E1 213 Pattern Recognition and Neural Networks

P.S. Sastry
sastry@iisc.ac.in

Reference Material

- R.O.Duda, P.E.Hart and D.G.Stork, 'Pattern Classification', Johy Wiley, 2002
- C.M.Bishop, 'Pattern Recognition and Machine Learning', Springer, 2006.
- T.Hastie, R.Tibshirani and J.Friedman 'The Elements of Statistical Learning: Data Mining, Inference and Prediction', Springer, 2009.
- S Shalev-Shwartz and S. Ben-David, 'Understanding Machine Learning: From Theory to Algorithms', Cambridge University Press, 2014.
- I.Goodfellow, Y.Bengio and A. Courville, 'Deep Learning', MIT Press, 2016
- A.Zhang, Z.C.Lipton, M.Li, A.J.Smola, 'Dive into Deep Learning', 2019 (free PDF available)
- Link to my online video course on Pattern Recognition: https://nptel.ac.in/courses/117/108/117108048/

Course Prerequisites / Background needed

- Probability Theory
  - joint distributions/densities of multiple random variables, conditional distributions, expectation, conditional expectation, laws of large numbers, (convergence of seq of random variables, Markov chains)
- Matrix theory/Linear Algebra
  - vector spaces, bases, dimension, linear transformations, matrices, rank, nulity, eigen values and eigen vectors
- Optimization techniques
  - unconstrained optimization, necessary and sufficient conditions, descent methods, gradient descent, second order methods, constrained optimization, Kuhn-Tucker conditions

Course grading

- Mid-Term Tests and Assignments: 50%
  Final Exam: 50%
- Two mid-term tests and 4-5 assignments (involves programming)

Before we begin

Please ask yourself whether this is the right course for you

- This is a first-level course on ML
- This course follows a statistical (or probability-based) approach to ML.
- Course is more 'theoretical' (algorithms and analysis)
- This course does NOT teach any Python programming or use of any standard packages.
- This is not a course on deep learning. Many topics other than neural network models would be covered.

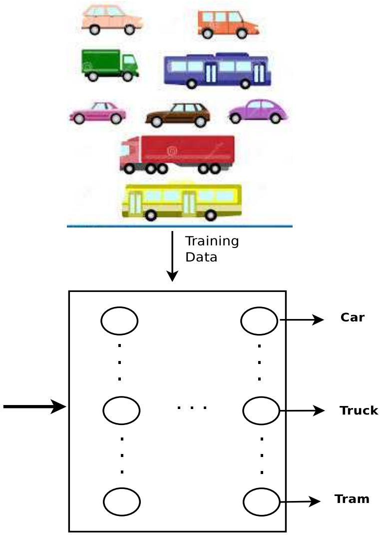
Machine Learning

- Methods for automatically detecting ‘regularities’ in data in a form that is suitable for prediction or other decision-making scenarios.
- We can think of machine learning as a principled approach to fitting models for many different kinds of data.

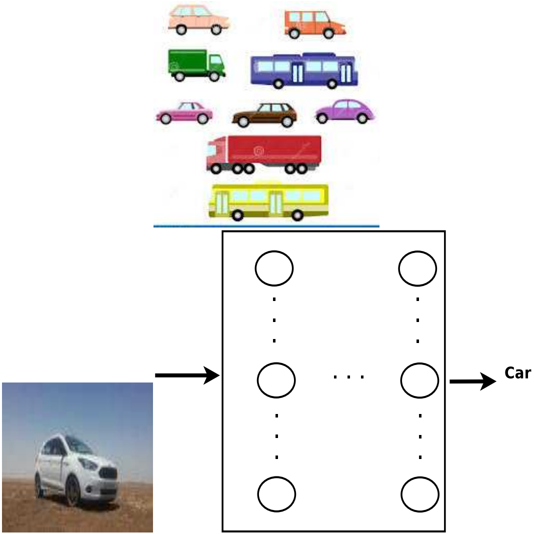
Machine learning encompasses many different data analysis scenarios.

