Recap

- We consider PR as a two step process – Feature measurement/extraction and Classification

Recap

- We consider PR as a two step process – Feature measurement/extraction and Classification
- A classifier is to map feature vectors to Class labels.

Recap

- We consider PR as a two step process – Feature measurement/extraction and Classification
- A classifier is to map feature vectors to Class labels.
- Function learning is a closely related problem.

Recap

- We consider PR as a two step process – Feature measurement/extraction and Classification
- A classifier is to map feature vectors to Class labels.
- Function learning is a closely related problem.
- The main information we have for the design is a training set of examples.

Recap

- We consider PR as a two step process – Feature measurement/extraction and Classification
- A classifier is to map feature vectors to Class labels.
- Function learning is a closely related problem.
- The main information we have for the design is a training set of examples.
- In both cases we need to learn from (training) examples.

Recap

- In statistical pattern recognition, we model variations of feature values through probability distributions.

Recap

- In statistical pattern recognition, we model variations of feature values through probability distributions.
- The statistical viewpoint gives us one way of looking for ‘optimal’ classifier.

Recap

- In statistical pattern recognition, we model variations of feature values through probability distributions.
- The statistical viewpoint gives us one way of looking for ‘optimal’ classifier.
- We saw Bayes classifier – put the pattern into the class with highest posterior probability.

Recap

- In statistical pattern recognition, we model variations of feature values through probability distributions.
- The statistical viewpoint gives us one way of looking for ‘optimal’ classifier.
- We saw Bayes classifier – put the pattern into the class with highest posterior probability.
- The Bayes classifier is optimal in the sense of minimizing probability of misclassification.

Recall notation

- $\mathcal{X}$ – feature space. Usually $\Re^n$.
- $\mathcal{Y}$ – set of class labels.
- $X = (x_1, \cdots x_n)^T$ – feature vector.
- A classifier is a function

$$h : \mathcal{X} \rightarrow \mathcal{Y} \, (= \{0, \, 1\})$$

A classifier maps feature vectors to class labels.

- f_0, f_1 – class conditional densities (over $\mathcal{X}$)
- p_i= Prob$[y(X) = i]$, $i = 0, 1$ – prior probabilities.
- $q_i(X) =$ Prob$[y(X) = i|X]$, $i = 0, 1.$ – posterior probabilities.

Recall notation

- Bayes Theorem:

$$f_{Y|X}(y|x) = \frac{f_{X|Y}(x|y) f_Y(y)}{f_X(x)}$$

- By Bayes theorem:

$$q_i(X) = \frac{f_i(X) p_i}{Z}$$

where $Z = f_0(X)p_0 + f_1(X)p_1$ is the normalising constant.

Bayes Classifier

- The Bayes classifier

$$\begin{aligned} h_B(X) &= 0 \text{ if } q_0(X) > q_1(X) \\ &= 1 \text{ otherwise} \end{aligned}$$

Bayes Classifier

- The Bayes classifier

$$\begin{aligned} h_B(X) &= 0 \text{ if } q_0(X) > q_1(X) \\ &= 1 \text{ otherwise} \end{aligned}$$

- $q_0(X) > q_1(X)$ is same as $p_0 f_0(X) > p_1 f_1(X)$.

Bayes Classifier

- The Bayes classifier

$$\begin{aligned} h_B(X) &= 0 \text{ if } q_0(X) > q_1(X) \\ &= 1 \text{ otherwise} \end{aligned}$$

- $q_0(X) > q_1(X)$ is same as $p_0 f_0(X) > p_1 f_1(X)$.
- Minimizes probability of error in classification.

Optimality

- Each classifier h maps $\mathcal{X}$ to $\{0, 1\}$.

Optimality

- Each classifier h maps $\mathcal{X}$ to $\{0, 1\}$.
- For any classifier h, let
  $R_i(h) = \{X \in \mathcal{X} \ : \ h(X) = i\}, i = 0, 1.$
- That is, $R_0(h)$ is the set of all feature vectors that get classified as Class-0 by the classifier h.

Optimality

- Each classifier h maps $\mathcal{X}$ to $\{0, 1\}$.
- For any classifier h, let
  $R_i(h) = \{X \in \mathcal{X} \ : \ h(X) = i\}, i = 0, 1.$
- That is, $R_0(h)$ is the set of all feature vectors that get classified as Class-0 by the classifier h.
- Let $F(h)$ denote probability of error for h as defined earlier.

Optimality

$$F(h) \;=\; \mathsf{P}[X \in R_1(h), X \in \text{C-0}] + \mathsf{P}[X \in R_0(h), X \in \text{C-1}]$$

Optimality

$$
\begin{aligned}
F(h) &= \mathsf{P}[X \in R_1(h), X \in \text{C-0}] + \mathsf{P}[X \in R_0(h), X \in \text{C-1}] \\
&= p_0 \mathsf{P}[X \in R_1(h) | X \in \text{C-0}] + \\
&\qquad p_1 \mathsf{P}[X \in R_0(h) | X \in \text{C-1}]
\end{aligned}
$$

Optimality

$$
\begin{aligned}
F(h) &= \mathsf{P}[X \in R_1(h), X \in \text{C-0}] + \mathsf{P}[X \in R_0(h), X \in \text{C-1}] \\
&= p_0 \mathsf{P}[X \in R_1(h) | X \in \text{C-0}] + \\
&\qquad p_1 \mathsf{P}[X \in R_0(h) | X \in \text{C-1}] \\
&= p_0 \int_{R_1(h)} f_0(X) dX + p_1 \int_{R_0(h)} f_1(X) dX
\end{aligned}
$$

Optimality

$$
\begin{aligned}
F(h) &= \mathsf{P}[X \in R_1(h), X \in \text{C-0}] + \mathsf{P}[X \in R_0(h), X \in \text{C-1}] \\
&= p_0 \mathsf{P}[X \in R_1(h) | X \in \text{C-0}] + \\
&\qquad p_1 \mathsf{P}[X \in R_0(h) | X \in \text{C-1}] \\
&= p_0 \int_{R_1(h)} f_0(X) dX + p_1 \int_{R_0(h)} f_1(X) dX
\end{aligned}
$$

- Note that for each X we add either $p_0 f_0(X)$ or $p_1 f_1(X)$ in calculating the error integral.

