Pattern Recognition

Xiaojun Qi

-- REU Site Program in CVMA
(2010 Summer)

Outline

- Neural Networks
- Support Vector Machines
- Hidden Markov Model
- Linear Discriminant Analysis

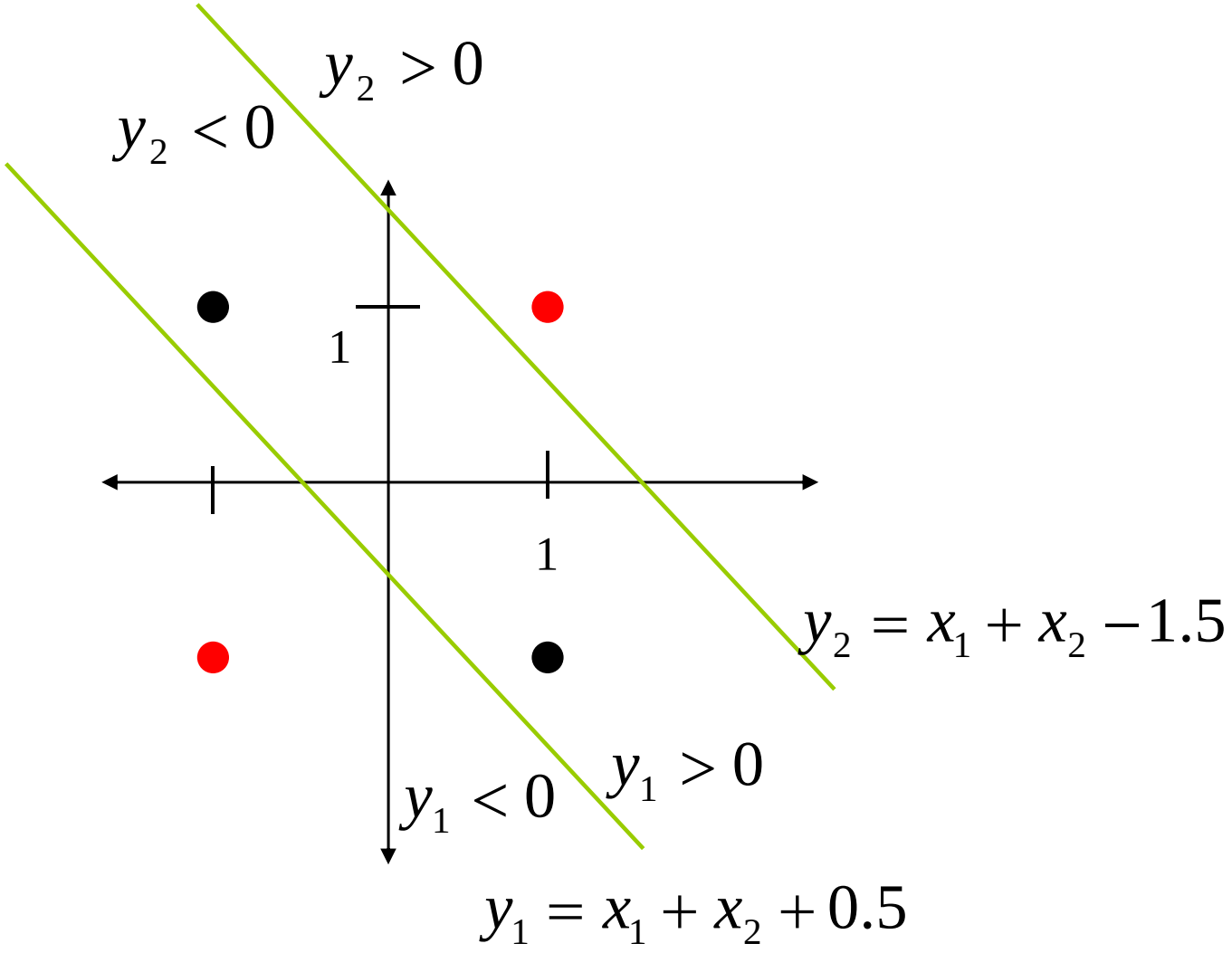
Introduction: Neural Network

- There are many problems for which linear discriminants are insufficient for minimum error
- Multilayer neural network (a.k.a., Feedforward NN) is an approach that learns the nonlinearity (i.e., the weights) at the same time as the linear discriminant
  - Adopt the **backpropagation algorithm** or **generalized delta rule** (extension of Least Mean Square algorithm) for training of weights
  - Neural networks are a **flexible heuristic technique** for doing statistical pattern recognition with complicated models
  - **Network architecture or topology** plays an important role for neural net classification and the optimal topology depends on the problem at hand

Introduction: NN (Cont.)

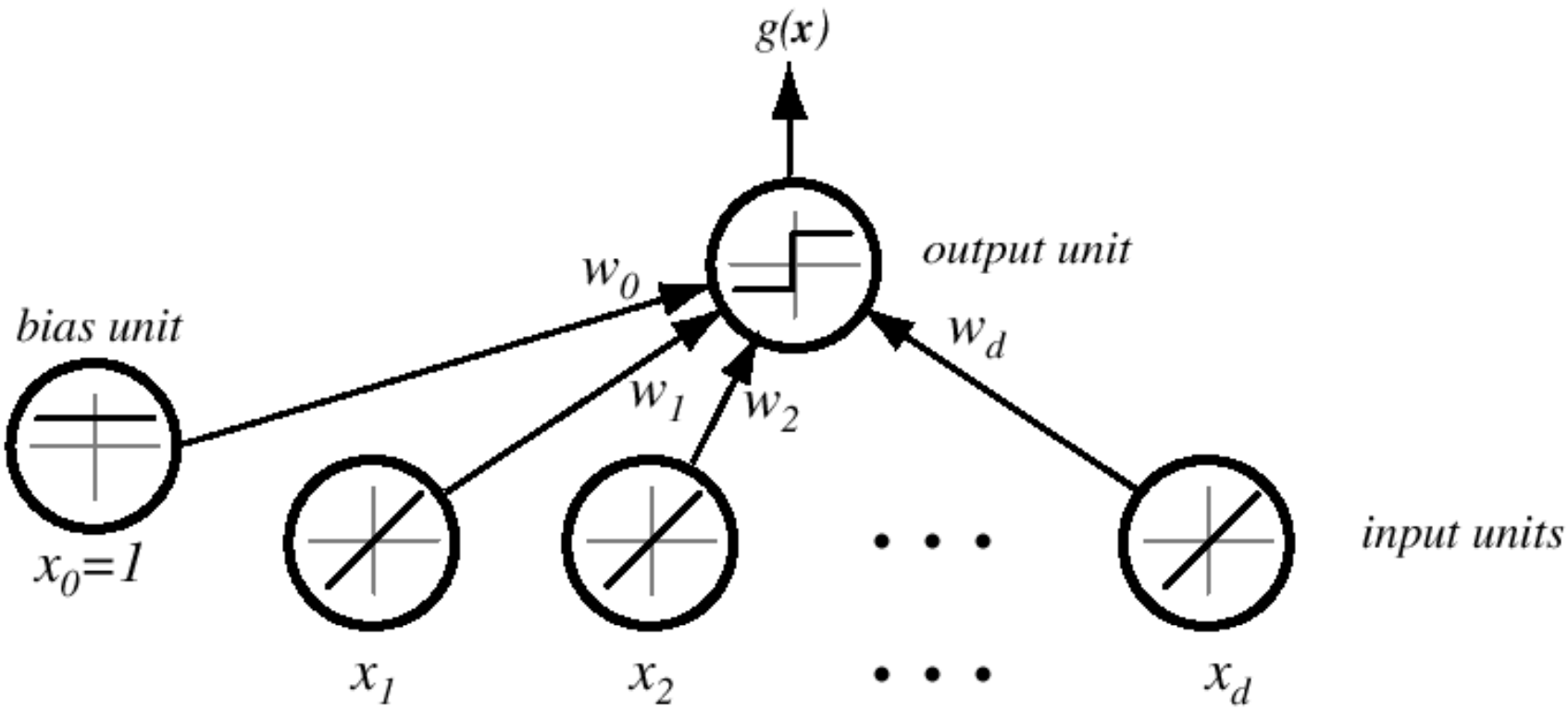
- The goal of training neural networks (NNs) is not to learn an exact representation of the training data itself, but rather to build a statistical model of the process which generates the data.
- In practical applications of a feedforward NN, if the network is over-fit to the noise on the training data, especially for the small-number training samples case, it will memorize training data and give poor generalization.
- Controlling an appropriate complexity of the network can improve generalization. There are two main approaches for this purpose:
  - Model selection
  - Regularization.