- Predictive or Supervised Learning:
  - Classification or regression: Given training data, $\{(X_i, y_i), i = 1, 2, \cdots\}$, learn to predict an attribute (y) based on others (X).
  - Classification: $y_i \in \{C_1, \cdots, C_M\}$; Regression: $y_i \in \Re$.

Supervised learning



Supervised learning



Machine learning encompasses many different data analysis scenarios.

- Predictive or Supervised Learning:
  - Classification or regression: Given training data, $\{(X_i, y_i), i = 1, 2, \cdots\}$, learn to predict an attribute (y) based on others (X).
  - Classification: $y_i \in \{C_1, \cdots, C_M\}$; Regression: $y_i \in \Re$.
  - Supervised in the sense we know the 'correct answer' (modulo noise) for training data.
  - Variations: multi-label classification, semi-supervised learning, ordinal regression, learning under label noise etc.
- This is the most common form of ML application. ("Current AI is better called Prediction Machines")
- In this course we mainly consider supervised learning.

Reinforcement Learning

- Reinforcement Learning: ‘Learning by doing’.
  For example, how we learn to ride a bicycle.
- The feedback we get during training is only ‘evaluative’ (and noisy).
- Training data essentially consists of sample trajectories.
- Useful in many decision making and control applications.
  (AlphaGO is a reinforcement learning system)

Unsupervised Learning

- Unsupervised Learning: Analysis of ‘unlabelled’ data.
  - Clustering
  - Frequent Patterns (Market basket analysis)
  - Dimensionality reduction, latent factor analysis.
  - Topic Discovery
  - Collaborative filtering (Imputing missing values)

Pattern Recognition

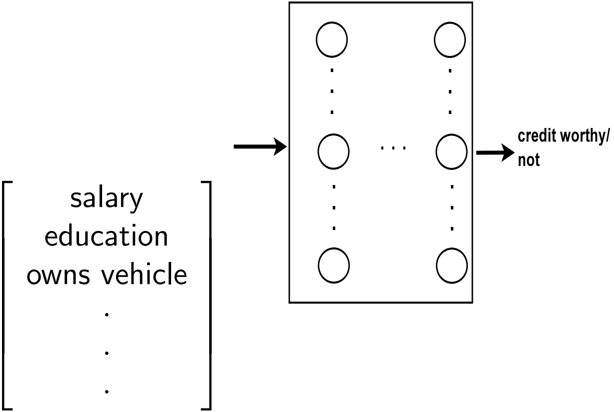
- Pattern Recognition is an older name (for supervised learning)
- A basic attribute of people – categorisation of sensory input
- Examples of (human) Pattern Recognition tasks
  - 'Reading' facial expressions
  - Recognising Speech
  - Reading a Document
  - Identifying a person by fingerprints
  - Diagnosis from medical images
  - Wine tasting

Machine Recognition of Patterns

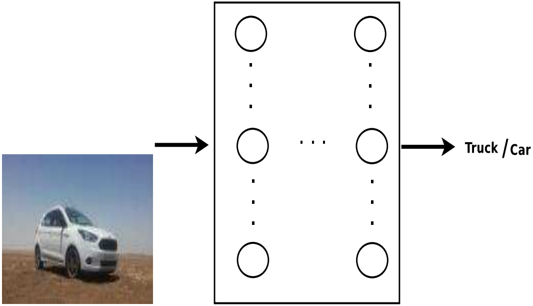
$$\text{Pattern} \rightarrow \boxed{\boxed{\text{feature extractor}} \xrightarrow{X} \boxed{\text{classifier}}} \rightarrow \text{class label}$$

- The input can be 'anything' and output is one of finitely many classes.

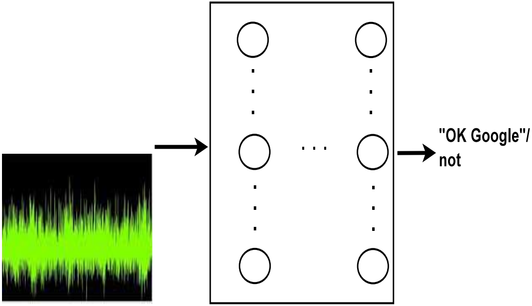
Input pattern can be a vector of measurements



Input Pattern can be an image



Input Pattern can be a 1D time signal



Machine Recognition of Patterns

$$\text{Pattern} \rightarrow \boxed{\boxed{\text{feature extractor}} \xrightarrow{X} \boxed{\text{classifier}}} \rightarrow \text{class label}$$

- Input pattern can be a vector of measurements, an image, a 1D time signal, a multi-variate time series, a video $\cdots$
- Feature extractor makes some measurements on the input pattern.
- X is called *Feature Vector*. Often, $X \in \Re^n$.
- Classifier maps each feature vector to a class label.
- Features to be used are problem-specific.