Optimality

- For Bayes Classifier,
  $R_0(h_B) = \{X \ : \ p_0 f_0(X) > p_1 f_1(X)\}$ and
  $R_1(h_B) = \{X \ : \ p_1 f_1(X) \geq p_0 f_0(X)\}$.

Optimality

- For Bayes Classifier,
  $R_0(h_B) = \{X \ : \ p_0 f_0(X) > p_1 f_1(X)\}$ and
  $R_1(h_B) = \{X \ : \ p_1 f_1(X) \geq p_0 f_0(X)\}$. Then

$$F(h_B) \ = \ \int_{R_1(h_B)} p_0 f_0(X) dX + \int_{R_0(h_B)} p_1 f_1(X) dX$$

Optimality

- For Bayes Classifier,
  $R_0(h_B) = \{X \ : \ p_0 f_0(X) > p_1 f_1(X)\}$ and
  $R_1(h_B) = \{X \ : \ p_1 f_1(X) \geq p_0 f_0(X)\}$. Then

$$
\begin{aligned}
F(h_B) &= \int_{R_1(h_B)} p_0 f_0(X) dX + \int_{R_0(h_B)} p_1 f_1(X) dX \\
&= \int_{\mathcal{X}} \min \left(p_0 f_0(X), p_1 f_1(X)\right) dX
\end{aligned}
$$

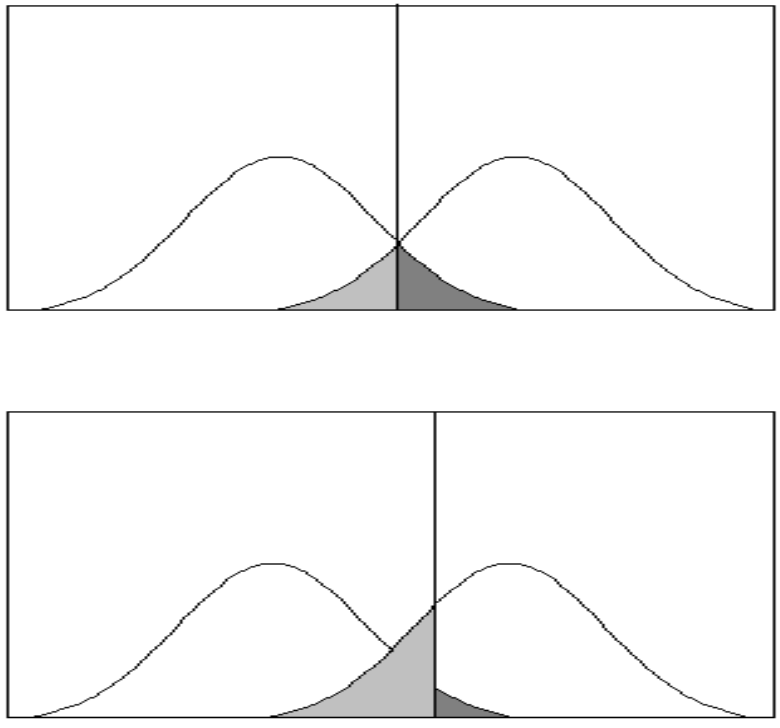
Optimality

- For Bayes Classifier,
  $R_0(h_B) = \{X \ : \ p_0 f_0(X) > p_1 f_1(X)\}$ and
  $R_1(h_B) = \{X \ : \ p_1 f_1(X) \geq p_0 f_0(X)\}$. Then

$$
\begin{aligned}
F(h_B) &= \int_{R_1(h_B)} p_0 f_0(X) dX + \int_{R_0(h_B)} p_1 f_1(X) dX \\
&= \int_{\mathcal{X}} \min \left(p_0 f_0(X), p_1 f_1(X)\right) dX
\end{aligned}
$$

- Hence Bayes classifier is optimal. (That is, given the knowledge of the probability densities, no other classifier performs better).

Optimality



Bayes Classifier contd.

- How does one implement Bayes Classifier?

Bayes Classifier contd.

- How does one implement Bayes Classifier?
- We need posterior probabilities, $q_i(X)$. Not generally available.

Bayes Classifier contd.

- How does one implement Bayes Classifier?
- We need posterior probabilities, $q_i(X)$. Not generally available.
- It is enough if we can get $f_i(X)$ and p_i, i=0,1. These can be estimated from the training set of examples.

Bayes Classifier contd.

- How does one implement Bayes Classifier?
- We need posterior probabilities, $q_i(X)$. Not generally available.
- It is enough if we can get $f_i(X)$ and p_i, i=0,1. These can be estimated from the training set of examples.
- There are different techniques for estimating the class conditional densities.

Bayes Classifier contd.

- How does one implement Bayes Classifier?
- We need posterior probabilities, $q_i(X)$. Not generally available.
- It is enough if we can get $f_i(X)$ and p_i, i=0,1. These can be estimated from the training set of examples.
- There are different techniques for estimating the class conditional densities.
- We can implement a Bayes Classifier with the estimated quantities.

Bayes Classifier (Contd.)

- Bayes classifier minimizes probability of error (misclassification).

Bayes Classifier (Contd.)

- Bayes classifier minimizes probability of error (misclassification).
- There are two kinds of errors in classification.

Bayes Classifier (Contd.)

- Bayes classifier minimizes probability of error (misclassification).
- There are two kinds of errors in classification.
- Classifying – C-0 as C-1 or C-1 as C-0
- False Positive or False negative; Type-I or Type-II; False Alarm or Missed detection

Bayes Classifier (Contd.)

- Bayes classifier minimizes probability of error (misclassification).
- There are two kinds of errors in classification.
- Classifying – C-0 as C-1 or C-1 as C-0
- False Positive or False negative; Type-I or Type-II; False Alarm or Missed detection
- The ‘costs’ for these errors may be different.
- We may want to trade one type of errors with the other type

Statistical PR contd.

- A more general way to assign figure of merit is to use a **loss function**, $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re^{+}$.

Statistical PR contd.

- A more general way to assign figure of merit is to use a **loss function**, $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re^{+}$.
- The idea is that $L(h(X), y(X))$ denotes the loss suffered by h on a pattern X.

Statistical PR contd.

- A more general way to assign figure of merit is to use a **loss function**, $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re^{+}$.
- The idea is that $L(h(X), y(X))$ denotes the loss suffered by h on a pattern X.
- Now we can define

$$F(h) = E[L(h(X), y(X))]$$

The $F(h)$ is called **risk** of h.

