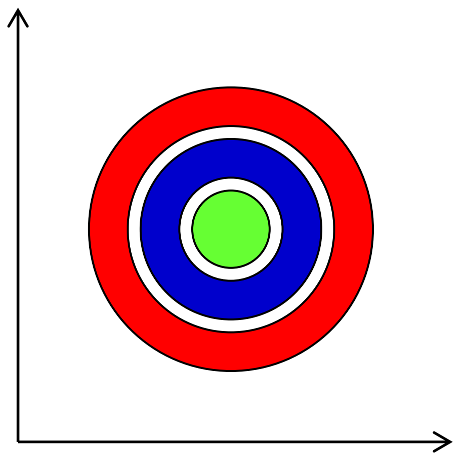
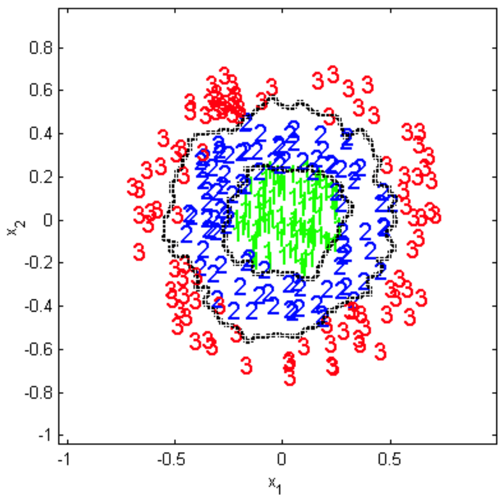
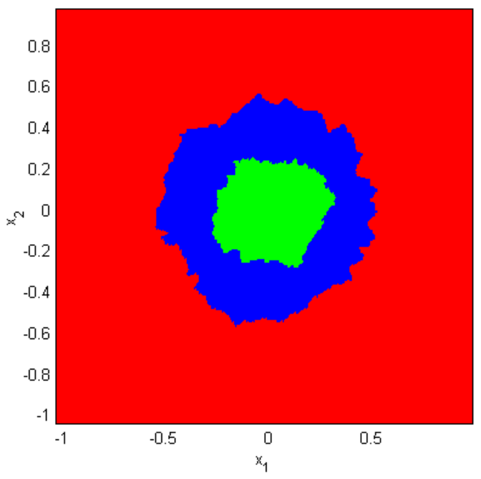
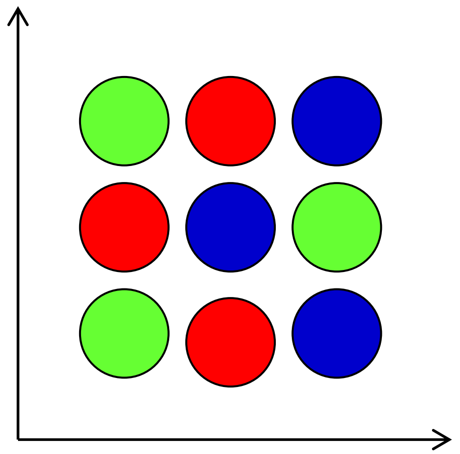
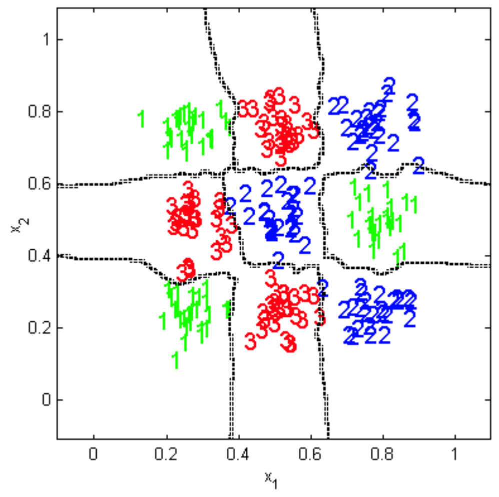
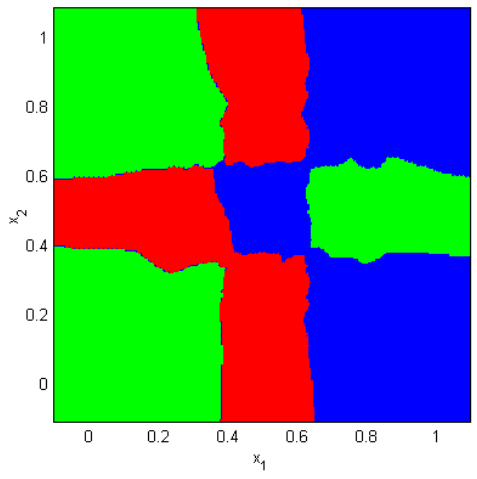
- kNN is a very intuitive method that classifies unlabeled examples based on their similarity to examples in the training set
    - Given unlabeled example x_u, find the k closest examples in the training set and assign x_u to the most represented class among the k-subset
- The kNN classifier only requires:
    - An integer k
    - A set of labeled examples (training data)
    - A "closeness" measure, generally Euclidean