Machine Recognition of Patterns

$$\text{Pattern}\rightarrow \boxed{\boxed{\text{feature extractor}} \xrightarrow{X} \boxed{\text{classifier}}} \rightarrow\text{class label}$$

- Feature Extraction and classification may be fused together
  (For example, Neural Network models)
- We can think of pattern recognition system as mapping input pattern (can be viewed as a vector, X) to class label.

Many Examples

- Speech recognition
- Document Classification (e.g., spam detection, sentiment analysis)
- Biometrics-based authentication,
- Video Surveillance,
- Credit Screening,
- $\dots$

A Simple Pattern Recognition Problem

- Problem: ‘Spot the Right Candidate’
- Features:
  - x_1: Marks based on academic record
  - x_2: Marks in the interview
- A Classifier: $ax_1 + bx_2 > c \Rightarrow$ ‘Good’
  Another Classifier: $x_1x_2 > c \Rightarrow$ 'Good’
  (or $(x_1 + a)(x_2 + b) > c$).
- Design of classifier:
  We have to choose a specific form for the classifier.
  What values to use for parameters such as a, b, c?

Another simple PR problem

- Problem: "Sentiment Analysis"
- Pattern: A text document

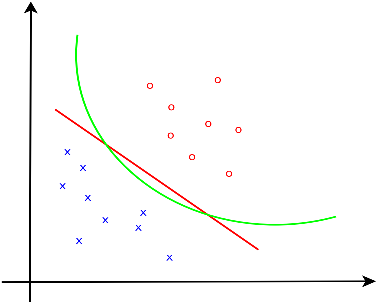
  "every now and then a movie comes along from a suspect studio with every indication that it will be a stinker and to everybody's surprise perhaps even of the studio the film becomes a critical darling ..."

- We want to classify it as, say, positive or negative (without full semantic understanding!)
- Features??
  We can use, e.g., "bag of words" representation
  (Feature vector can be very high dimensional)
- Structure for Classifier – ??

Designing Classifiers

- Need to decide how feature vector values determine the class.
  (How different marks reflect goodness of candidate)
  (How different words indicate positive sentiment)
- In most applications, not possible to design classifier from 'physics of the problem'.
- Often the only information available for the design is:
  A training set of example patterns.
- Training set: $\{(X_i, y_i), \;\; i = 1, \ldots, \ell\}$.
  Here X_i is an example feature vector of class y_i.

Training set: Spot-Right-Candidate



Regression Problems

- Closely related problem. Output is continuous-valued rather than discrete as in classifiers.
- Here training set of examples could be $\{(X_i, y_i), \;\; i = 1, \ldots, \ell\}$, $X_i \in \mathcal{X}, y_i \in \Re$.
- The prediction variable, y, is continuous; rather than taking finitely many values.
- Similar learning techniques needed to infer the underlying functional relationship between X and y. (Regression function of y on X).

Examples of regression problems

- Time series prediction: Given a series $x_1, x_2, \cdots$, find a function to predict x_n.
- Based on past values: Find a 'best' function $\hat{x}_n = h(x_{n-1}, x_{n-2}, \cdots, x_{n-p})$
  - Predict stock prices, exchange rates etc.
  - Linear prediction model used in speech analysis
  - Equalisers in communication systems
- More general predictors can use other variables also.
  - Predict rainfall based on measurements and (possibly) previous years' data.