Loss functions

- The 0-1 loss fn:

$$\begin{aligned} L(a,b) &= 0 \text{ if } a = b \\ &= 1 \text{ otherwise.} \end{aligned}$$

Loss functions

- The 0-1 loss fn:

$$\begin{aligned} L(a,b) &= 0 \text{ if } a = b \\ &= 1 \text{ otherwise.} \end{aligned}$$

Now
$F(h) = E[L(h(X), y(X))] = \text{Prob}[h(X) \neq y(X)]$.
(Same as before)

Loss functions

- A more general loss function is to have $L(0,1) \neq L(1,0)$. ($L(0,0) = L(1,1) = 0$).

Loss functions

- A more general loss function is to have $L(0,1) \neq L(1,0)$. ($L(0,0) = L(1,1) = 0$).
- Now $F(h)$ is the expected cost of misclassification.

Loss functions

- A more general loss function is to have $L(0,1) \neq L(1,0)$. ($L(0,0) = L(1,1) = 0$).
- Now $F(h)$ is the expected cost of misclassification.
- The relative values of $L(0,1)$ and $L(1,0)$ determine how we trade errors.

Bayes Classifier to minimize risk

- Bayes classifier under the more general loss function is given by

$$\begin{aligned} h_B(X) &= 0 \text{ if } \frac{q_0(X)}{q_1(X)} > \frac{L(0,1)}{L(1,0)} \\ &= 1 \text{ otherwise} \end{aligned}$$

Bayes Classifier to minimize risk

- Bayes classifier under the more general loss function is given by

$$\begin{aligned} h_B(X) &= 0 \text{ if } \frac{q_0(X)}{q_1(X)} > \frac{L(0,1)}{L(1,0)} \\ &= 1 \text{ otherwise} \end{aligned}$$

- If $L(0,1) = L(1,0)$ – same as the earlier one

Bayes Classifier to minimize risk

- Bayes classifier under the more general loss function is given by

$$\begin{aligned} h_B(X) &= 0 \text{ if } \frac{q_0(X)}{q_1(X)} > \frac{L(0,1)}{L(1,0)} \\ &= 1 \text{ otherwise} \end{aligned}$$

- If $L(0,1) = L(1,0)$ – same as the earlier one
- If $L(0,1) = 3L(1,0)$ then we want $q_0(X) > 3q_1(X)$ to say Class-0.
- Can be shown to be optimal as before.

- Bayes classifier needs knowledge of class conditional densities and prior probabilities. These need to be estimated using the training set.
- This can be computationally expensive or may not be always feasible.
- There are other methods for obtaining classifiers.

Nearest Neighbour (NN) Classifier (Rule)

- A simple classifier that often performs very well.

Nearest Neighbour (NN) Classifier (Rule)

- A simple classifier that often performs very well.
- We store some feature vectors from the training set as *prototypes*. (Can be the whole training set!). Note that we know the (correct) class label for each prototype.

Nearest Neighbour (NN) Classifier (Rule)

- A simple classifier that often performs very well.
- We store some feature vectors from the training set as *prototypes*. (Can be the whole training set!). Note that we know the (correct) class label for each prototype.
- Given a new pattern (feature vector) X we find the prototype X' that is closest to X. Then classify X into the same class as X'.

Nearest Neighbour Classifier contd.

- A variation: k-NN rule.
  Find the k prototypes closest to X. Classify X into the majority class of these prototypes.

Nearest Neighbour Classifier contd.

- A variation: k-NN rule.
  Find the k prototypes closest to X. Classify X into the majority class of these prototypes.
- There are two main issues in designing an NN classifier.
  - Selection of Prototypes
  - Distance between feature vectors

Nearest Neighbour Classifier contd.

- A variation: k-NN rule.
  Find the k prototypes closest to X. Classify X into the majority class of these prototypes.
- There are two main issues in designing an NN classifier.
  - Selection of Prototypes
  - Distance between feature vectors
- A very simple classifier to design and operate.
- Time and memory needs depend on number of prototypes and complexity of distance function.

Nearest Neighbour Classifier contd.

- Selection of Prototypes: How many? How to select?

Nearest Neighbour Classifier contd.

- Selection of Prototypes: How many? How to select?
- Distance function: Can use Euclidean distance.
  $d(X, X') = (\sum (x_i - x'_i)^2)^{0.5}$

Nearest Neighbour Classifier contd.

- Selection of Prototypes: How many? How to select?
- Distance function: Can use Euclidean distance.
  $d(X, X') = (\sum (x_i - x'_i)^2)^{0.5}$
- A better method may be
  $d(X, X') = \left(\sum (\frac{x_i - x'_i}{\sigma_i})^2 \right)^{0.5}$
  Here σ_i^2 is the (estimated) variance of i^{th} feature.

Nearest Neighbour Classifier contd.

- Selection of Prototypes: How many? How to select?
- Distance function: Can use Euclidean distance.
  $d(X, X') = (\sum (x_i - x'_i)^2)^{0.5}$
- A better method may be
  $d(X, X') = \left(\sum (\frac{x_i - x'_i}{\sigma_i})^2 \right)^{0.5}$
  Here σ_i^2 is the (estimated) variance of i^{th} feature.
- A more general form is:

$$d(X, X') = \left((X - X')^T \Sigma^{-1} (X - X') \right)^{0.5}$$

where Σ is the (estimated) covariance matrix. Called Mahalanobis distance.

Nearest Neighbour Classifier

- The NN rule does not really use any statistical viewpoint.

Nearest Neighbour Classifier

- The NN rule does not really use any statistical viewpoint.
- If we have a sequence of *iid* examples, asymptotically, the probability of error by NN rule is never more than twice the Bayes error.

Nearest Neighbour Classifier

- The NN rule does not really use any statistical viewpoint.
- If we have a sequence of *iid* examples, asymptotically, the probability of error by NN rule is never more than twice the Bayes error.
- The NN rule is also related to certain non-parametric methods of estimating class conditional densities

Another approach: Discriminant functions

- Consider the following structure for classifier

$$\begin{aligned} h(X) &= 1 \text{ if } g(X) > 0 \\ &= 0 \text{ Otherwise} \end{aligned}$$

g is called a discriminant function.