XOR Problem

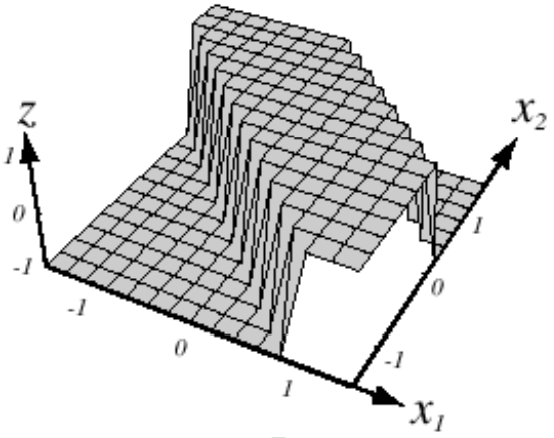


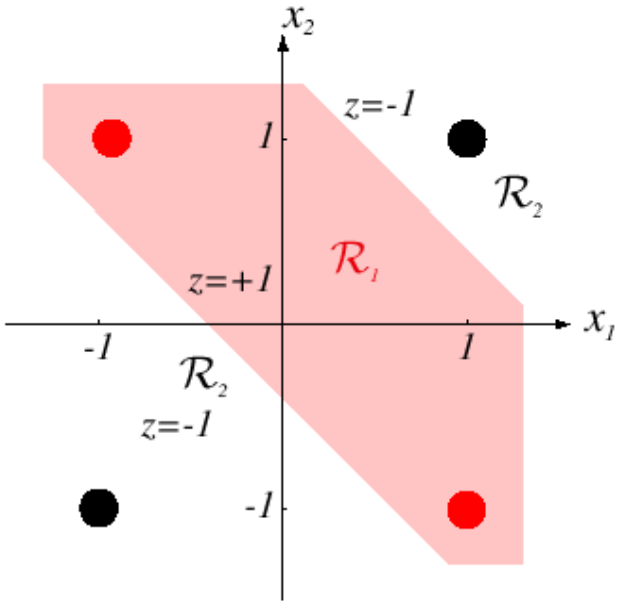
To be implemented by neural network

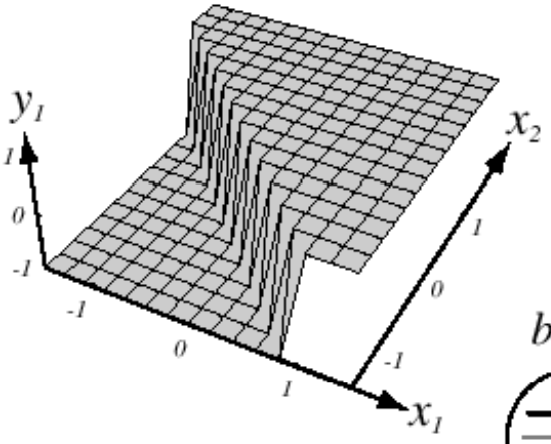
Two-Layer Network

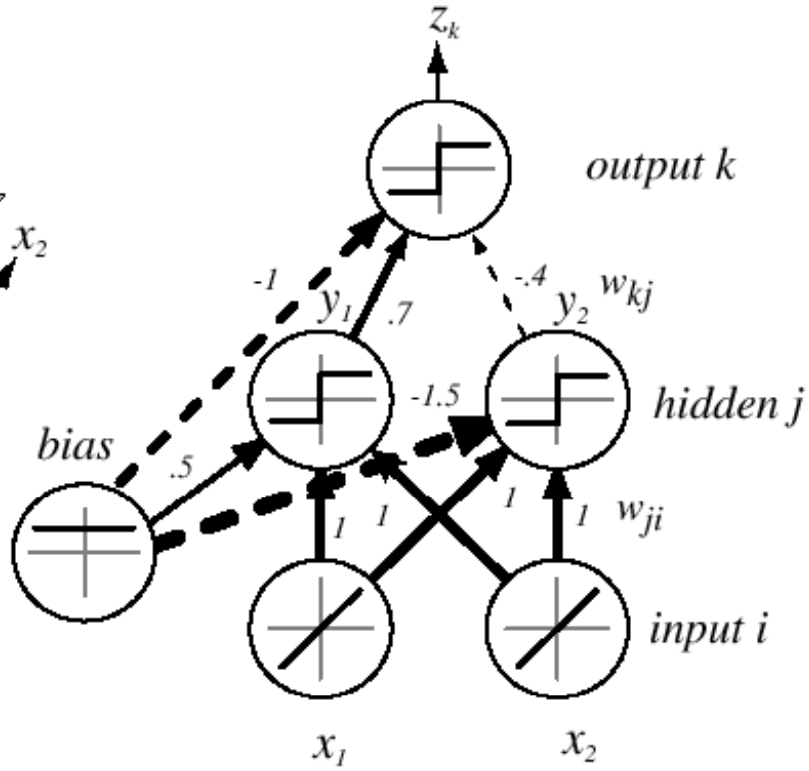


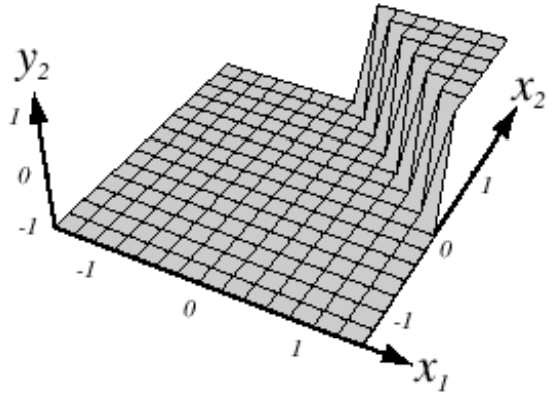
Three-Layer Network











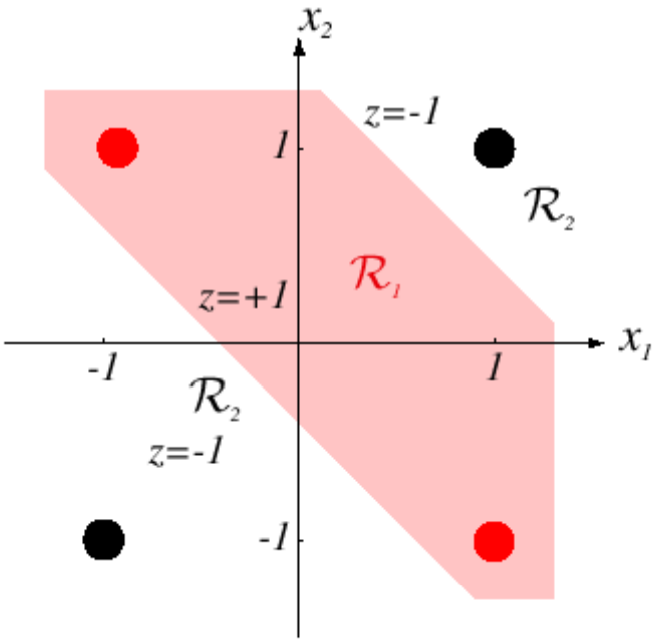


FIGURE 6.1. The two-bit parity or exclusive-OR problem can be solved by a three-layer network. At the bottom is the two-dimensional feature $x_1 x_2$-space, along with the four patterns to be classified. The three-layer network is shown in the middle. The input units are linear and merely distribute their feature values through multiplicative weights to the hidden units. The hidden and output units here are linear threshold units, each of which forms the linear sum of its inputs times their associated weight to yield *net*, and emits a +1 if this *net* is greater than or equal to 0, and −1 otherwise, as shown by the graphs. Positive or "excitatory" weights are denoted by solid lines, negative or "inhibitory" weights by dashed lines; each weight magnitude is indicated by the line's thickness, and is labeled. The single output unit sums the weighted signals from the hidden units and bias to form its *net*, and emits a +1 if its *net* is greater than or equal to 0 and emits a −1 otherwise. Within each unit we show a graph of its input-output or activation function—$f(net)$ versus *net*. This function is linear for the input units, a constant for the bias, and a step or sign function elsewhere. We say that this network has a 2-2-1 fully connected topology, describing the number of units (other than the bias) in successive layers. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Feedforward Operation and Classification