Example of Regression: Equaliser

$$\boxed{\text{Tx}} \rightarrow x(k) \rightarrow \boxed{\text{channel}} \rightarrow Z(k) \rightarrow \boxed{\text{filter}} \rightarrow y(k) \rightarrow \boxed{\text{Rx}}$$

- We want $y(k) = x(k)$.
- Design (or adapt) the filter to achieve this. We can choose a filter as

$$y(k) = \sum_{i=0}^{T} a_i Z(k-i)$$

Find 'best' a_i – a function learning problem.

- Training set: $\{(x(k), Z(k)), k = 1, 2, \cdots, N\}$

- Both classification and regression involve learning from examples.
- As we said they represent the basic problem addressed under Machine learning.

Learning from Examples – Generalization

- To obtain a classifier (or a regression function) we use the training set.
- We 'know' the class label of patterns (or the values for prediction variable) in the training set.
- Errors on the training set do not necessarily tell how good is the classifier.
- Any classifier that amounts to only storing the training set is useless.
- Interested in the 'generalization abilities' – how does our classifier perform on unseen or new patterns.

- In practice we assess the generalization ability of a a learnt classifier using the so called *t*est set.
- Basic tenet of ML: Training data is 'similar' to the data on which the classifier is expected to perform well.

Designing Classifiers

- The classifier should perform well inspite of inherent variability of patterns and noise in feature extraction and/or in class labels as given in training set.
- The difficulties are
  - Lot of variability in patterns of a single class
  - Feature vectors of patterns from different classes can be arbitrarily close.
  - Noise in measurements

Variability in Patterns

INTRODUCTION 3

Figure 1.1. Variety of different images all representing the same character *A* (from Hofstadter's *Metamagical Themes: Questing for the Essence of Mind and Pattern* [HOF1985]).

- Statistical Pattern Recognition – An approach where the variabilities are captured through probabilistic models.

The Statistical Approach

- We assume that the training examples are drawn *iid* according to some joint distribution of (X, y).
- The learnt classifier is evaluated using random examples drawn from the same distribution.
- Given a distribution from which data comes, we can ask what is the 'best' classifier.
- We can use the training data to learn the relevant distributions and then implement the 'best' classifier.

Notation

- $\mathcal{X}$ – the feature space. (We take $\mathcal{X} = \Re^n$).
- $\mathcal{Y} = \{1, \cdots, K\}$ – the set of class labels.
- The simplest case is $K = 2$. (Many times, we consider a 2-class problem to simplify notation).
- We can, in principle, solve a general problem if we can tackle the $K = 2$ case. (E.g., 'one-Vs-rest')

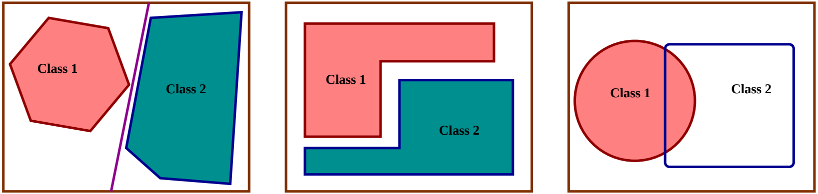
Notation – Classifiers

- A classifier: $h : \mathcal{X} \rightarrow \mathcal{Y}$
- Sometimes we take $h : \mathcal{X} \rightarrow \mathcal{A} \subset \Re^K$ as a classifier. (Here K is the number of classes).
- The j^{th} component of $h(\cdot)$ could be 'score' for class-j.
- We will write h for any classifier.

Notation – Class conditional densities

- f_i – the probability density function of the feature vectors from class-i, $i = 0, 1$.
- f_i are called *class conditional densities*.
- Let $\mathbf{X} = (X_1, \cdots, X_n) \in \Re^n$ represent the feature vector.
- Then f_i is the (conditional) joint density of the random variables $X_1, \cdots, X_n$ given that $\mathbf{X}$ is from class-i.

- Class conditional densities model the variability in the feature values.
- For example, the two classes can be uniformly distributed in the two regions as shown.



Notation – prior and posterior probabilities

- $p_i = \text{Prob}[y = i]$ – prior probabilities
- $q_i(\mathbf{x}) = \text{Prob}[y = i | \mathbf{X} = \mathbf{x}]$ – posterior probabilities
- By Bayes theorem

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

where $Z = \sum_i f_i(\mathbf{x}) p_i$.