Another approach: Discriminant functions

- Consider the following structure for classifier

$$\begin{aligned} h(X) &= 1 \text{ if } g(X) > 0 \\ &= 0 \text{ Otherwise} \end{aligned}$$

  g is called a discriminant function.

- If we choose $g(X) = q_1(X) - q_0(X)$, then this is the Bayes classifier.

Another approach: Discriminant functions

- Consider the following structure for classifier

$$\begin{aligned} h(X) &= 1 \text{ if } g(X) > 0 \\ &= 0 \text{ Otherwise} \end{aligned}$$

  g is called a discriminant function.

- If we choose $g(X) = q_1(X) - q_0(X)$, then this is the Bayes classifier.
- Instead of assuming functional form for class conditional densities, we can assume a functional form for g and learn the needed function.

Linear Discriminant functions

- Suppose g is specified by a parameter vector $W \in \Re^N$. We write $g(W, X)$ for $g(X)$ now.

Linear Discriminant functions

- Suppose g is specified by a parameter vector $W \in \Re^N$. We write $g(W, X)$ for $g(X)$ now.
- For example, $N = n + 1$ and

$$g(W, X) = \sum_{i=1}^{n} w_i x_i + w_0$$

Linear Discriminant functions

- Suppose g is specified by a parameter vector $W \in \Re^N$. We write $g(W, X)$ for $g(X)$ now.
- For example, $N = n + 1$ and

$$g(W, X) = \sum_{i=1}^{n} w_i x_i + w_0$$

$W = (w_0, \ldots, w_n)^T \in \Re^N$ is the parameter vector, and $X = (x_1, \ldots, x_n)^T \in \Re^n$ is the feature vector.

Linear Discriminant functions

- Suppose g is specified by a parameter vector $W \in \Re^N$. We write $g(W, X)$ for $g(X)$ now.
- For example, $N = n + 1$ and

$$g(W, X) = \sum_{i=1}^{n} w_i x_i + w_0$$

$W = (w_0, \ldots, w_n)^T \in \Re^N$ is the parameter vector, and $X = (x_1, \ldots, x_n)^T \in \Re^n$ is the feature vector.

- Known as **linear discriminant function**.

- A linear discriminant function based classifier is

$$\begin{aligned} h(X) &= 1 \text{ if } \sum w_i x_i + w_0 > 0 \\ &= 0 \text{ otherwise} \end{aligned}$$

- A linear discriminant function based classifier is

$$\begin{aligned} h(X) &= 1 \text{ if } \sum w_i x_i + w_0 > 0 \\ &= 0 \text{ otherwise} \end{aligned}$$

- Let us take $X = (1\ x_1\ x_2\ \cdots x_n)^T$. Called the augumented feature vector.
- Recall $W = (w_0\ w_1\ \cdots w_n)^T$.

- A linear discriminant function based classifier is

$$\begin{aligned} h(X) &= 1 \text{ if } \sum w_i x_i + w_0 > 0 \\ &= 0 \text{ otherwise} \end{aligned}$$

- Let us take $X = (1\ x_1\ x_2\ \cdots x_n)^T$. Called the augumented feature vector.
- Recall $W = (w_0\ w_1\ \cdots w_n)^T$.
- Now the classifier is: $h(X) = \text{sgn}(W^T X)$.
- One of the earliest classifiers considered (called Perceptron).

Linear discriminant functions contd.

- The training set, $\{(X_i, y_i), i = 1, \cdots, \ell\}$, of patterns is said to be **linearly separable** if there exists W^* such that

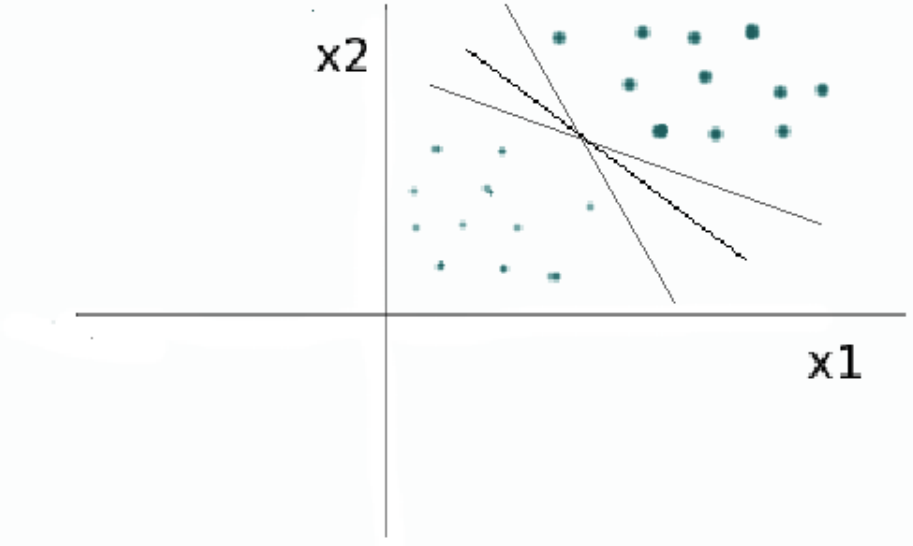
$$\begin{aligned} X_i^T W^* &> 0 \text{ if } y_i = 1 \\ &< 0 \text{ if } y_i = 0 \end{aligned}$$

Linear discriminant functions contd.

- The training set, $\{(X_i, y_i), i = 1, \cdots, \ell\}$, of patterns is said to be **linearly separable** if there exists W^* such that

$$\begin{aligned} X_i^T W^* &> 0 \text{ if } y_i = 1 \\ &< 0 \text{ if } y_i = 0 \end{aligned}$$

- Any W^* that satisfies the above is called (or rather, corresponds to) a separating hyperplane. (There exist infinitely many separating hyperplanes)


x2
x1

Learning linear discriminant functions

- We need to learn ‘optimal’ W from the training samples.

Learning linear discriminant functions

- We need to learn ‘optimal’ W from the training samples.
- Perceptron learning algorithm is one of the earliest algorithms for learning linear discriminant functions.
- Finds a separating hyperplane if the training set is linearly separable.