- Advantages
  - Analytically tractable
  - Simple implementation
  - Nearly optimal in the large sample limit ($n \rightarrow \infty$)

$$P[error]_{Bayes} < P[error]_{1NN} < 2P[error]_{Bayes}$$

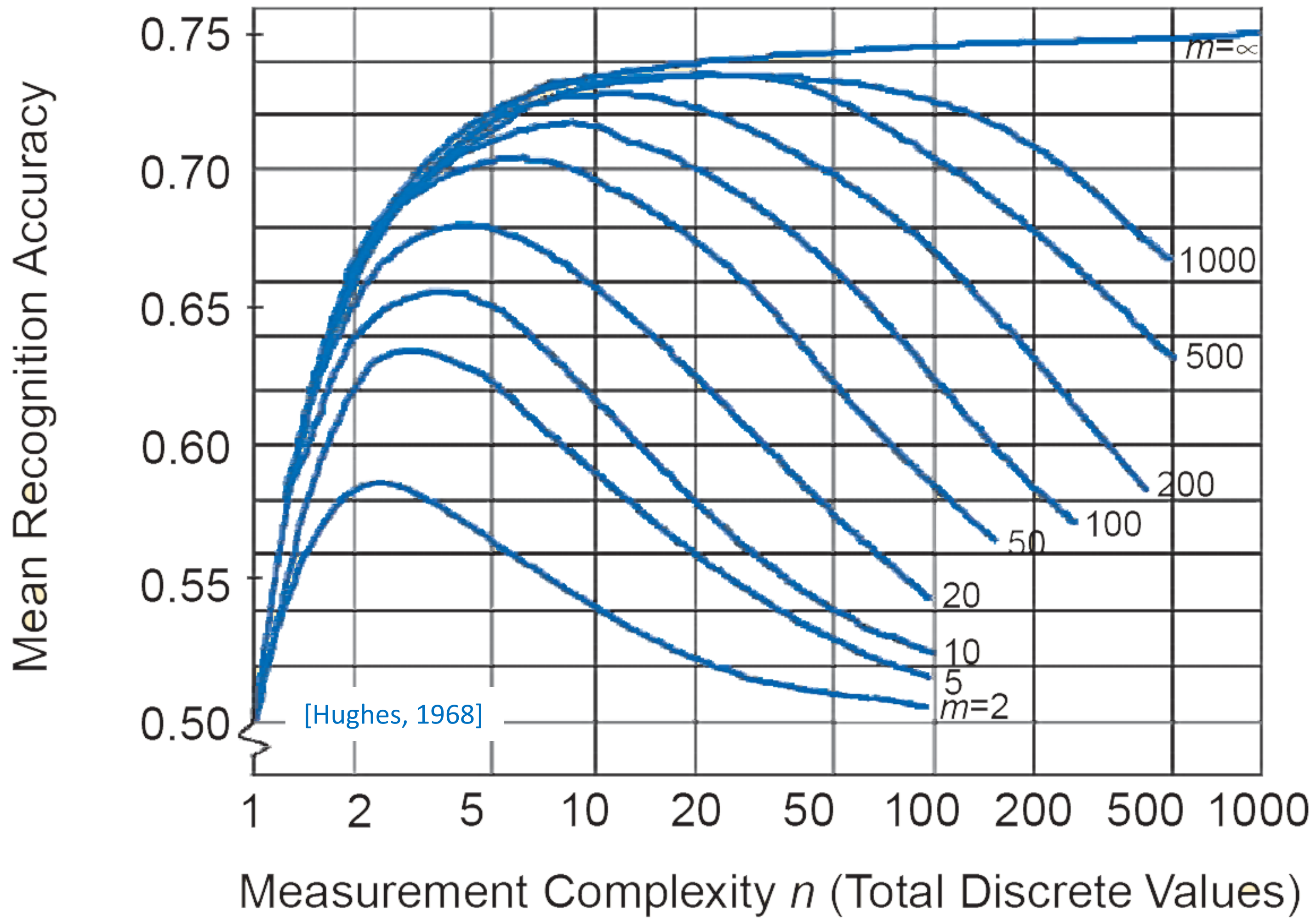
  - Uses local information, which can yield highly adaptive behavior
  - Lends itself very easily to parallel implementations
- Disadvantages
  - Large storage requirements
  - Computationally intensive recall
  - <u>Highly susceptible to the curse of dimensionality</u>
- 1NN versus kNN
  - The use of large values of k has two main advantages
    - Yields smoother decision regions
    - Provides probabilistic information
  - However, too large a value of k is detrimental
    - It destroys the locality of the estimation
    - In addition, it increases the computational burden