- We would not have knowledge of f_i, p_i etc.
- All we have is the training set of examples.
- We have to use the training data to learn a 'model'

- The statistical viewpoint gives us one way of looking for ‘optimal’ classifier.
- We can say, for example, we want a classifier that has least probability of misclassifying a random pattern (drawn from the underlying distributions).

Bayes Classifier (2-class case)

- The Bayes classifier:

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

- Given the underlying probability model, this is the optimal classifier for minimizing probability of error. (We show this later).
- $q_0(\mathbf{x}) > q_1(\mathbf{x})$ is same as $p_0 f_0(\mathbf{x}) > p_1 f_1(\mathbf{x})$.
  We need to learn p_0, p_1, f_0, f_1 from the data to implement Bayes Classifier.
- This is an example of what is called a Generative Model based approach.

Generative Models

- Training data: $\{(\mathbf{X}_i, y_i), \;\; i = 1, \ldots, \ell\}$
- We view them as *iid* realizations of $(\mathbf{X}, y)$.
- Generative model: Joint distribution of $(\mathbf{X}, y)$.
- For example, learning f_i and p_i.
- Bayes classifier is a generative model based approach
- There are different ways to represent generative models.

Learning a Generative Model

- Given *iid* data, $\mathcal{D} =< \mathbf{z_1}, \cdots, \mathbf{z_n} >$ we want to estimate the underlying probability distribution.
- One can assume a parametric model: $f(\mathbf{z}|\theta)$.
- That is, we assume form of distribution is known and we need to estimate the parameters, θ.
- Parameters θ are learnt using training data.
- The density (with learnt parameter values) is used to derive the optimal classifier.
- There are different techniques to fit a density model to given data.

Discriminative models

- We model only the conditional distribution of y conditioned on $\mathbf{X}$.
- That is, we (directly) learn $q_i(\mathbf{X})$.
- We need some representation for the posterior probabilities.
- Or some other parameterized representation for the classifier.

How to Rate Different Classifiers

- We can rate different classifiers, for example, by

$$F(h) = \mathrm{Prob}[h(\mathbf{X}) \neq y(\mathbf{X})]$$

  where $y(\mathbf{X})$ denotes the (random variable that is the) class of $\mathbf{X}$.
- $F(h)$ is probability that h misclassifies a random X.
- Optimal classifier – one with lowest value of F.
- Bayes Classifier minimizes this F.
- Note that for a given h we cannot even calculate $F(h)$ unless we know the joint distribution of (X, y).

Risk of a Classifier

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

$$F(h) = E[L(h(\mathbf{X}), y(\mathbf{X}))]$$

  The $F(h)$ is called **risk** of h.
- Now the objective is to find a classifier with minimum risk.

Example: 0–1 Loss Function

- The 0-1 loss fn:

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

  Now $F(h) = E[L(h(\mathbf{X}), y(\mathbf{X}))] = \text{Prob}[h(\mathbf{X}) \neq y(\mathbf{X})]$. (Same as before)
- A more general loss function, in the 2-class case, 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.

Learning Classifiers

- Recall we want h to minimize

$$F(h) = E[L(h(\mathbf{X}), y(\mathbf{X}))]$$

- We can (in principle) achieve this if we know joint distribution of $(\mathbf{X}, y)$.
- Example: Bayes classifier
- Hence one approach is to learn the joint distribution from data.
- Once we learn the joint distribution, it can be used for other purposes also.

Learning Classifiers

- We want a h to minimize

$$F(h) = E[L(h(\mathbf{X}), y(\mathbf{X}))]$$

- We can choose a parametric form: $h(W, \cdot)$.
- For example, for a K-class problem

$$h(W, \mathbf{x}) = j \text{ iff } g_j(W, \mathbf{x}) > g_i(W, \mathbf{x}), \; \forall i \neq j$$

- We can think of $g_i(W, \mathbf{x})$ to be approximation of $q_i(\mathbf{x})$.
- The g_i are called discriminant functions
- We can learn W to minimize risk.
- A discriminative approach.