Learning linear discriminant functions

- We need to learn ‘optimal’ W from the training samples.
- Perceptron learning algorithm is one of the earliest algorithms for learning linear discriminant functions.
- Finds a separating hyperplane if the training set is linearly separable.
- We can also have a risk-minimization approach to learning discriminant functions.

Learning discriminant functions

- The question is: How to ‘evaluate’ different W?

Learning discriminant functions

- The question is: How to ‘evaluate’ different W?
- We can use the old function, F.

$$F(W) = E[L(h(W, X), y(X))]$$

Learning discriminant functions

- The question is: How to ‘evaluate’ different W?
- We can use the old function, F.

$$F(W) = E[L(h(W, X), y(X))]$$

- We still have the problem of how to minimize F.
- Given a W, we can not calculate $F(W)$ unless we know the underlying probability distributions.

Learning discriminant functions

- The question is: How to ‘evaluate’ different W?
- We can use the old function, F.

$$F(W) = E[L(h(W, X), y(X))]$$

- We still have the problem of how to minimize F.
- Given a W, we can not calculate $F(W)$ unless we know the underlying probability distributions.
- But we can approximate expectation by ‘sample average’.

- Consider the function $\hat{F}$ defined by

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

where $\{(X_i, y_i), i = 1, \cdots, \ell\}$ is the training set of examples.

- Consider the function $\hat{F}$ defined by

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

  where $\{(X_i, y_i), i = 1, \cdots, \ell\}$ is the training set of examples.

- Then $\hat{F}(W)$ is a good approximation to $F(W)$. (Law of large numbers; assume examples are *iid*)

- Consider the function $\hat{F}$ defined by

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

  where $\{(X_i, y_i), i = 1, \cdots, \ell\}$ is the training set of examples.

- Then $\hat{F}(W)$ is a good approximation to $F(W)$. (Law of large numbers; assume examples are *iid*)

- So, we can minimize $\hat{F}$ instead.

Learning discriminant functions

- $\hat{F}$ measures the 'error' of classifier $h(W, \cdot)$ on the training samples.
- F measures error on the full population.

Learning discriminant functions

- $\hat{F}$ measures the 'error' of classifier $h(W, \cdot)$ on the training samples.
- F measures error on the full population.
- But F cannot be calculated because we do not know the underlying probability distributions. ($\hat{F}$ can be claculated)
- If we have 'sufficient' number of 'representative' training samples then minimizer of $\hat{F}$ would be 'good enough'.

Learning discriminant functions

- $\hat{F}$ measures the 'error' of classifier $h(W, \cdot)$ on the training samples.
- F measures error on the full population.
- But F cannot be calculated because we do not know the underlying probability distributions. ($\hat{F}$ can be calculated)
- If we have 'sufficient' number of 'representative' training samples then minimizer of $\hat{F}$ would be good enough.
- Would discuss these issues later. (Statistical learning theory)

Learning discriminant functions contd.

- Recall

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

- How to find W that minimizes $\hat{F}(W)$?

Learning discriminant functions contd.

- Recall

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

- How to find W that minimizes $\hat{F}(W)$?
- In special cases, if classes are separable, we may be able to find W^* with $\hat{F}(W^*) = 0$.

Learning discriminant functions contd.

- Recall

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

- How to find W that minimizes $\hat{F}(W)$?
- In special cases, if classes are separable, we may be able to find W^* with $\hat{F}(W^*) = 0$.
- In general we may need some optimization techniques.

Learning discriminant functions contd.

- Recall

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} L(h(W, X_i), y_i)$$

  where $h(X) = \text{sgn}(W^T X)$.

- As defined, h (and hence $\hat{F}$) are discontinuous. Also, if we use 0–1 loss function, L is also discontinuous.

Learning discriminant functions contd.

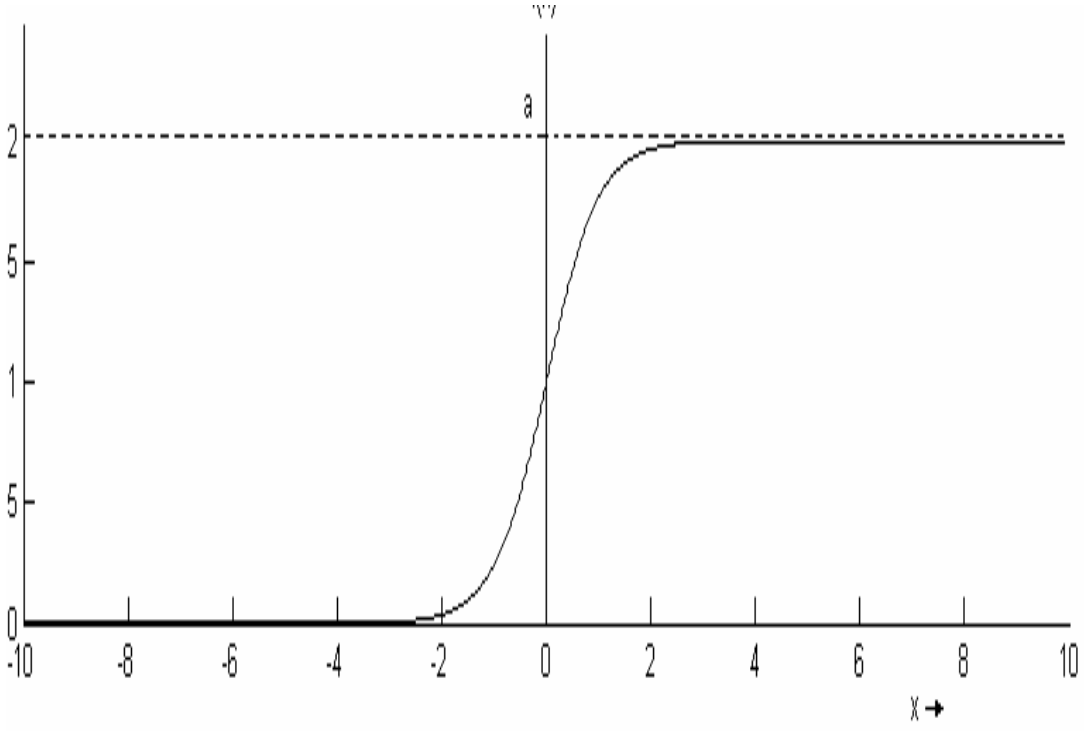
- We can redefine h so that it takes values in $[0,\ 1]$ rather than in $\{0,\ 1\}$.