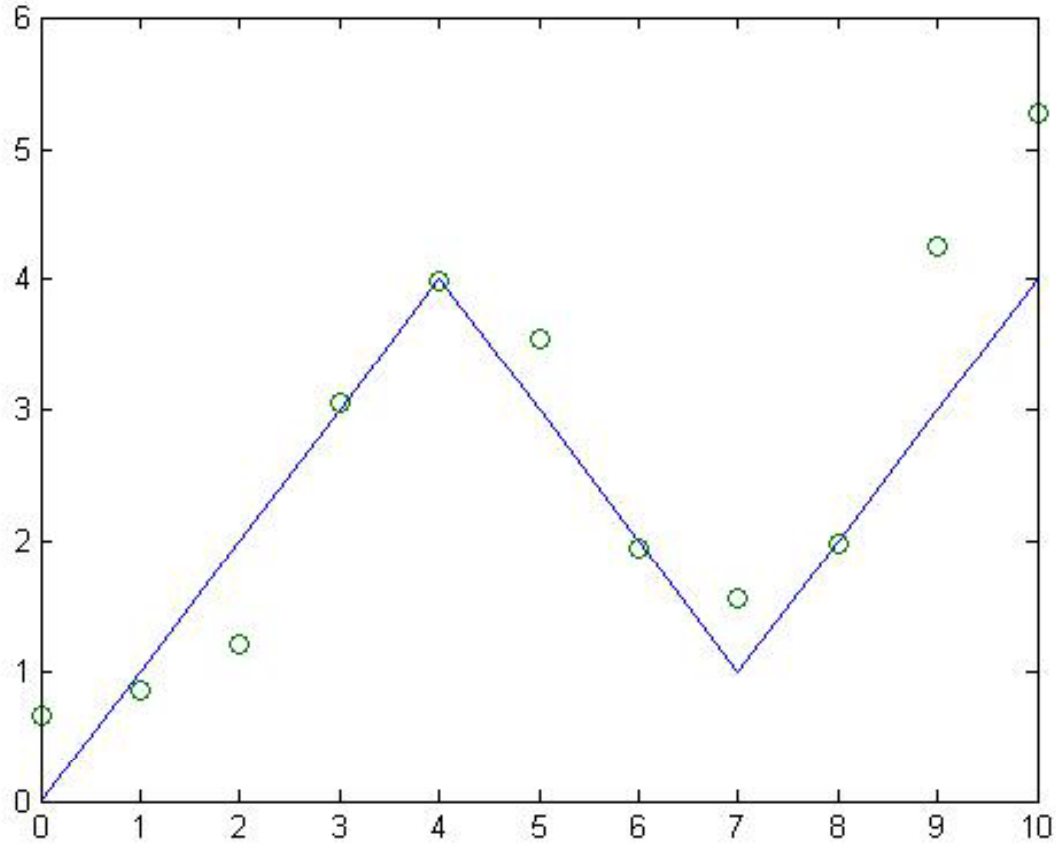
- Input, hidden, and output layers
  - input units -- represent the components of a feature vector
  - output units -- represent the number of categories or the *c* discriminant function values for classification
  - Bias unit -- connected to each unit other than the input units
- Layers are interconnected by modifiable weights

Introduction: Model Selection

- Model selection for a feedforward NN requires choosing the number of hidden neurons and connection weights. The common statistical approach to model selection is to estimate the generalization error for each model and to choose the model minimizing this error.
  - selecting or adjusting the complexity (number of hidden layers and number of neurons in each layer) of the network

Matlab Illustration

```
P = [0 1 2 3 4 5 6 7 8 9 10];  % input data containing one value
T = [0 1 2 3 4 3 2 1 2 3 4];    % output data containing the categorical info.
net = newff(P,T,5);                % one hidden layer with 5 neurons
net.trainParam.epochs = 50;
net = train(net,P,T);
Y = sim(net,P);
plot(P,T,P,Y,'o')
```



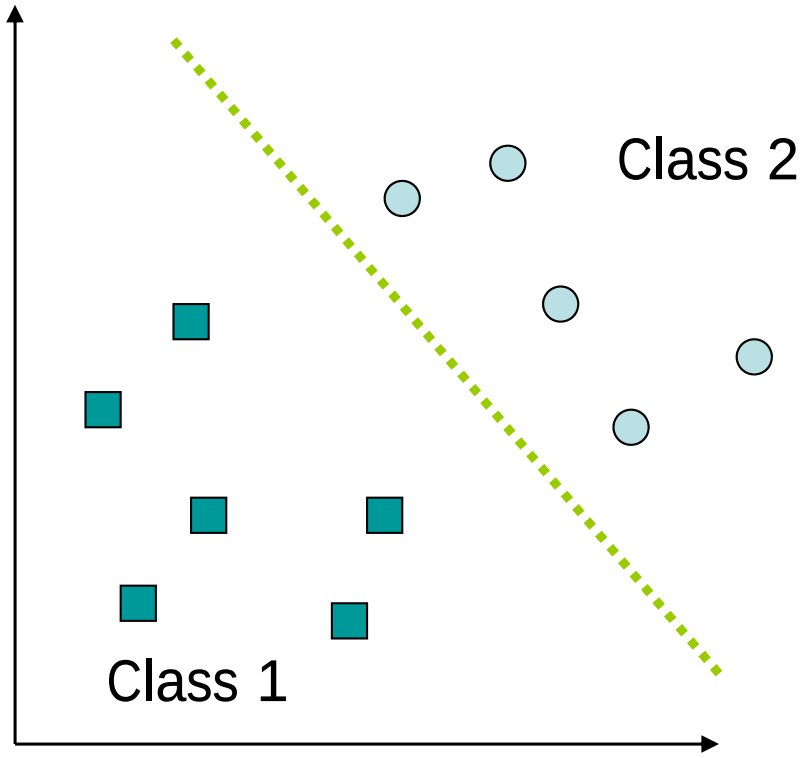
History of SVM

- SVM is a classifier derived from statistical learning theory by Vapnik and Chervonenkis
- SVM becomes famous when, using pixel maps as input, it gives accuracy comparable to sophisticated neural networks with elaborated features in a handwriting recognition task.
- Currently, SVM is closely related to:
  - Kernel methods, large margin classifiers, reproducing kernel Hilbert space, Gaussian process

Linear Support Vector Machines (LSVMs)

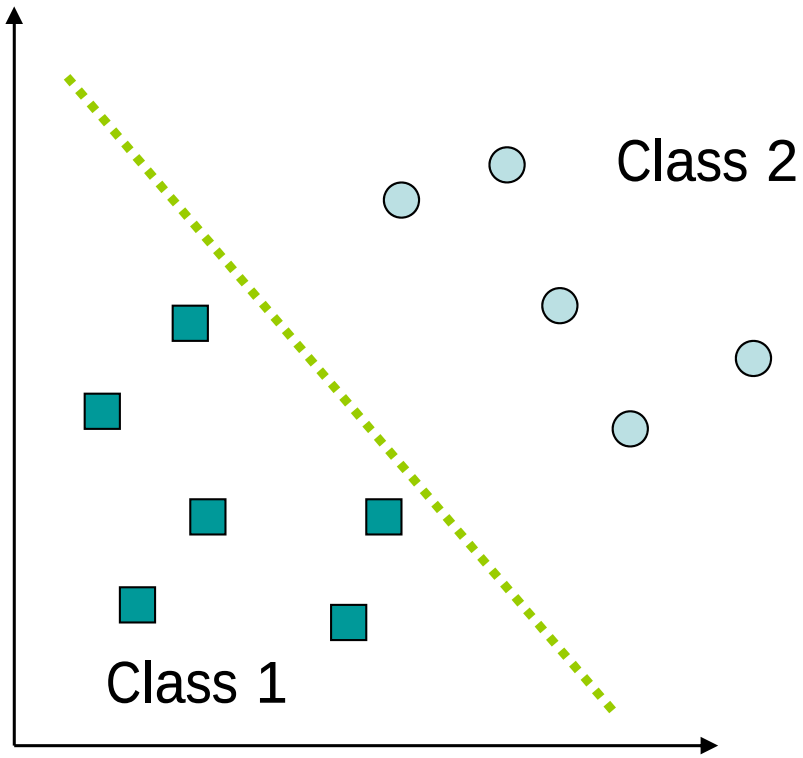
- LSVMs are linear pattern classifiers
- The goal is to find an **optimal** hyper-plane to separate two classes. Two important issues related to LSVMs are:
  - Optimization problem formulation
  - Statistical properties of the optimal hyper-plane