Linear Classifiers

- Consider 2-class case with $h(W, \mathbf{x}) = \text{sign}(g(W, \mathbf{x}))$.
- Choose: $g(W, \mathbf{x}) = W^T\mathbf{x} + b$
- Perceptron is one of the earliest such Linear Classifiers.
- $h(W, \mathbf{x}) = \text{sign}(g(W, \mathbf{x}))$ is often called a discriminant function based Classifier.
- $g(W, \mathbf{x}) = W^T\mathbf{x} + b$ is called a linear discriminant function.

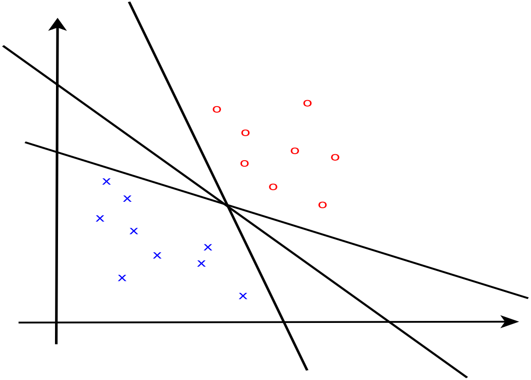
Linear Separability

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

$$\begin{aligned} \mathbf{X}_i^T W^* + b^* &> 0 \text{ if } y_i = 1 \\ &< 0 \text{ if } y_i = 0 \end{aligned}$$

- Any W^*, b^* that satisfies the above is called a separating hyperplane. (The separating hyperplane is given by $\mathbf{X}^T W^* + b^* = 0$).
- There exist infinitely many separating hyperplanes if the set is linearly separable.

Linearly Separable Data – infinitely many separating hyperplanes



Learning Classifiers

- Consider $h(W, \mathbf{X}) = \text{sign}(g(W, \mathbf{X}))$.
- Now each classifier is defined by a parameter vector, W.
- So, we need to find W to minimize

$$F(W) = E[L(h(W, \mathbf{X}), y(\mathbf{X}))]$$

- Since we do not know the relevant joint distribution, this is not a straight-forward optimization problem.
- We can approximate the Expectation above by sample mean.

Empirical Risk

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

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

  where $\{(\mathbf{X}_i, y_i), i = 1, \cdots, \ell\}$ is the training set of examples.
- Then $\hat{F}(W)$, called empirical risk, is a good approximation to $F(W)$.
  (Law of large numbers; assume examples are *iid*)
- So, we can minimize $\hat{F}$ instead.

Empirical Risk Minimization

- Empirical risk is the ‘training error’

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

- We search over a family of classifiers to minimize empirical risk.
- The true risk is the ‘test error’

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

- We actualy want the minimizer of the true risk.
- If we have ‘sufficient’ number of ‘representative’ training samples then minimizer of $\hat{F}$ would be good enough.
- Empirical Risk Minimization is a standard ML strategy.

Learning Linear Classifiers

- Linear classifier (2-class): $h(W, \mathbf{X}) = \text{sign}(W^T \mathbf{X})$.
- We need to find W to minimize

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

- In general we need some optimization techniques.
- As defined, h (and hence $\hat{F}$) may be discontinuous. Also, if we use 0–1 loss function, L is also discontinuous.
- We can, e.g., take $h(W, \mathbf{X}) = W^T \mathbf{X}$ and use losses other than 0–1 loss.

Example – Learning a Linear Classifier

- **squared error loss**: $L(h(W, \mathbf{X}), y) = (h(W, \mathbf{X}) - y)^2$
- Suppose we use squared error loss with a linear classifier

$$\hat{F}(W) = \frac{1}{\ell} \sum_{i=1}^{\ell} (W^T \mathbf{X}_i - y_i)^2$$

.

- Now we can use standard optimization techniques to minimize $\hat{F}$.
- This is called the linear least squares method.