Learning discriminant functions contd.

- We can redefine h so that it takes values in $[0,\ 1]$ rather than in $\{0,\ 1\}$.
- Consider the case of linear discriminant function.
- Earlier: $h(W, X) = sign(W^T X)$.
- Now we can take: $h(W, X) = \frac{1}{1+\exp(-W^T X)}$.

Learning discriminant functions contd.

- We can redefine h so that it takes values in $[0,\ 1]$ rather than in $\{0,\ 1\}$.
- Consider the case of linear discriminant function.
- Earlier: $h(W, X) = sign(W^T X)$.
- Now we can take: $h(W, X) = \frac{1}{1+\exp(-W^T X)}$.
- This is called a sigmoid function. Now h is differentiable.


a
2
5
1
5
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
X

- Redefine Loss: $L(a, b) = (a - b)^2$
  Called Squared error loss function.

- Redefine Loss: $L(a,b) = (a-b)^2$
  Called Squared error loss function.
- As before, let $F(h) = EL(h(X), y(X))$. Then,

$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell}(h(W,X) - y(X))^2$$

.

- Redefine Loss: $L(a,b) = (a-b)^2$
  Called Squared error loss function.
- As before, let $F(h) = EL(h(X), y(X))$. Then,

$$\hat{F}(W) = \frac{1}{\ell}\sum_{i=1}^{\ell}(h(W,X) - y(X))^2$$

  .
- Now we can use standard optimization techniques to minimize $\hat{F}$.
- The classical LMS algorithm is of this type.

- There are efficient algorithms for learning linear discriminant functions.
- We would be discussing some techniques for learning linear discriminant functions.

- There are efficient algorithms for learning linear discriminant functions.
- We would be discussing some techniques for learning linear discriminant functions.
- We can have the discriminant function, $g(W, X)$ to be nonlinear.

- There are efficient algorithms for learning linear discriminant functions.
- We would be discussing some techniques for learning linear discriminant functions.
- We can have the discriminant function, $g(W, X)$ to be nonlinear.
- The idea of minimizing $\hat{F}$ using a squared error loss function and a continuous h would work even if g is nonlinear.
- But we need good ways to parameterize classes of nonlinear discriminant function based classifiers.

Beyond Linear Models

- Learning linear models (classifiers) is generally efficient.

Beyond Linear Models

- Learning linear models (classifiers) is generally efficient.
- However, linear models are not always sufficient.
- Linear separability is a restrictive assumption for classifiers.
- Best linear functions may still be a poor fit.

Beyond Linear Models

- Learning linear models (classifiers) is generally efficient.
- However, linear models are not always sufficient.
- Linear separability is a restrictive assumption for classifiers.
- Best linear functions may still be a poor fit.
- How to tackle general situations?
- Here are some possible viewpoints.

Neural network idea

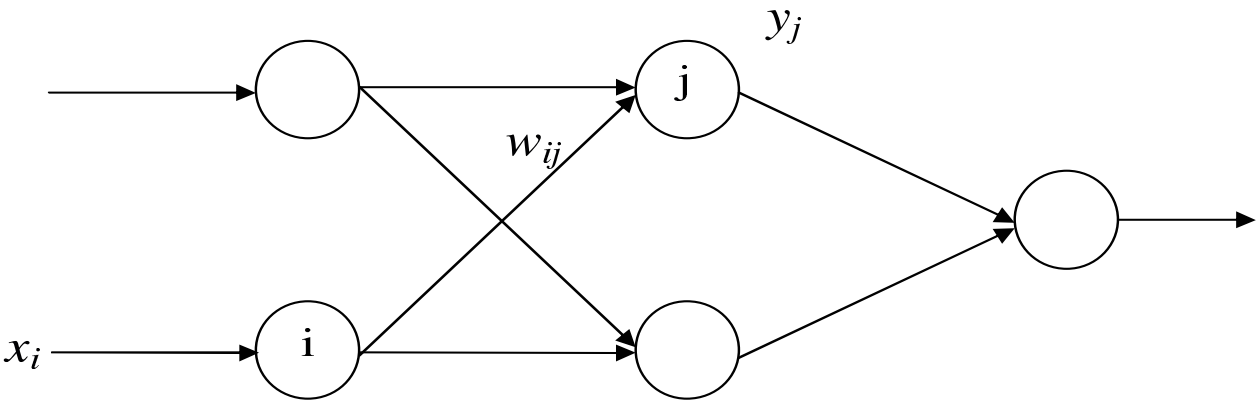
- Find a ‘good’ parameterized class of nonlinear discriminant functions

Neural network idea

- Find a ‘good’ parameterized class of nonlinear discriminant functions
- Multilayer feedforward neural nets are one such class

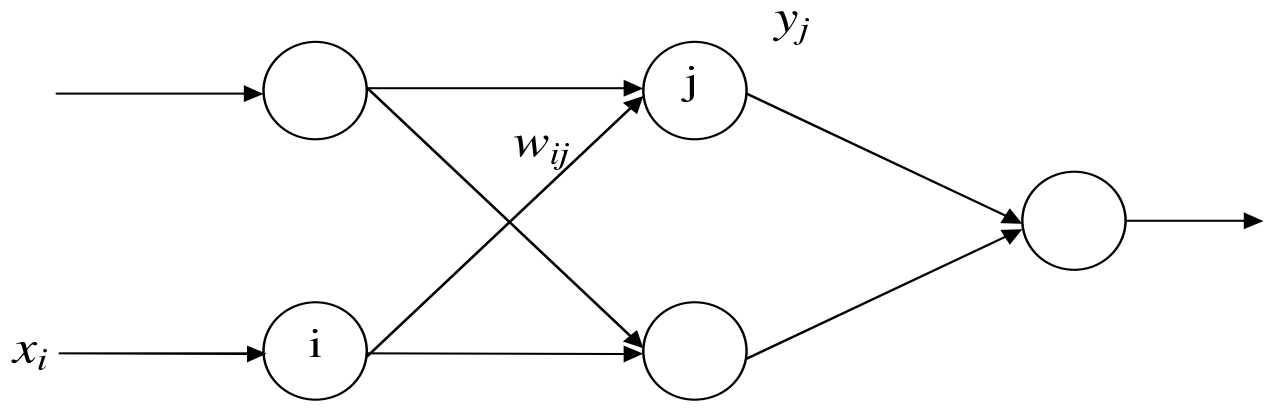
Neural network idea

- Find a ‘good’ parameterized class of nonlinear discriminant functions
- Multilayer feedforward neural nets are one such class



