QF 624: Machine Learning for Financial Applications

Technical Material (Appendices)

Master of Science in Quantitative Finance
Lee Kong Chian School of Business

Saurabh Singal

July 2018

Additional Material

- Appendix 1: Clustering
- Appendix 2: Gaussian Mixture Models (GMM)
- Appendix 3: Support Vector Machines (SVM)
- Appendix 4: Hidden Markov Model (HMM)
- Appendix 5 Neural Networks
- Appendix 6: Backpropagation explained in depth

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Appendix-1: Clustering

- What is Clustering?
- Divide data points into groups, such that points within each group are similar to each other and dissimilar to points outside of this group.
- The two broad families: Agglomerative and Divisive Clustering

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Agglomerative Clustering

- In **Agglomerative** clustering, each data point is initially assigned to one cluster,
  - At subsequent steps, we merge two clusters into one,
  - The number of clusters is reduced by one at each step.
  - We start with N clusters and end with 1 cluster.
  - Bottoms up approach

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Divisive clustering

- In Divisive clustering, all the data points are assigned to a single cluster
- and at each time step, we split a cluster.
- The number of clusters is increased by one at each step.
- We start with 1 cluster and end with N clusters.
- Top Down approach

SMU
SINGAPORE MANAGEMENT UNIVERSITY

K-means

- K-means clustering was developed at Bell Labs (the home of UNIX, C, C++ and several scripting shells).
- The idea is to choose cluster centers, (initialize step) and assign each data point to the cluster based on proximity to the cluster centre; then re-calculate the cluster centres and re-assign the cluster memberships. Stop when assignments don't change. Note this is a heuristic; there is no guarantee that the optimum is found.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Fuzzy Clustering

- A data point may belong to more than one cluster
- The Fuzzy c-means algorithm
- There is a parameter m called fuzzifier. In the limit, it is similar to k-means.
- The distance of each data point to the cluster centre is computed, and the nearer it is to a given cluster centre, the higher is its membership weight for the cluster. These weights are normalized so that they sum to 1.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Appendix 2: Gaussian Mixture Models

- We mentioned Gaussian Mixture Models; here is an example on how to fit a mixture of two Gaussian distributions to an observed set of points.
- The algorithm used is called EM Algorithm (Expectation Maximization) and it is essentially an application of Bayes' Theorem.
- This example is based on : Gaussian Mixture Models and Introduction to HMM's (Michael Picheny, Bhuvana Ramabhadran, Stanley F. Chen) ,
- We wrote the MATLAB code to estimate the model and the R code to plot the diagrams and is included in the appendix.
- Suppose you have the following 10 points and we want to fit a mixture of two Gaussian distributions, $N_1 (\mu_1 , \sigma_1)$ and $N_2 (\mu_2 , \sigma_2)$ along with the probability p_1 that a point comes from N_1 (and probability $p_2 = 1 - p_1$ that the point is drawn from N_2).
- Data = 8.4, 7.6, 4.2, 2.6, 5.1, 4.0, 7.8, 3.0, 4.8, 5.8

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Using Expectation Maximization to fit GMM

- To find the probability observing a given point x, we need to sum over all possible values of hidden variable h, which can be 1 or 2 (indicating from which distribution the point x comes from).
- Start with initial estimates: $p_1 = p_2 = 0.5$; $\mu_1 = 4$, $\mu_2 = 7$, $\sigma_1^2 = \sigma_2^2 = 1$

- $P(x) = \sum_{h=1}^{h=2} P(x,h) = \sum_{h=1}^{h=2} P(x|h)P(h)$
- For each data point xi , assign single hidden value h_i.
- $h_i = \arg max_h P(h) P(x_i|h)$

- Identify GMM component generating each point.
- Update parameters in $P(h), P(x|h)$ by simply counting and normalizing to get MLE for μj , j , σj .
- Posterior prob $\tilde{P}(h|xi) = \frac{P(h,xi)}{\sum_h P(h,xi)}$.