Linear Least squares method

- Consider a 2-class problem with $y \in \{0,\ 1\}$.
- Suppose we want to minimize (over all possible g)

$$F = E[g(\mathbf{X}) - y]^2$$

- Solution: $g^*(\mathbf{X}) = E[y|\mathbf{X}] = P[y = 1|\mathbf{X}] = q_1(\mathbf{X})$
- Thus, in linear least squares method, we are trying to find 'best linear approximation' of the posterior probability

Logistic Regression

- Linear function is not a good model for posterior probability.
- We can instead choose a model

$$P[y = 1|\mathbf{X}] = \frac{1}{1 + \exp(-W^T\mathbf{X})}$$

- This is known as Logistic Regression

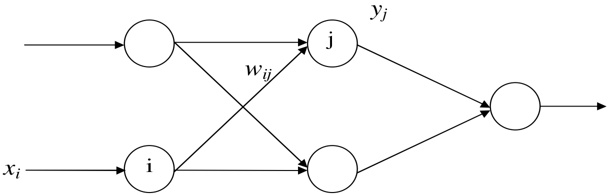
- There are efficient algorithms for learning linear classifiers.
- In learning such linear models we can use $W^T\Phi(X)$ instead of W^TX where $\Phi(X) = [\phi_1(X), \cdots, \phi_m(X)]^T$ as long as ϕ_i are fixed functions.
- It is like using $z_i = \phi_i(X)$ as features.

Beyond Linear Models

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

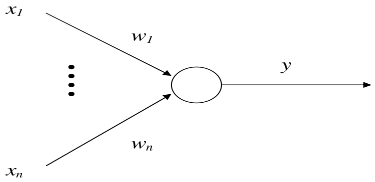
The Neural Networks Approach

- 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.
- Models are inspired by the architecture of brain

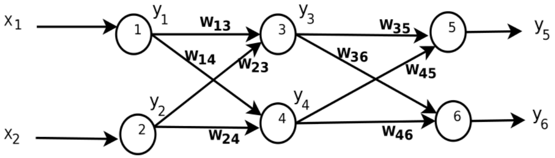
Single Neuron Models



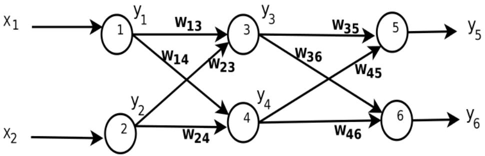
- x_i are inputs into the (artificial) neuron and w_i are the corresponding *weights*. y is the output of the neuron
- Net input : $\eta = \sum_j w_j x_j$
- output: $y = f(\eta)$, where $f(.)$ is called activation function (Perceptron, AdaLinE are such models).

Network of Neurons

- We can connect a number of such units or neurons to form a network. Inputs to a neuron can be outputs of other neurons (and/or external inputs).



- Notation:
  y_j – output of j^{th} neuron;
  w_{ij} – weight of connection from neuron i to neuron j.



- Each neuron computes weighted sum of inputs and passes it through its activation function, to compute output
- For example, output of neuron 5 is

$$
\begin{aligned}
y_5 &= f_5\left(w_{35}\, y_3 + w_{45}\, y_4\right) \\
&= f_5\left(w_{35}\, f_3(w_{13}y_1 + w_{23}y_2) + w_{45}\, f_4(w_{14}y_1 + w_{24}y_2)\right)
\end{aligned}
$$

- By convention, we take $y_1 = x_1$ and $y_2 = x_2$.
- Here, x_1, x_2 are inputs and y_5, y_6 are outputs.

- A single neuron ‘represents’ a class of functions from $\Re^m$ to $\Re$.
- Specific set of weights realizes specific functions.
- By interconnecting many units/neurons, networks can represent more complicated functions from $\Re^m$ to $\Re^{m'}$.
- The architecture constrains the function class that can be represented. Weights define specific function in the class.

Feedforward networks

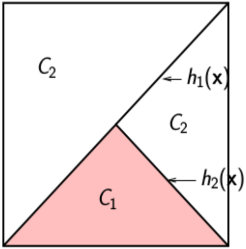
- These networks are very effective in learning nonlinear pattern classifiers.
- They learn under supervised learning framework – given examples of needed input-output behaviour, the weights are adapted to achieve that function.
- Trained through minimizing empirical risk (training error) using, e.g., squared error loss.
- Gradient descent on this empirical risk is the so called backpropagation algorithm.