Neural network idea

- Find a ‘good’ parameterized class of nonlinear discriminant functions
- Multilayer feedforward neural nets are one such class



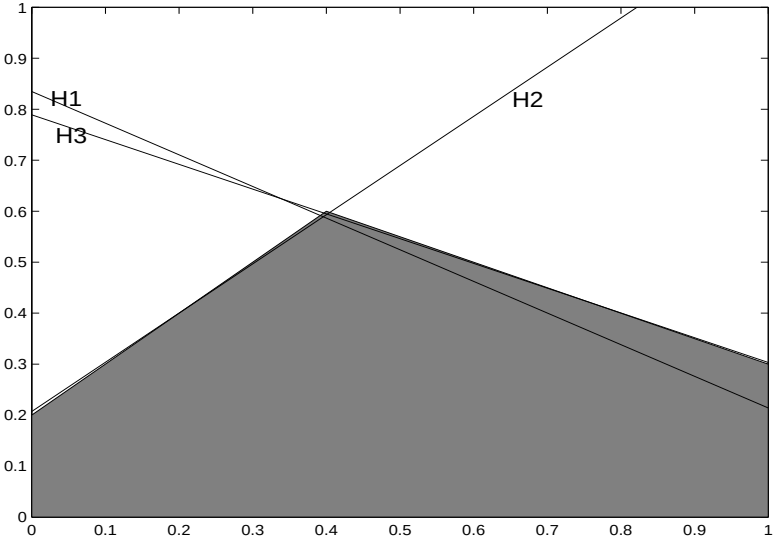
- Nonlinear functions are built up through composition of summation and sigmoids.
- Useful for both classification and Regression.

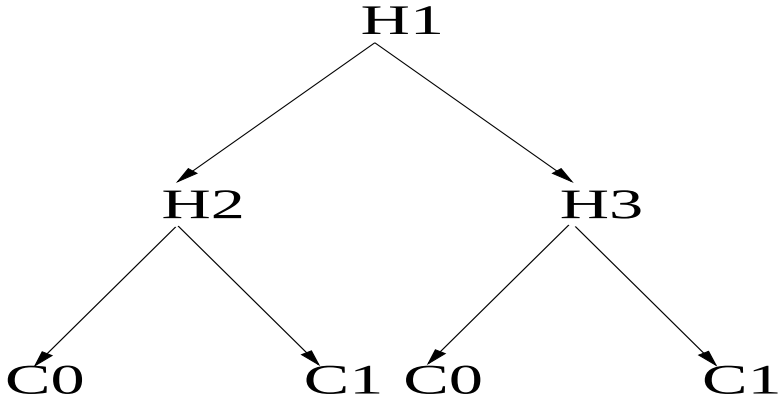
Decision Tree idea

- Divide feature space so that a linear classifier is enough in each region (e.g. Decision Trees).

Decision Tree idea

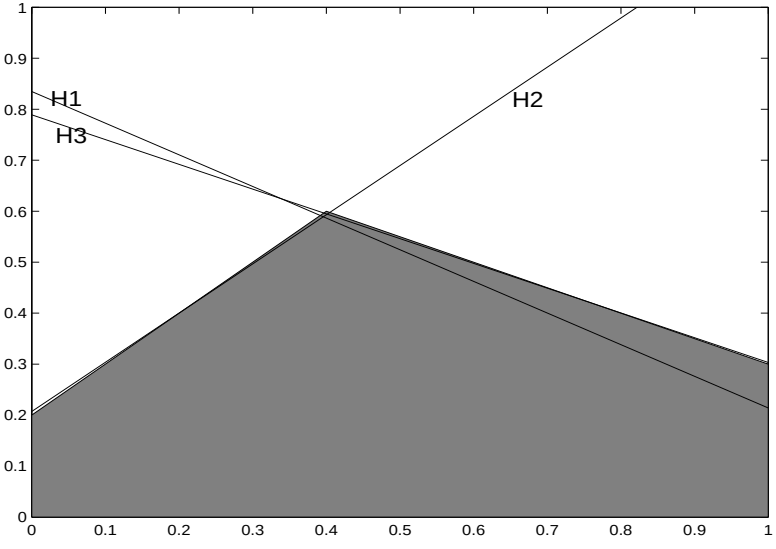
- Divide feature space so that a linear classifier is enough in each region (e.g. Decision Trees).

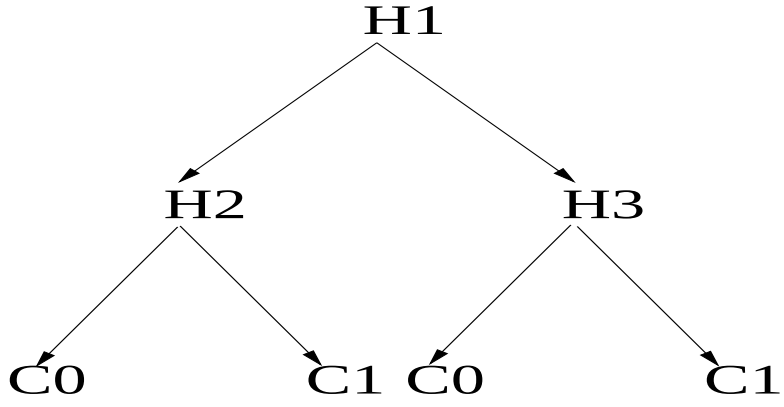




Decision Tree idea

- Divide feature space so that a linear classifier is enough in each region (e.g. Decision Trees).





- Such tree-based models are possible for regression also.