x2
x1

x2
x1


x2
x1

x2
x1

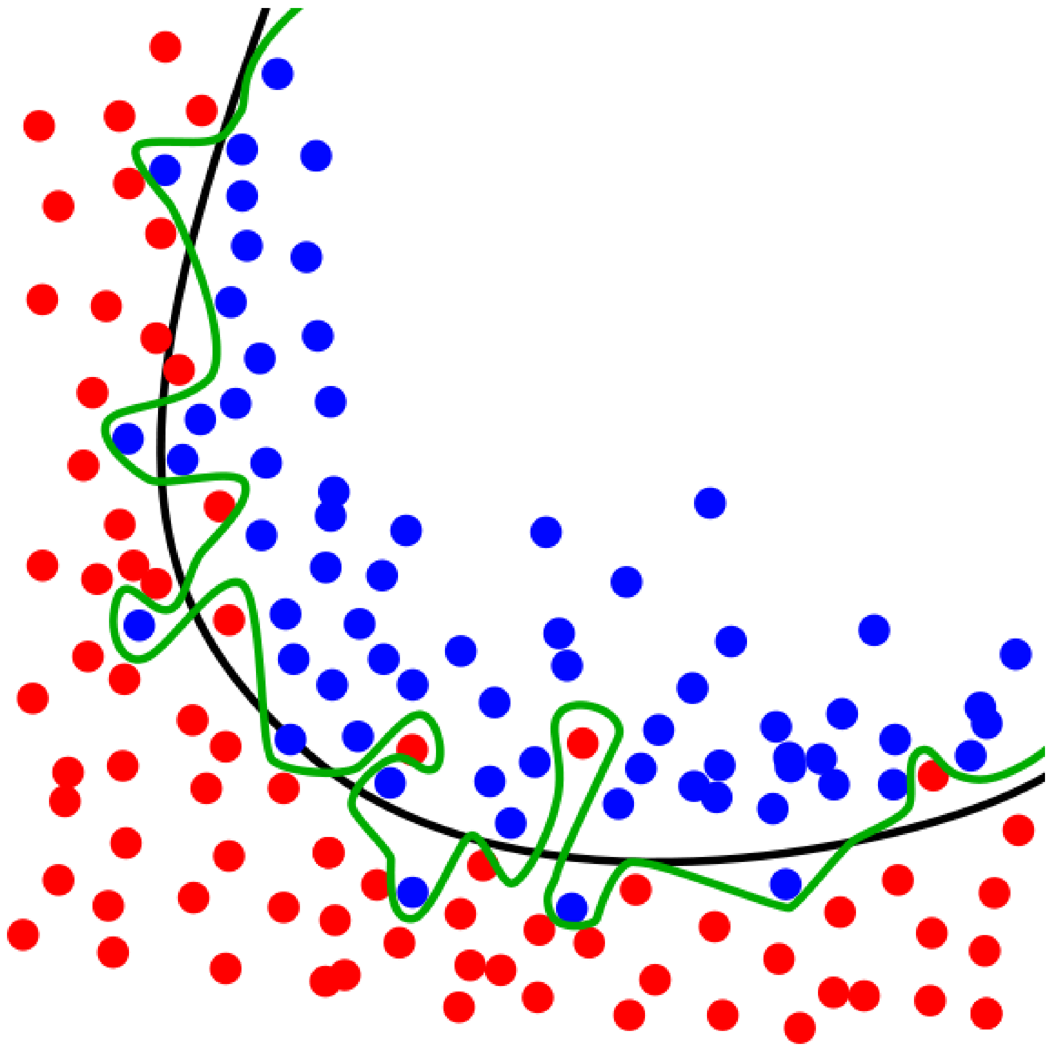
Dimensionality reduction

The curse of dimensionality

- The number of examples needed to accurately train a classifier grows exponentially with the dimensionality of the model
  - In theory, information provided by additional features should help improve the model's accuracy
  - In reality, however, additional features increase the risk of *overfitting*, i.e., memorizing noise in the data rather than its underlying structure
- The curse of dimensionality is an issue of trainability
  - For a given sample size, there is a maximum number of features above which the classifier's performance degrades rather than improves
  - In most cases, the additional information that is lost by discarding some features is (more than) compensated by a more accurate mapping in the lower-dimensional space



Mean Recognition Accuracy
0.75
0.70
0.65
0.60
0.55
0.50
1
2
5
10
20
50
100
200
500
1000