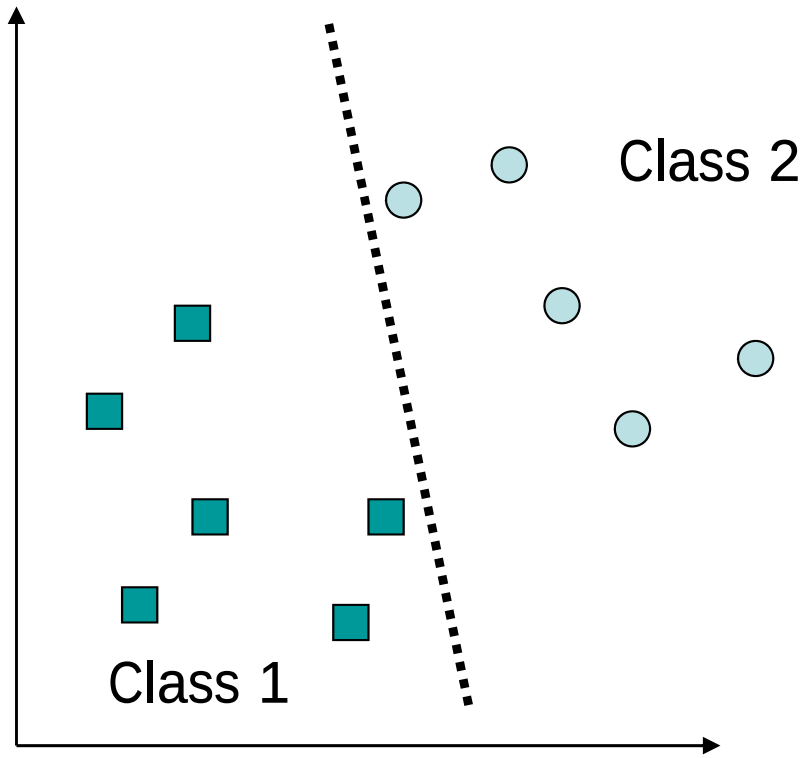
Two Class Problem: Linear Separable Case



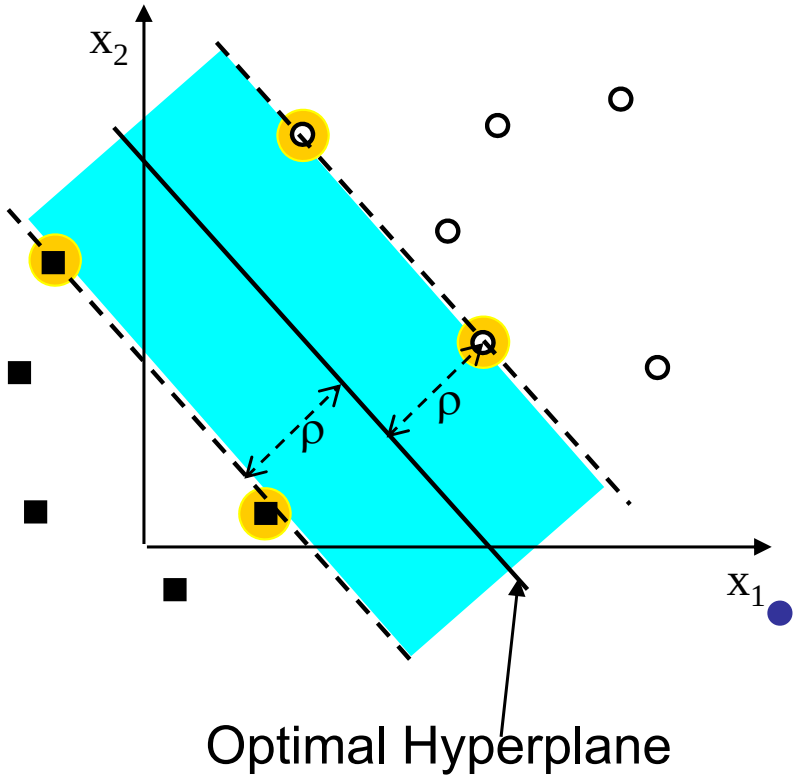
- Many decision boundaries can separate these two classes
- Which one should we choose?

Example of Bad Decision Boundaries





Optimal Hyper-plane and Support Vectors -- Linearly Separable Case



- Optimal hyper-plane should be in the center of the gap. It gives the largest margin (ρ) of separation between the classes. This optimal hyperplane bisects the shortest line between the convex hulls of the two classes. It is also orthogonal to this line.
- Support Vectors – A relative small subset of the data points on the boundaries (margins).

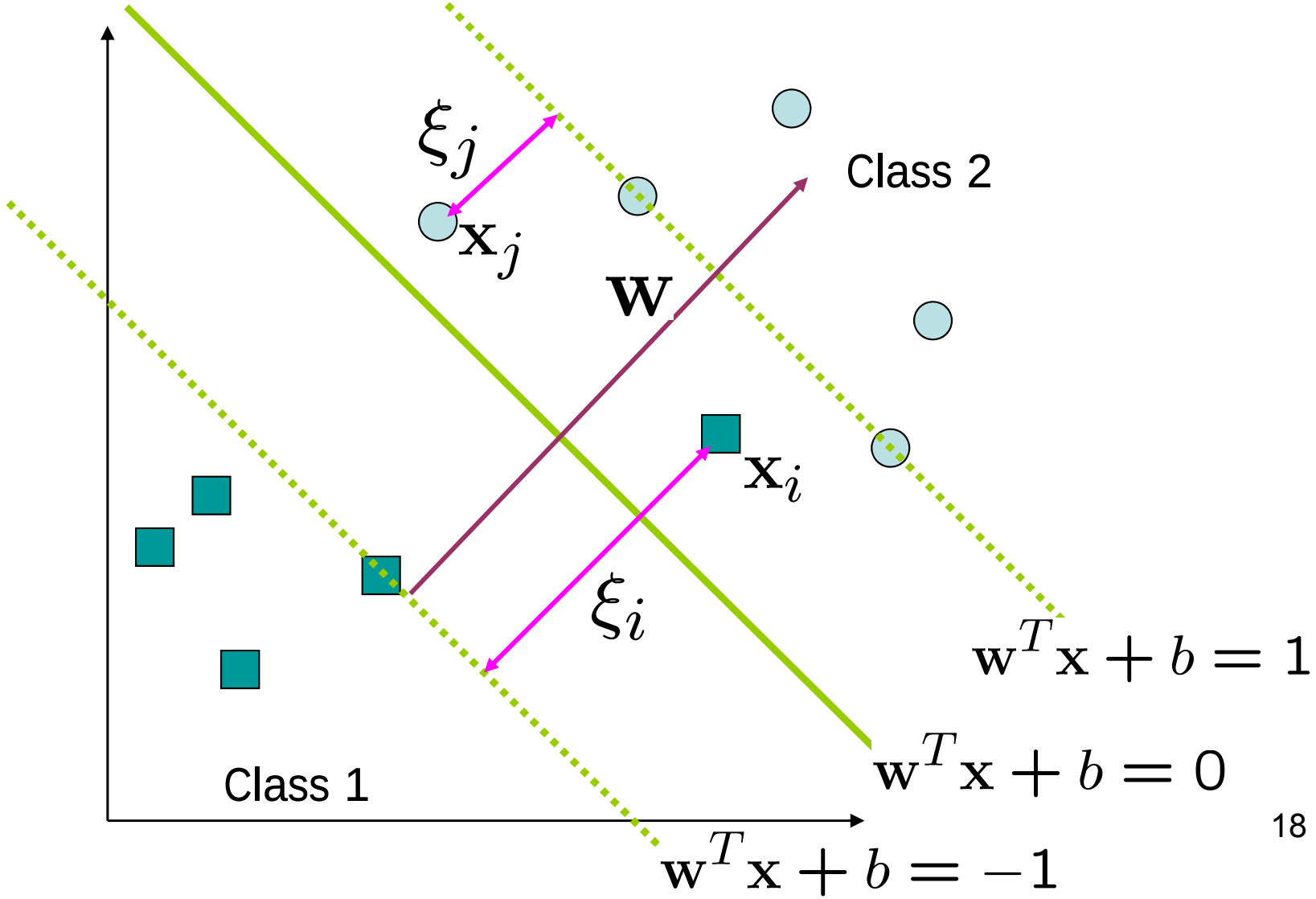
Support vectors alone can determine the optimal hyper-plane. That is, the optimal hyperplane is completely determined by these support vectors.