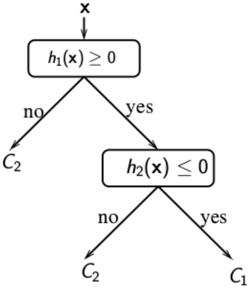
The Decision Tree Approach

- Divide feature space so that a linear classifier is enough in each region
- Gives us a piece-wise linear classifier.
- Decision trees represent such classifiers.

Decision Trees – Overview

- A decision tree is a very popular piece-wise linear classification method.
- Here is an example decision tree





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

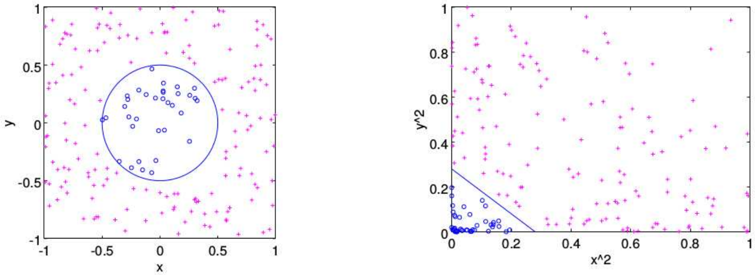
Decision Trees – Overview

- Each (non-leaf) node is labelled with a so called ‘split rule’.
- All leaf nodes are labelled with a class label.
- Given a feature vector, at each node we go to one of the children based on the value of split rule on this feature vector.
- When we reach a leaf that gives the classification of the feature vector.
- Most decision tree algorithms learn the tree in a top-down fashion under a greedy heuristic.

Support Vector Machines idea

- Map X nonlinearly into a high dimensional space and try a linear classifier there.
- Let $X = [x_1 \quad x_2]$ and let $\phi : \Re^2 \to \Re^6$ given by $Z = [z_0\ z_1 \cdots z_5] = \phi(X) = [1 \quad x_1 \quad x_2 \quad x_1^2 \quad x_2^2 \quad x_1 x_2]$
- 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
$$g(Z) = a_0 z_0 + a_1 z_1 + a_2 z_2 + a_3 z_3 + a_4 z_4 + a_5 z_5$$
is a linear discriminant function in the '$\phi(X)$' space.

Transforming Patterns to become Linearly Separable



- The SVM method does this mapping implicitly using the so called kernel function.
- It results in a quadratic programming problem (optimizing a quadratic objective function under linear constraints). Computationally efficient.

Many Other issues

- Model Selection: How to choose and validate a parametric model.
- Model Estimation: What we considered – methods to learn the classifier
- Model Assessment: How to estimate true risk of learnt classifier (training, validation and testing)
- Scalability of algorithms
- Noise Robustness
- Dimensionality Reduction/Models with Latent variables.
- Classifier Combinations

  ⋮

Organization of this course

- Bayes classifier for minimizing risk (and some variations)
- Estimation of class conditional densities.
- Parametric and non-parametric models
- ML and Bayesian estimation
- Mixture models and EM algorithm
- Probabilistic graphical models (?)

- Learning linear classifiers and regression models
- Perceptron and LMS algorithm
- Linear Least squares estimation.
- Logistic regression.

- Simple introduction to statistical learning theory
- Probably Approximately Correct (PAC) learning framework
- Complexity of a learning problem – VC dimension
- Consistency of Empirical Risk minimization

- Learning Nonlinear models – Neural networks.
- Feedforward networks and Backpropagation
- RBF Networks (?)
- Some issues in deep neural networks
- CNNs, Autoencoders
- Recurrent neural networks (?)
- RBMs, other generative models such as GAN, variational autoencoders (?)

- Learning nonlinear models – SVMs
- Optimal Separating hyperplane
- SVM (Kernel based) methods for classification and regression

- Assessing learnt classifier, cross validation, Bagging and Boosting
- Classifier combinations, AdaBoost
- PCA, Feature Extraction/dimensionality reduction

Conclusions

- ML comprises of methods to fit models to data (for prediction/decision making)
- Statistical approach to ML uses probability models
- Generative models learn joint distribution of $(\mathbf{X}, y)$.
- Discriminative models learn conditional distribution of y given $\mathbf{X}$.
- Empirical risk minimization is a general strategy for learning.

Thank You!