Measurement Complexity *n* (Total Discrete Values)
m=∞
1000
500
200
100
50
20
10
5
m=2
[Hughes, 1968]



http://commons.wikimedia.org/wiki/File:Overfitting.svg

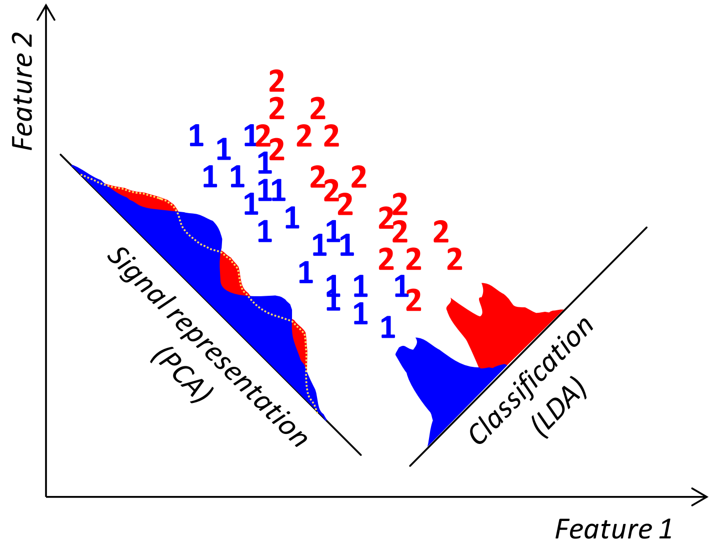
Dimensionality reduction

- How do we beat the curse of dimensionality?
  - By incorporating prior knowledge (e.g., parametric models)
  - By enforcing smoothness in the target function (e.g., regularization)
  - By reducing the dimensionality (e.g., feature selection/extraction)
- Here we focus on the last option

- Dimensionality reduction methods can be broadly grouped into
  - Feature extraction methods: creating a subset of new features by combinations of the existing features $y = f(x)$
    - Methods of this type include Principal Components Analysis (PCA) and Fisher's Linear Discriminant Analysis (LDA)
  - Feature selection methods: choosing a subset of all the features
    - Methods of this kind include sequential selection techniques (forward, backward), and won't be reviewed here