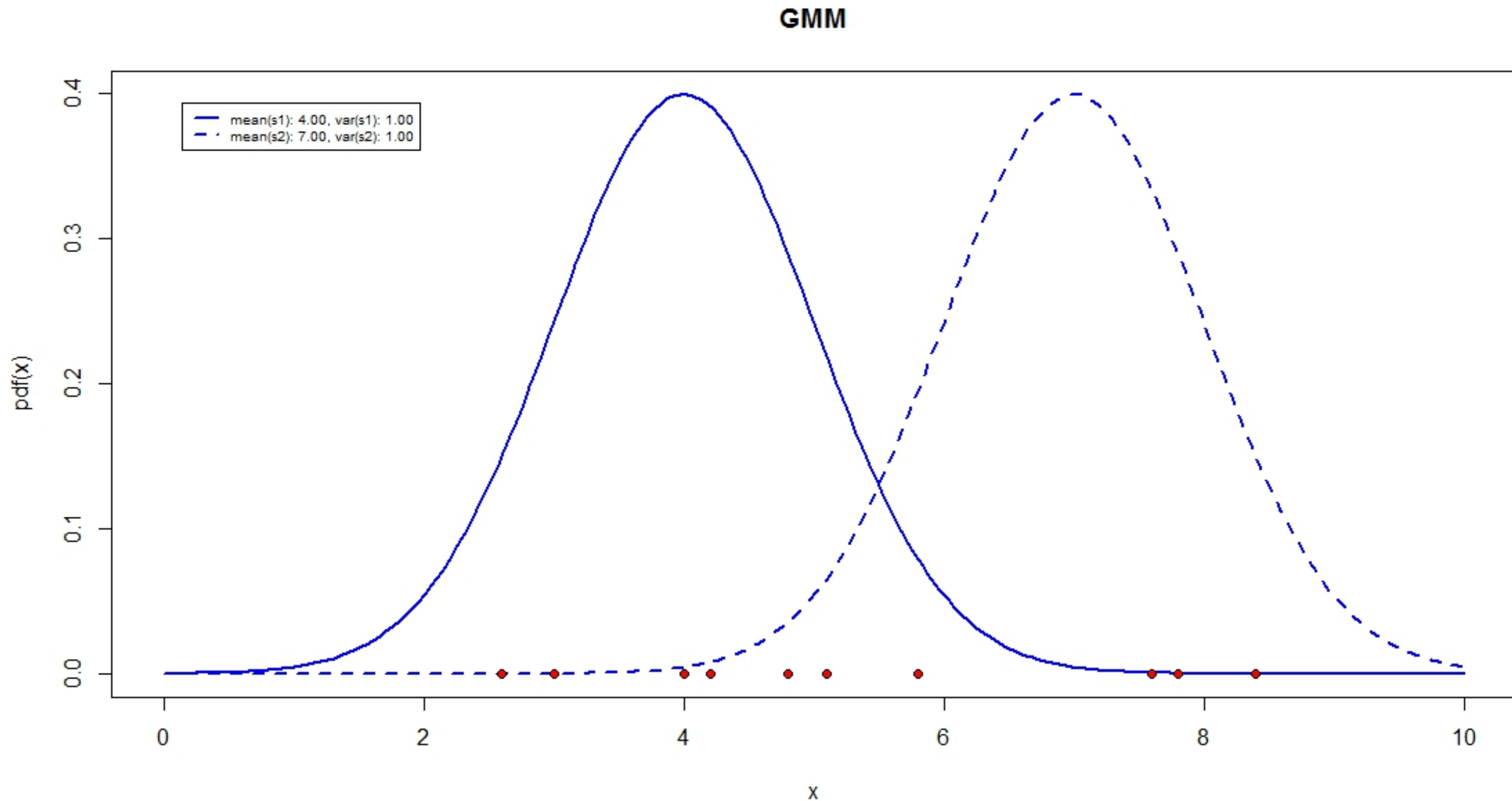
SMU SINGAPORE MANAGEMENT UNIVERSITY

GMM: Parameters after each Iteration

- After only a few iterations, the algorithm converges to the following parameter estimates: p_1 =0.7, μ_1 =4.22, σ_1^2 =1.13; and p_2 = 0.3, μ_2 = 7.93 , σ_2^2 = 0.12
- The intermediate values of the parameters at each iteration are shown in the table below. The next few figures show the distributions plotted at some of the iterations.