Soft Margin SVMs: Nonseparable Cases

- In a real problem it is unlikely that a line will exactly separate the data. Even if a curved decision boundary is possible (as it will be after adding the nonlinear data mapping), exactly separating the data is probably not desirable:
  - If the data has noise and outliers, a smooth decision boundary that ignores a few data points is better than one that loops around the outliers.

Soft Margin SVMs: Illustration

- We allow "error" ξ_i in classification

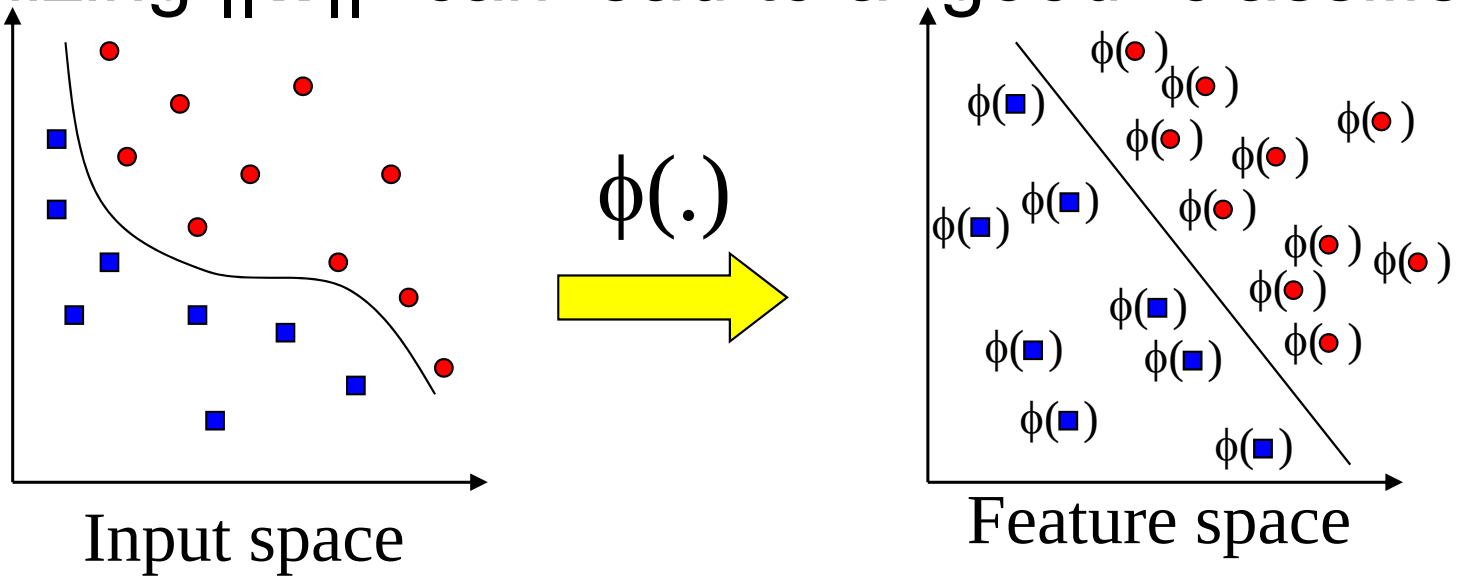


Extension to Non-linear Decision Boundary: Non-linear SVMs

- Key idea: transform $\mathbf{x}_i$ to a higher dimensional space to "make life easier"
  - Input space: the space that $\mathbf{x}_i$ are in
  - Feature space: the space of $\phi(\mathbf{x}_i)$ after transformation
- Why transform?
  - Linear operation in the feature space is equivalent to non-linear operation in input space
  - The classification task can be "easier" with a proper transformation. Example: XOR

Non-Linear SVMs (Cont.)

- Possible problem of the transformation
  - –High computation burden and hard to get a good estimate
- SVM solves the following two issues simultaneously
  - –Kernel tricks for efficient computation
  - –Minimizing $||\mathbf{w}||^2$ can lead to a "good" classifier



Matlab Illustration

```
% Select features for classification
    load fisheriris
    data = [meas(:,1), meas(:,2)];

% Extract the Setosa class (i.e., the label info)
    groups = ismember(species,'setosa');

% Randomly select training and test sets
    [train, test] = crossvalind('holdOut',groups);
    cp = classperf(groups);
```

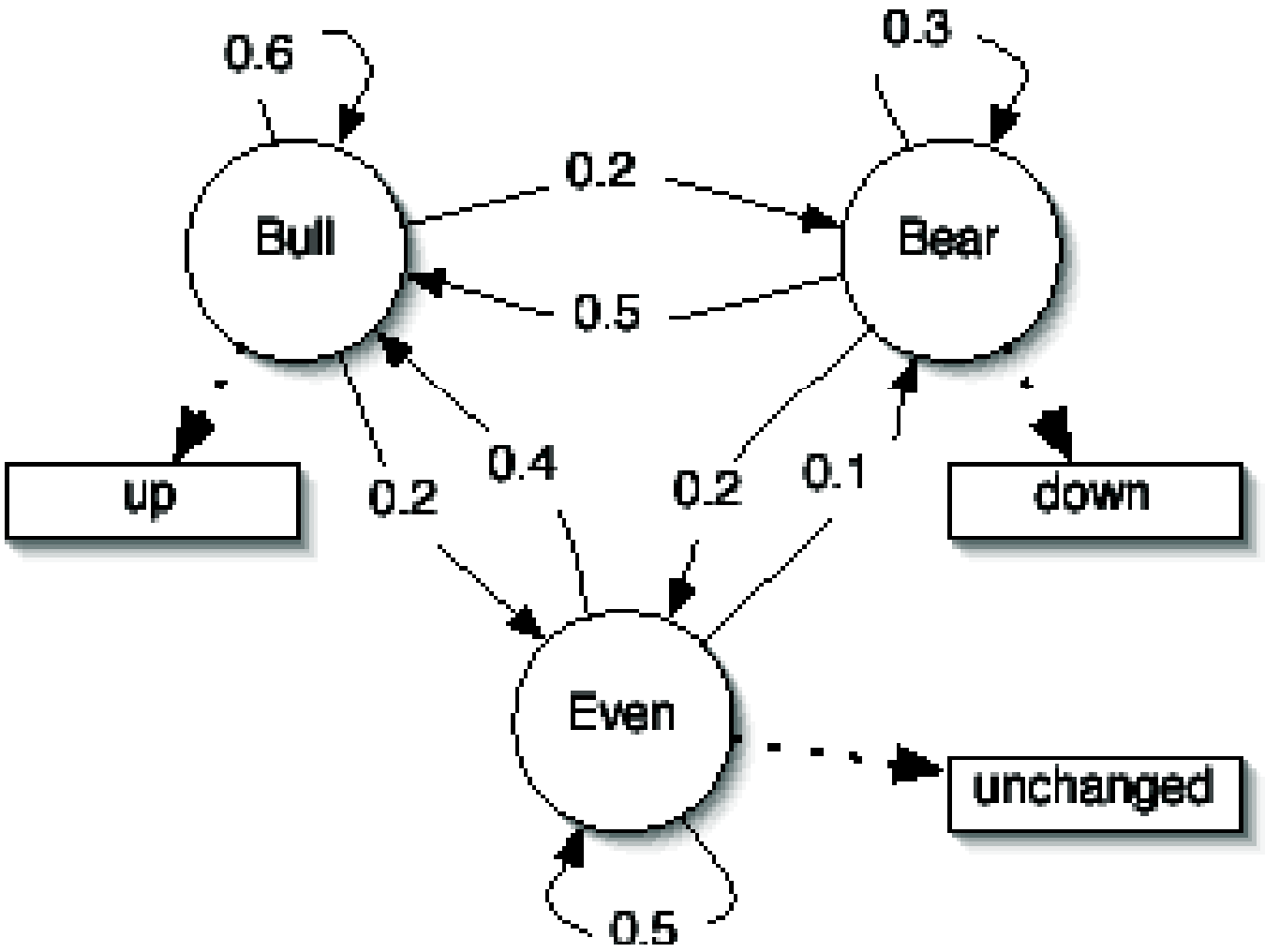
Matlab Illustration (Cont.)

```
% Use a linear SVM
  svmStruct =svmtrain(data(train,:),groups(train));

% Classify the test set using svmclassify
  classes = svmclassify(svmStruct,data(test,:));

% See how well the classifier performed
        classperf(cp,classes,test);
        cp.CorrectRate
```