Feature extraction

- Two types of criteria are commonly used
  - Signal representation: The goal of feature selection is to accurately represent the samples accurately in a lower-dimensional space
  - Classification: The goal of feature selection is to enhance the class-discriminatory information in the lower-dimensional space
- Within the realm of linear feature extraction $(y = f(x) = Ax)$, two techniques are generally used
  - Principal Components Analysis, which we mentioned earlier in terms of the Karhunen-Loewe transform
  - Fisher's Linear Discriminants Analysis, which shares strong connections with the quadratic classifiers we reviewed earlier



Principal Components Analysis

- Project the data onto the eigenvectors v_i corresponding to the largest eigenvalues λ_i of the data's covariance matrix Σ

$$y = Ax$$
$$A = [v_1 v_2 \dots v_M]$$
$$\lambda_i v_i = \Sigma v_i$$

- PCA finds orthogonal directions of largest variance
- If data is Gaussian, PCA finds independent axes; otherwise, PCA simply de-correlates the features
- However, directions of high variance do not necessarily contain discriminatory information

Linear Discriminants Analysis

– Define scatter matrices

- Within class

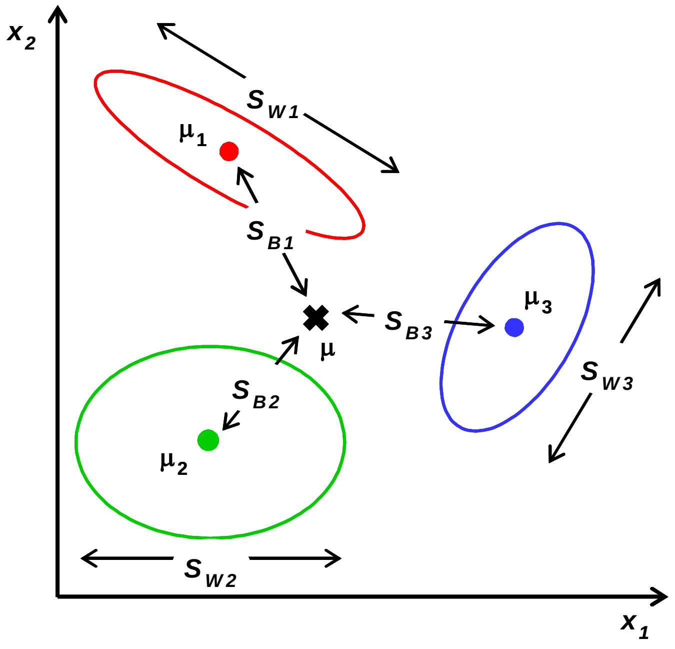
$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} \sum_{x \in \omega_i} (x - \mu_i)(x - \mu_i)^T$$

- Between class

$$S_B = \sum_{i=1}^{C} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

- Then maximize ratio

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|}$$



- Solution
  - Optimal projections are the eigenvectors of the largest eigenvalues of the generalized eigenvalue problem

$$(S_B - \lambda_i S_W) v_i = 0$$

- Notes
  - $S_B$is the sum of C matrices of rank one or less, and the mean vectors are constrained by $\Sigma_i \mu_i = \mu$
  - Therefore, S_B will be at most of rank $C-1$, and LDA produces at most $C-1$ feature projections
- Limitations
  - Overfitting
  - Information not in the mean of the data
  - Classes significantly non Gaussian

ex11p1.m

Computing principal components analysis

Computing linear discriminants analysis

Clustering

Supervised vs. unsupervised learning

- The methods discussed so far have focused on classification
  - A pattern consisted of a pair of variables $\{x, \omega\}$, where x was a feature vector, and ω was the concept behind the observation
  - Such pattern recognition problems are called supervised (training with a teacher) since the system is given the correct answer
- Now we explore methods that operate on unlabeled data
  - Given a collection of feature vectors $X = \{X_1, X_2 \dots X_n\}$ without class labels ω_i, the goal is build a model that captures the structure of the data
  - These methods are called unsupervised (training without a teacher) since they are not provided the correct answer
- In particular, we will explore two methods that will later be reused when we discussed hidden Markov models
  - Gaussian mixture models (and the EM algorithm), and
  - K means clustering