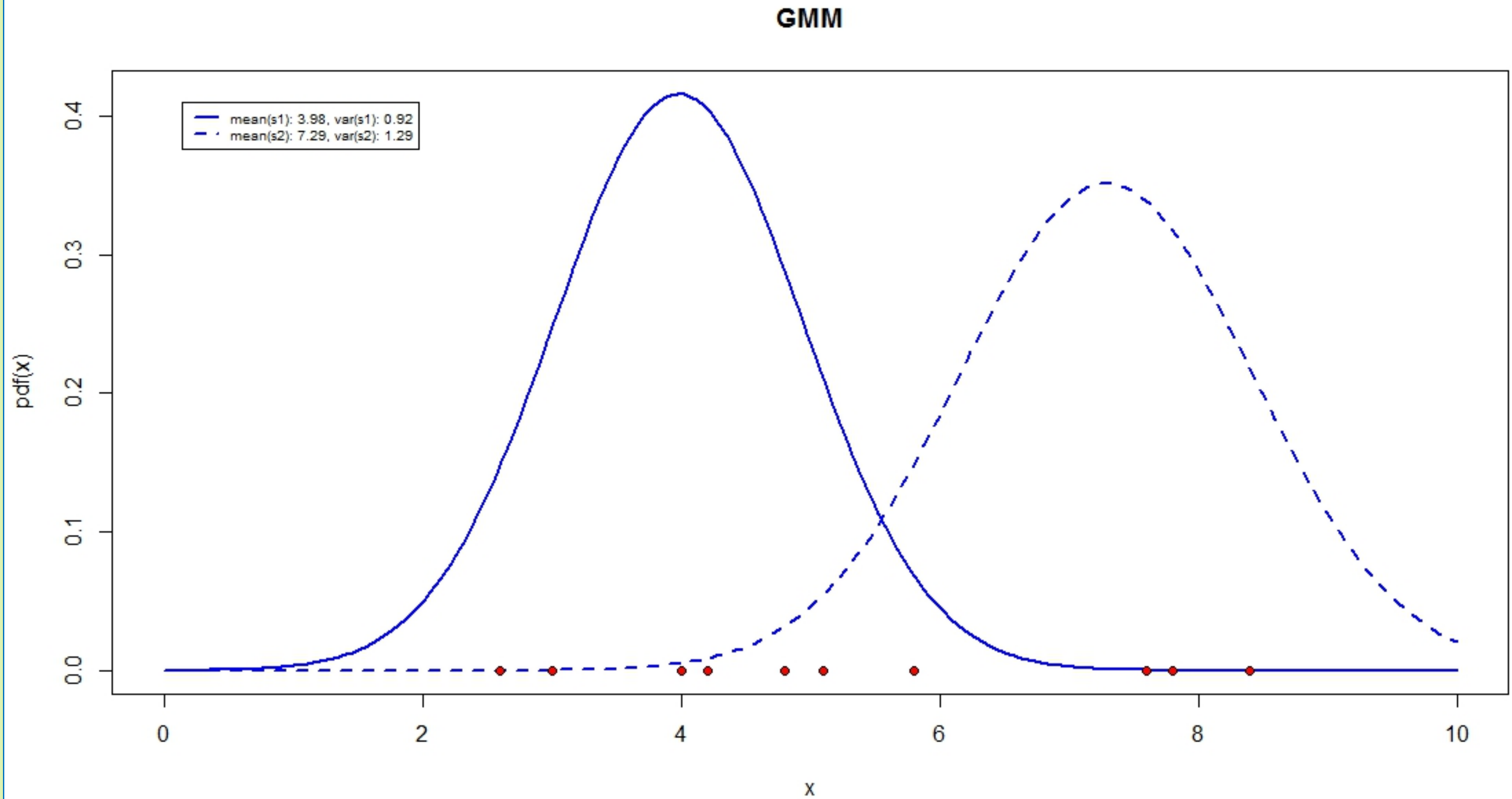
Iteration	p_1	μ_1	σ_1^2	p_2	μ_2	σ_2^2
1	0.59	3.98	0.92	0.41	7.29	1.29
2	0.62	4.03	0.97	0.38	7.41	1.12
3	0.64	4.08	1.00	0.36	7.54	0.88
4	0.66	4.14	1.05	0.34	7.69	0.59
5	0.69	4.19	1.10	0.31	7.85	0.28
6	0.70	4.22	1.13	0.30	7.93	0.12
7	0.70	4.22	1.13	0.30	7.93	0.12
8	0.70	4.22	1.13	0.30	7.93	0.12
9	0.70	4.22	1.13	0.30	7.93	0.12
10	0.70	4.22	1.13	0.30	7.93	0.12