Hidden Markov Model Introduction

- The Hidden Markov Model (HMM) is a popular statistical tool for modeling a wide range of time series data.
- In the context of **natural language processing** (NLP), HMMs have been applied with great success to problems such as part-of-speech tagging and noun-phrase chunking.
- It also provides useful techniques for **object tracking** and has been used quite widely in computer vision. Its recursive nature makes it well suited to real-time applications and its ability to predict provides further computational efficiency by reducing image search.

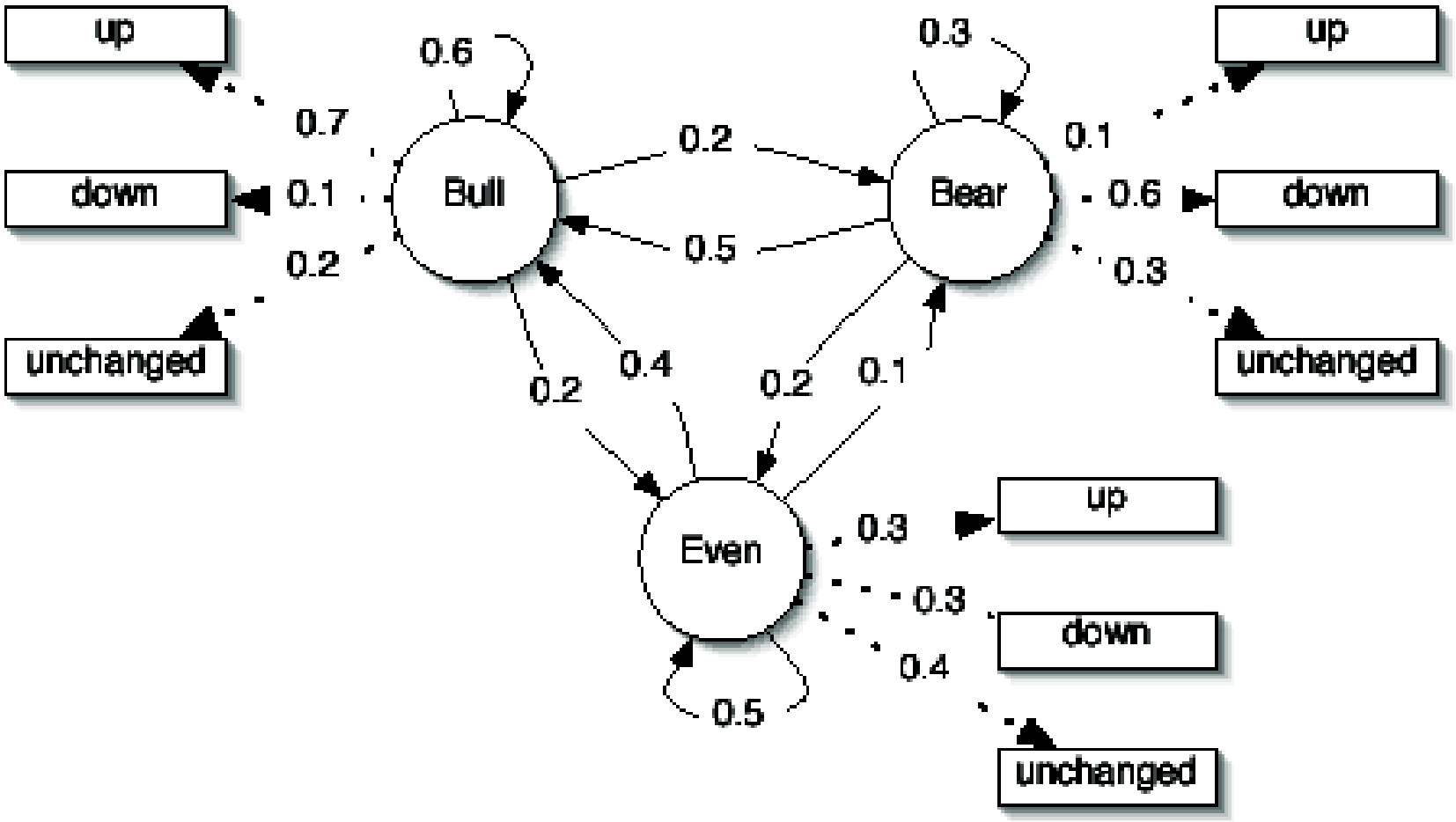
Markov Processes



Markov Processes (Cont.)

- The model presented describes a simple model for a stock market index.
- The model has three states: Bull, Bear and Even. It has three index observations up, down, and unchanged.
- The model is a finite state automaton, with probabilistic transitions between states. Given a sequence of observations, e.g., up-down-down, we can easily verify that the state sequence that produced those observations was: Bull-Bear-Bear, and the probability of the sequence is simply the product of the transitions, in this case $0.2 \times 0.3 \times 0.3$.

Hidden Markov Models (HMM)



HMM (Cont.)

- The new model now allows all observation symbols to be emitted from each state with a finite probability. This change makes the model much more expressive and able to better represent our intuition. For example, a bull market would have both good days and bad days, but there would be more good ones.
- The key difference is that now if we have the observation sequence up-down-down then we cannot say exactly what state sequence produced these observations and thus the state sequence is 'hidden'. We can however calculate the probability that the model produced the sequence, as well as which state sequence was most likely to have produced the observations.

Formal Definition of HMM

$$\gamma = (A, B, \pi) \qquad \text{(Eq. 1)}$$

- S is the state alphabet set, and V is the observation alphabet set:

$$S = (s_1, s_2, \cdots, s_N) \qquad \text{(Eq. 2)}$$

$$V = (v_1, v_2, \cdots, v_M) \qquad \text{(Eq. 3)}$$

- Let Q be a fixed state sequence of length T, and corresponding observations O:

$$Q = q_1, q_2, \cdots, q_T \qquad \text{(Eq. 4)}$$

$$O = o_1, o_2, \cdots, o_T \qquad \text{(Eq. 5)}$$

where $q_i \in S$ and $o_i \in V$

Formal Definition of HMM (Cont.)