Gaussian mixture models

- Consider the problem of modeling a pdf given a dataset of examples
  - If the form of the underlying pdf is known (e.g. Gaussian), the problem can be solved through parameter estimation
  - If the form of the pdf is unknown, the problem must to be solved with non-parametric density estimation methods such as Parzen windows
- We will now consider an alternative density estimation method: modeling the pdf with a mixture of parametric densities
  - These methods are sometimes known as semi-parametric
  - In particular, we will focus on mixture models of Gaussian densities

$$p(x|\theta) = \sum_{c=1}^{C} p(x|\theta_c) p(\theta_c)$$

- The GMM can be framed in terms of the ML criterion
  - Given a dataset of examples $X = \{X_1, X_2 \dots X_n\}$, we seek to find model parameters θ that maximize the log likelihood of the data

$$\hat{\theta}_{ML} = \arg\max_{\theta} p(X|\theta) = \arg\max_{\theta} \left[\sum_{i=1}^{n} \log\ p(x_i|\theta) \right] =$$
$$= \arg\max_{\theta} \left[\sum_{i=1}^{n} \log \sum_{c=1}^{C} p(x_i|\theta_c)\, p(\theta_c) \right]$$

  - where $\theta_c = \{\mu_c, \Sigma_c\}$ and $p(\theta_c)$ are the parameters and mixing coefficient of the c-th mixture component, respectively
- We could try to find the maximum of this function by differentiation
  - For $\Sigma_i = I\sigma_i$, the solution becomes [Bishop, 1995]

$$\frac{\partial[\quad]}{\partial \mu_c} = 0 \Rightarrow \hat{\mu}_c = \frac{\sum_{i=1}^{n} p(\theta_c|x_i) x_i}{\sum_{i=1}^{n} p(\theta_c|x_i)}$$
$$\frac{\partial[\quad]}{\partial \sigma_c} = 0 \Rightarrow \hat{\sigma}_c^2 = \frac{1}{d} \frac{\sum_{i=1}^{n} p(\theta_c|x_i) \|x_i - \hat{\mu}_c\|^2}{\sum_{i=1}^{n} p(\theta_c|x_i)}$$
$$\frac{\partial[\quad]}{\partial p(\theta_c)} = 0 \Rightarrow \hat{p}(\theta_c) = \frac{1}{n} \sum_{i=1}^{n} p(\theta_c|x_i)$$

- Notice that the previous equations are not a closed form solution
  - Parameters $\{\mu_c, \Sigma_c, p(\theta_c)\}$ also appear on the RHS because of Bayes rule!
  - Thus, this is a highly non-linear coupled system of equations
- These expressions, however, suggest that we may be able to use a fixed-point algorithm to find the maxima
  1. Begin with some value of the model parameters (the “old” values)
  2. Evaluate the RHS to obtain “new” values for the parameters
  3. Let these “new” values become the “old” ones and repeat the process
- Surprisingly, an algorithm this simple can be found that is guaranteed to increase the log-likelihood with every iteration
  - This example represents a particular case of a more general procedure known as the *Expectation-Maximization* algorithm

– A derivation of the update equations for the full-covariance mixture model can be found in [Nabney, 2002]
  - The final equations are provided here for those of you interested in experimenting with mixture models

$$p^{(k}(\theta_c) = \frac{1}{n}\sum_{i=1}^{n} p^{(k-1}(\theta_c|x_i)$$

$$\mu_c^{(k} = \frac{\sum_{i=1}^{n} p^{(k-1}(\theta_c|x_i)x_i}{\sum_{i=1}^{n} p^{(k-1}(\theta_c|x_i)}$$