SVM idea

- Map X nonlinearly into a high dimensional space and try a linear discriminant function there. (e.g. Support Vector Machines)

SVM idea

- Map X nonlinearly into a high dimensional space and try a linear discriminant function there. (e.g. Support Vector Machines)
- Let $X = [x_1 \quad x_2]$ and let $\phi : \Re^2 \rightarrow \Re^5$ given by $Z = \phi(X) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2]$

SVM idea

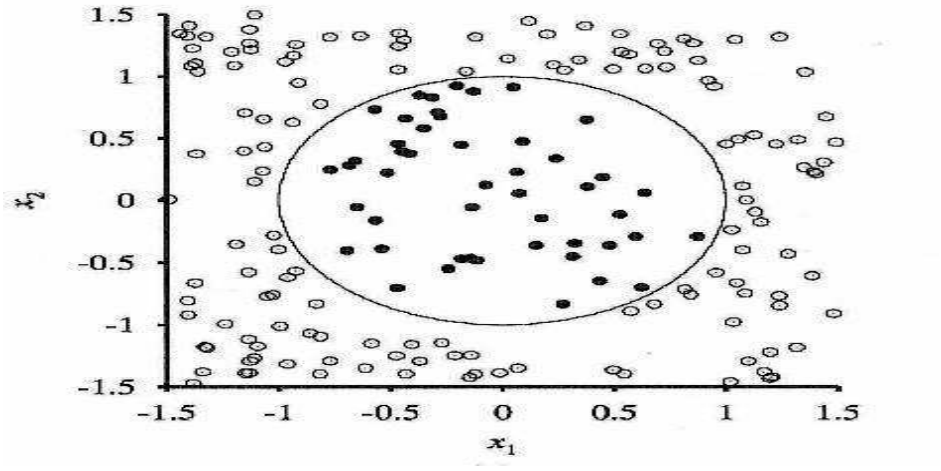
- Map X nonlinearly into a high dimensional space and try a linear discriminant function there. (e.g. Support Vector Machines)
- Let $X = [x_1 \quad x_2]$ and let $\phi : \Re^2 \rightarrow \Re^5$ given by $Z = \phi(X) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2]$
- Now,

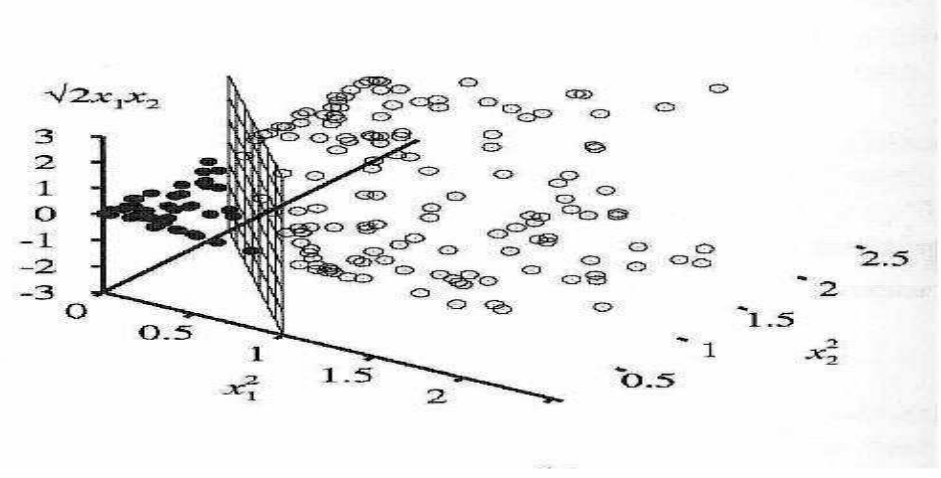
$$g(X) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2$$

is a quadratic discriminant function in $\Re^2$; but

$$g(Z) = a_0 + a_1 z_1 + a_2 z_2 + a_3 z_3 + a_4 z_4 + a_5 z_5$$

is a linear dscriminant function in the '$\phi(X)$' space.


1.5
1
0.5
0
-0.5
-1
-1.5
x_2
-1.5
-1
-0.5
0
0.5
1
1.5
x_1


$\sqrt{2}x_1x_2$
3
2
1
0
-1
-2
-3
0
0.5
1
1.5
2
x_1^2
0.5
1
1.5
2
2.5
x_2^2

Summary

- Bayes classifier minimizes risk under general loss functions. We need knowledge of class conditional densities to implement this.

Summary

- Bayes classifier minimizes risk under general loss functions. We need knowledge of class conditional densities to implement this.
- We can estimate the class conditional densities from training set of examples and implement Bayes classifier with the estimated densities.

Summary

- Bayes classifier minimizes risk under general loss functions. We need knowledge of class conditional densities to implement this.
- We can estimate the class conditional densities from training set of examples and implement Bayes classifier with the estimated densities.
- We will study some techniques for such estimation.

Summary

- Bayes classifier minimizes risk under general loss functions. We need knowledge of class conditional densities to implement this.
- We can estimate the class conditional densities from training set of examples and implement Bayes classifier with the estimated densities.
- We will study some techniques for such estimation.
- There are approaches other than Bayes Classifier.

Summary

- Nearest neighbour classifier: find a prototype that is closest to the given pattern and put the pattern in the class of the nearest neighbour.
  A popular variant is k-NN classifier.

Summary

- Nearest neighbour classifier: find a prototype that is closest to the given pattern and put the pattern in the class of the nearest neighbour.
  A popular variant is k-NN classifier.
- Nearest Neighbour method is a simple and surprisingly effective classifier.

Summary

- Nearest neighbour classifier: find a prototype that is closest to the given pattern and put the pattern in the class of the nearest neighbour.
  A popular variant is k-NN classifier.
- Nearest Neighbour method is a simple and surprisingly effective classifier.
- Another method of classifier design is based on discriminant functions.

Summary

- Nearest neighbour classifier: find a prototype that is closest to the given pattern and put the pattern in the class of the nearest neighbour.
  A popular variant is k-NN classifier.
- Nearest Neighbour method is a simple and surprisingly effective classifier.
- Another method of classifier design is based on discriminant functions.
- We have seen Linear discriminant functions as an example of this.

Summary

- One can use risk minimization approach for learning discriminant functions.

Summary

- One can use risk minimization approach for learning discriminant functions.
- Minimizing the squared error is one method of learning linear models.

Summary

- One can use risk minimization approach for learning discriminant functions.
- Minimizing the squared error is one method of learning linear models.
- We will discuss some methods of learning linear models.

Summary

- One can use risk minimization approach for learning discriminant functions.
- Minimizing the squared error is one method of learning linear models.
- We will discuss some methods of learning linear models.
- Often linear classifiers may not be able to deliver the performance needed.
- Neural Networks, SVMs are some approaches to learning nonlinear classifiers.