- **A** is a transition array, storing the probability of state j following state i . Note the state transition probabilities are independent of time:

$$A = [a_{ij}] \ , \ a_{ij} = P(q_t = s_j \mid q_{t-1} = s_i). \qquad \text{(Eq. 6)}$$

- **B** is the observation array, storing the probability of observation k being produced from the state i, independent of t:

$$B = [b_i(k)] \ , \ b_i(k) = P(o_t = v_k \mid q_t = s_i). \qquad \text{(Eq. 7)}$$

- **π** is the initial probability array:

$$\pi = [\pi_i] \ , \ \pi_i = P(q_1 = s_i) \ . \qquad \text{(Eq. 8)}$$

Two Assumptions

- Markov assumption: It states that the current state is dependent only on the previous state, this represents the memory of the model:

$$P(q_t \mid q_1^{t-1}) = P(q_t \mid q_{t-1}) \qquad \text{(Eq. 9)}$$

- Independent assumption: It states that the output observation at time t is dependent only on the current state, it is independent of previous observations and states:

$$P(o_t \mid o_1^{t-1}, q_1^t) = P(o_t \mid q_t) \qquad \text{(Eq. 10)}$$

The Power of HMM

- An HMM can perform a number of tasks based on sequences of observations:

1) Learning: Given an observation sequence $O=\{o_1, o_2, \ldots, o_T\}$ and a model, γ, the model parameters can be adjusted such as $P(O|\gamma)$ is maximized.

2) Prediction: An HMM model, γ, can predict observation sequences and their associated state sequences in which the probabilistic characteristics of the model are inherently reflected.

The Power of HMM (Cont.)

3) Sequence Classification: For a given observation sequence $O=\{o_1, o_2, \ldots, o_T\}$ by computing $P(O|\ \gamma_i)$ for a set of known models γ_i, the sequence can be classified as belonging to class i for which $P(O|\ \gamma_i)$ is maximized.

4) Sequence Interpretation: Given $O=\{o_1, o_2, \ldots, o_T\}$ and an HMM, γ, applying the Viterbi algorithm gives a single most likely state sequence $Q = \{q_1, q_2, \cdots, q_T\}$.

HMM Example
-- Detect Shadow

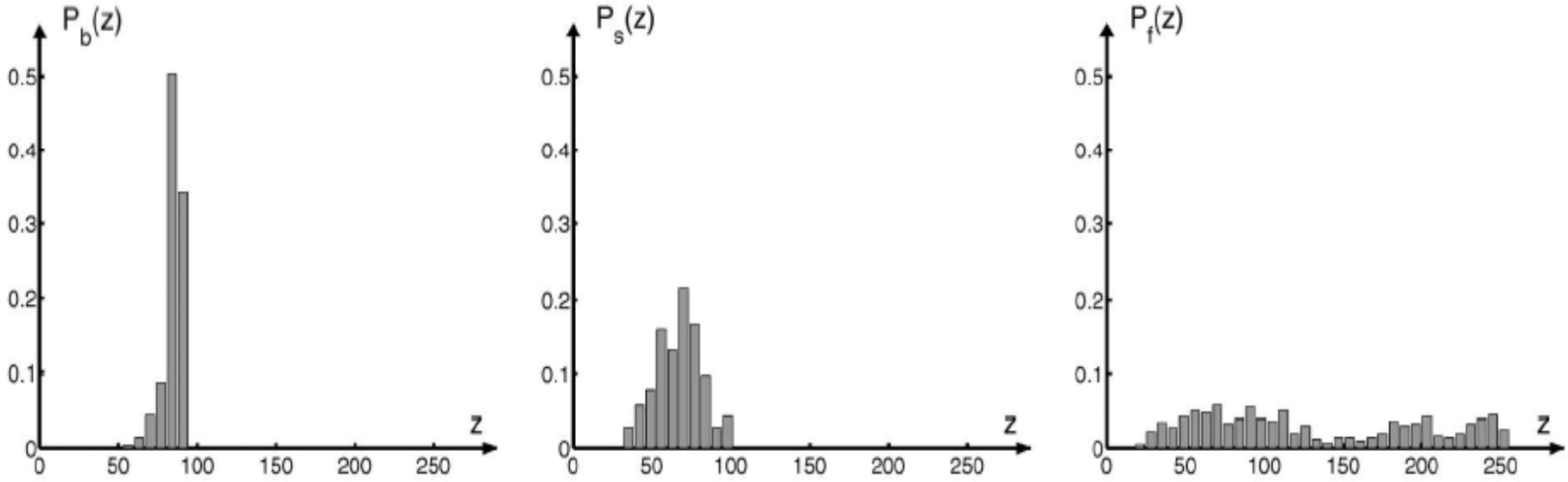
- Clearly state Background (B), Shadow (S), and Foreground (F)

- A good assumption:

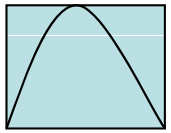
  When a pixel belongs to a group of B,S or F at current time, it is more likely that it still belong to that group at the next time.

- Each group of pixels are not totally separated in color domain.

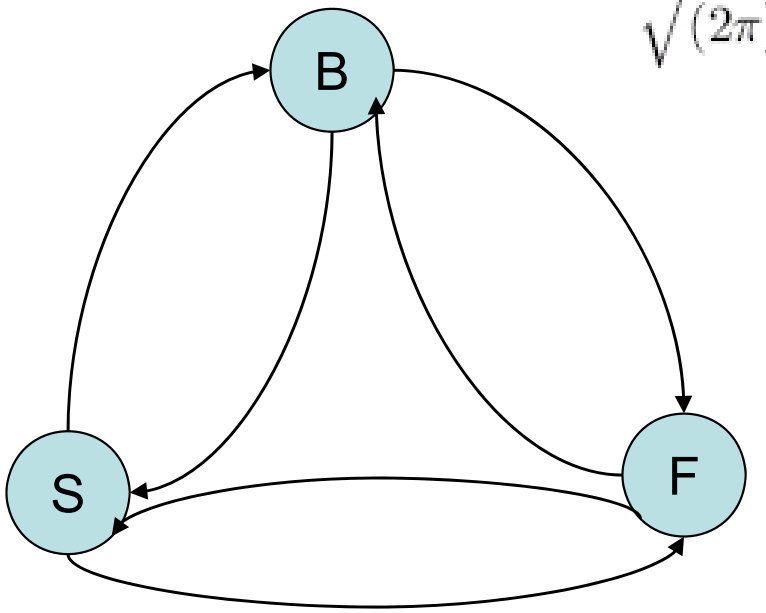
B, S, F Observation Models

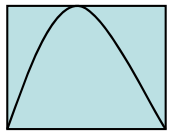


HMM Structure



$$b_i(v) = \frac{1}{\sqrt{(2\pi)^2 \det(\mathbf{\Sigma}_i)}} e^{-\frac{1}{2}(v-\mu_i)^t \mathbf{\Sigma}_i^{-1}(v-\mu_i)}, \quad i \in \{b, s\}.$$





Linear Discriminant Analysis (LDA) -- Face Identification

- Suppose a data set X exists, which might be face images, each of which is labeled with an identity. All data points with the same identity form a class. In total, there are C classes. That is:

$$X = \{X1, X2, \ldots, Xc\}$$

LDA

- The sample covariance matrix for the entire data set is then a NxN symmetric matrix

$$\sum = \frac{1}{M} \sum_{x \in X} \left[x - \mu \right] \left[x - \mu \right]^T$$

where M is the total number of faces.

- This matrix characterize the scatter of the entire data set, irrespective of class-membership.

LDA

- However, a within-classes scatter matrix, W, and a between-classes scatter matrix, B are also estimated.

$$W = \frac{1}{C}\sum_{c=1}^{C}\left\{\frac{1}{M_c}\sum_{x\in X_c}[x-\mu_c][x-\mu_c]^T\right\}$$

$$B = \frac{1}{C}\sum_{c=1}^{C}[\mu_c-\mu][\mu_c-\mu]^T$$

- Where Mc is the number of samples of class c, μ_c is the sample mean for class c, and μ is the sample mean for the entire data set X.

LDA

- The goal is to find a linear transform U which maximizes the between-class scatter while minimizing the within-class scatter.
- Such a transformation should retain class separability while reducing the variation due to sources other than identity, for example, illumination and facial expression.

LDA

- An appropriate transformation is given by the matrix U = [u1 u2 … u_K] whose columns are the eigenvectors of $W^{-1}B$.
- In other words, the generalized eigenvectors corresponding to the K largest eigenvalues

$$BU_k = \lambda_k W U_k$$

- There are at most C-1 non-zero generalized eigenvalues, so K<C

LDA

- The data are transformed as follows:

$$\alpha = U^T \left(x - \mu \right)$$

- After this transformation, the data has between-class scatter matrix $U^T B U$ and within-class scatter matrix $U^T W U$.