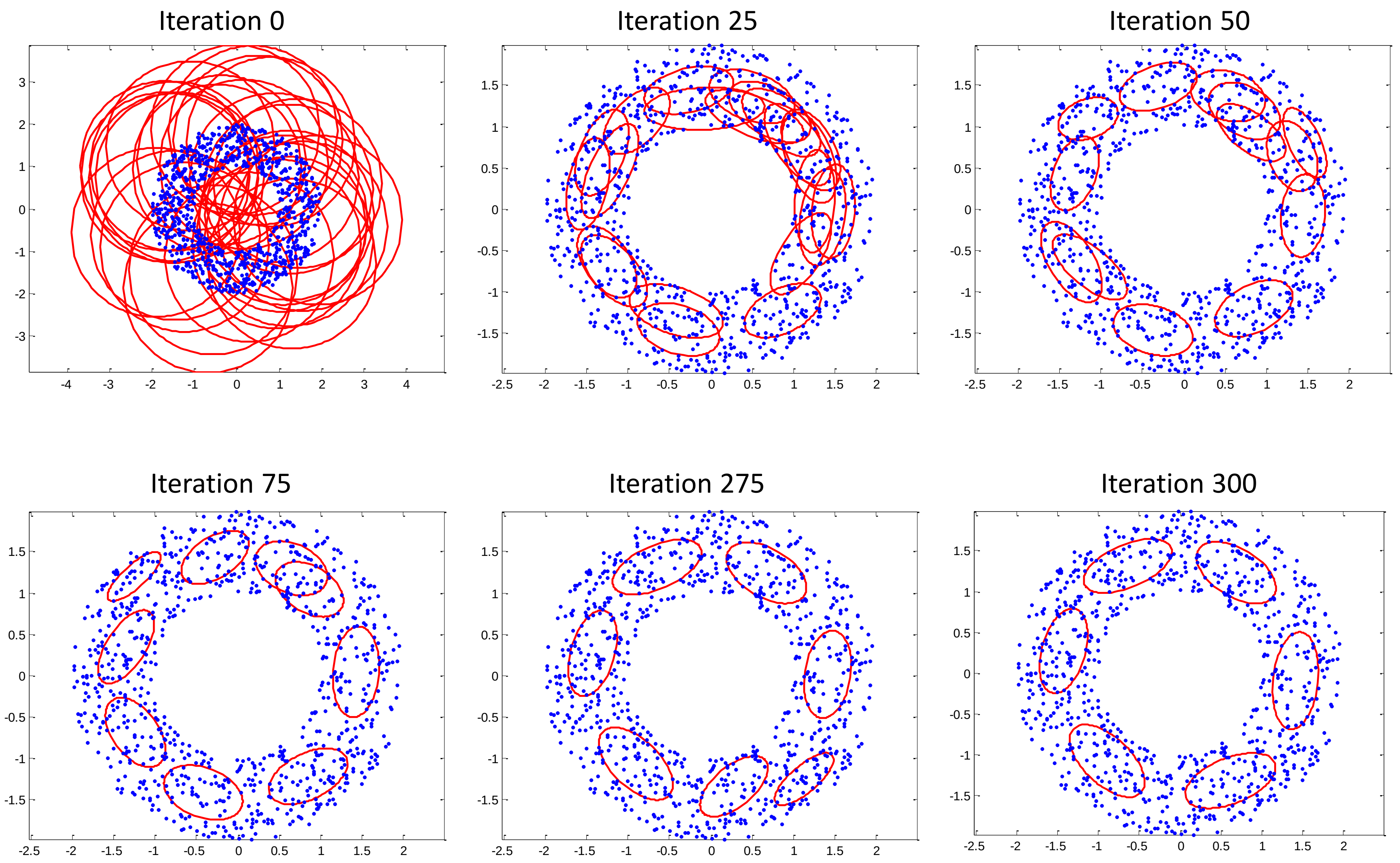
$$\Sigma_c^{(i} = \frac{\sum_{i=1}^{n} p^{(k-1}\,(\theta_c|x_i)\left(x_i-\mu_c^{(k}\right)\left(x_i-\mu_c^{(k}\right)^T}{\sum_{i=1}^{n} p^{(k-1}(\theta_c|x_i)}$$

  - Notice where the new parameters $\theta^{(k}$ and old parameters $\theta^{(k-1}$ appear on the RHS and compare these expressions to those earlier for the univariate case

GMM example

- Training set: $n = 900$ examples from a uniform pdf inside an annulus
- Model: GMM with $C = 30$ Gaussian components
- Training procedure
  - Gaussians centers initialized by choosing 30 arbitrary training examples
  - Covariance matrices initialized to be diagonal, with large variance compared to that of the training data
  - To avoid singularities, at every iteration the covariance matrices computed with EM were regularized with a small multiple of the identity matrix
  - Components whose mixing coefficients fell below a threshold are removed

- Illustrative results are provided in the next slide


Iteration 0
Iteration 25
Iteration 50
Iteration 75
Iteration 275
Iteration 300

k-means clustering

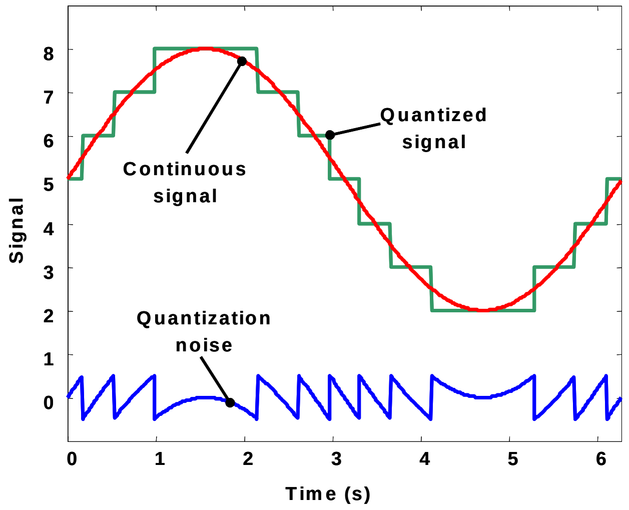
- The k-means algorithm is a simple procedure that attempts to group a collection of unlabeled examples $X = \{x_1 \dots x_n\}$ into one of C clusters
  - k-means seeks to find compact clusters, measured as

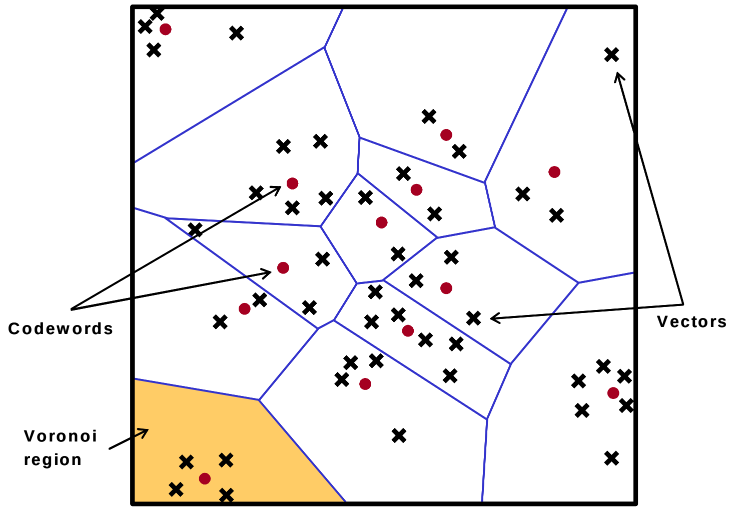
$$J_{MSE} = \sum_{c=1}^{C} \sum_{x \in \omega_c} \|x - \mu_c\|^2 \; ; \; \mu_c = \frac{1}{n_c} \sum_{x \in \omega_c} x$$

  - It can be shown that k-means is a special case of the GMM-EM algorithm
- Procedure

1. Define the number of clusters
2. Initialize clusters by
   a) an arbitrary assignment of examples to clusters or
   b) an arbitrary set of cluster centers (i.e., use some examples as centers)
3. Compute the sample mean of each cluster
4. Reassign each example to the cluster with the nearest mean
5. If the classification of all samples has not changed, stop, else go to step 3

– k-means is widely used in DSP for *Vector Quantization*
  - Unidimensional signal values are usually quantized into a number of levels (typically a power of 2 so the signal can be transmitted in binary format)
  - The same idea can be extended for multiple channels
    - However, rather than quantizing each separate channel, we can obtain a more efficient signal coding by quantizing the overall multidimensional vector into a small number of multidimensional prototypes (cluster centers)
  - Each cluster center is called a *codeword* and the collection of codewords is called a *codebook*





ex11p2.m
k-means clustering