- The matrix U is such that the determinant of the new between-class scatter is maximized while the determinant of the within-class scatter is minimized.

- This implies that the following ratio is to be maximized:

  $| U^T B U | / | U^T W U |$

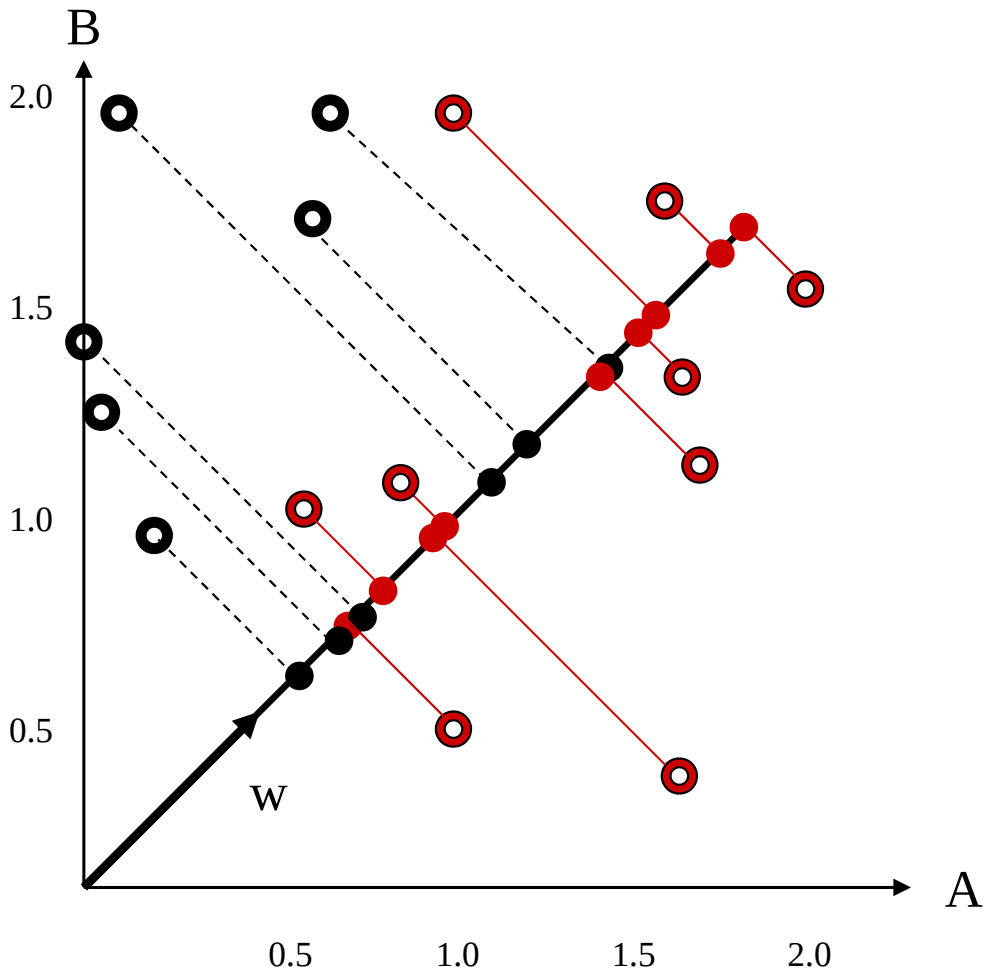
LDA

- In practice, the within-class scatter matrix (W) is often singular. This is nearly always the case when the data are image vectors with large dimensionality since the size of the data set is usually small in comparison (M<N).
- For this reason, PCA is first applied to the data set to reduce its dimensionality. The discriminant transformation is then applied to further reduce the dimensionality to C-1.

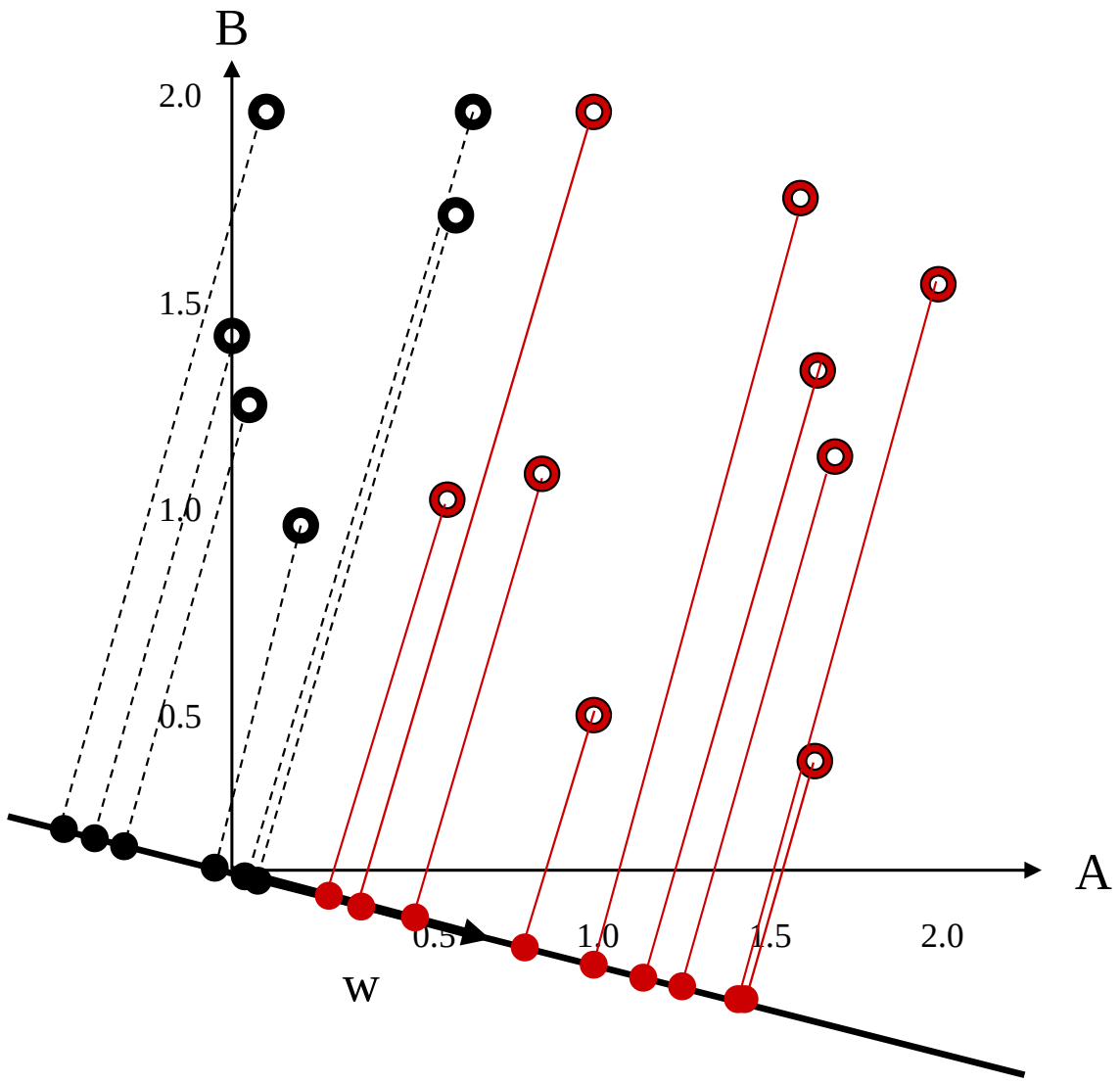
PCA vs. LDA

- PCA seeks directions that are efficient for representation;
- LDA seeks directions that are efficient for discrimination.
- To obtain good separation of the projected data, we really want the difference between the means to be large relative to some measure of the standard deviation for each class.

LDA Illustration
-- Bad Separation



LDA Illustration
-- Good Separation



LDA Illustration
-- 2-class case