GMM Figure using Initial Parameters



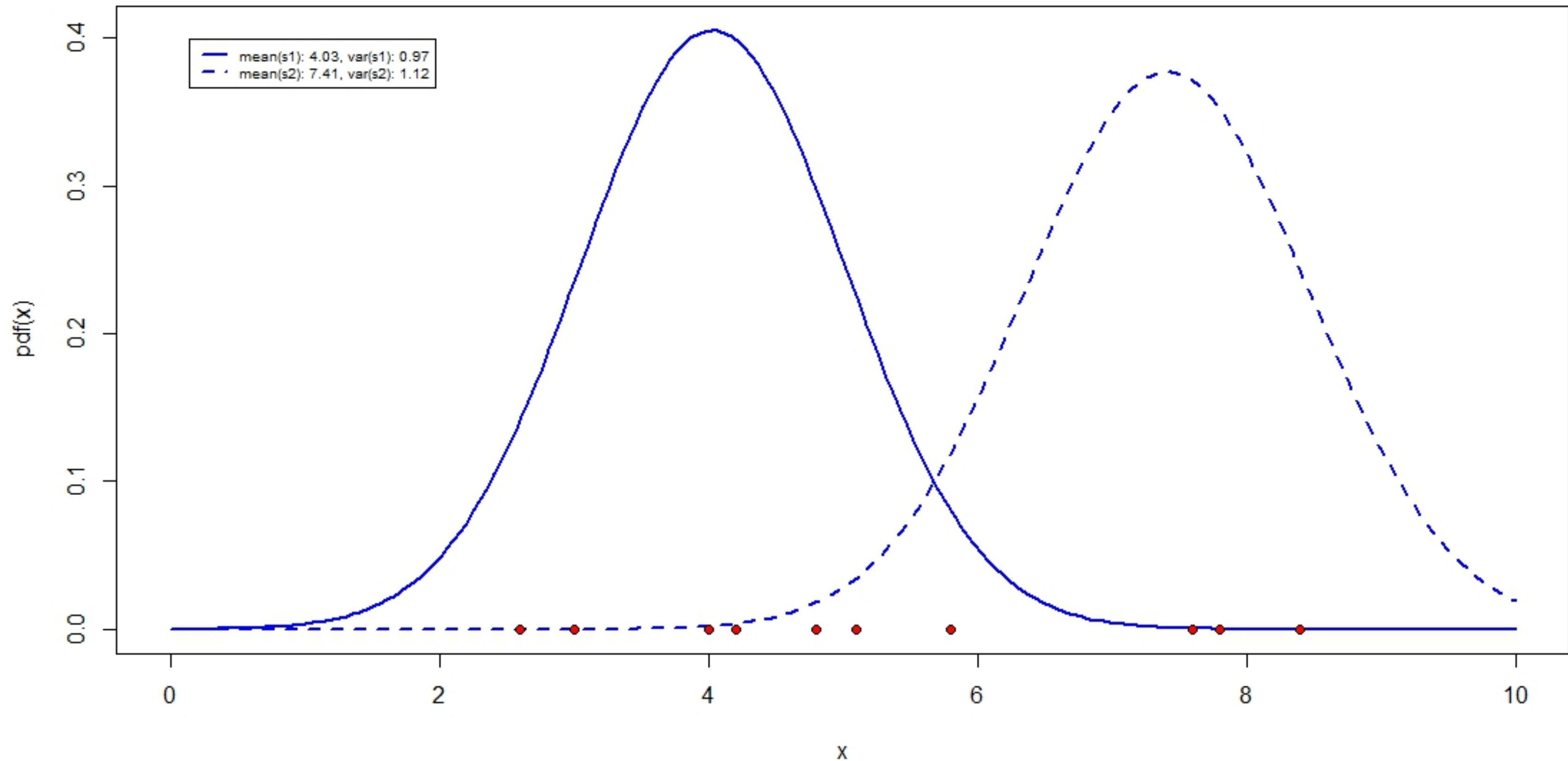
SMU
SINGAPORE MANAGEMENT UNIVERSITY

GMM-Figure after iteration 1



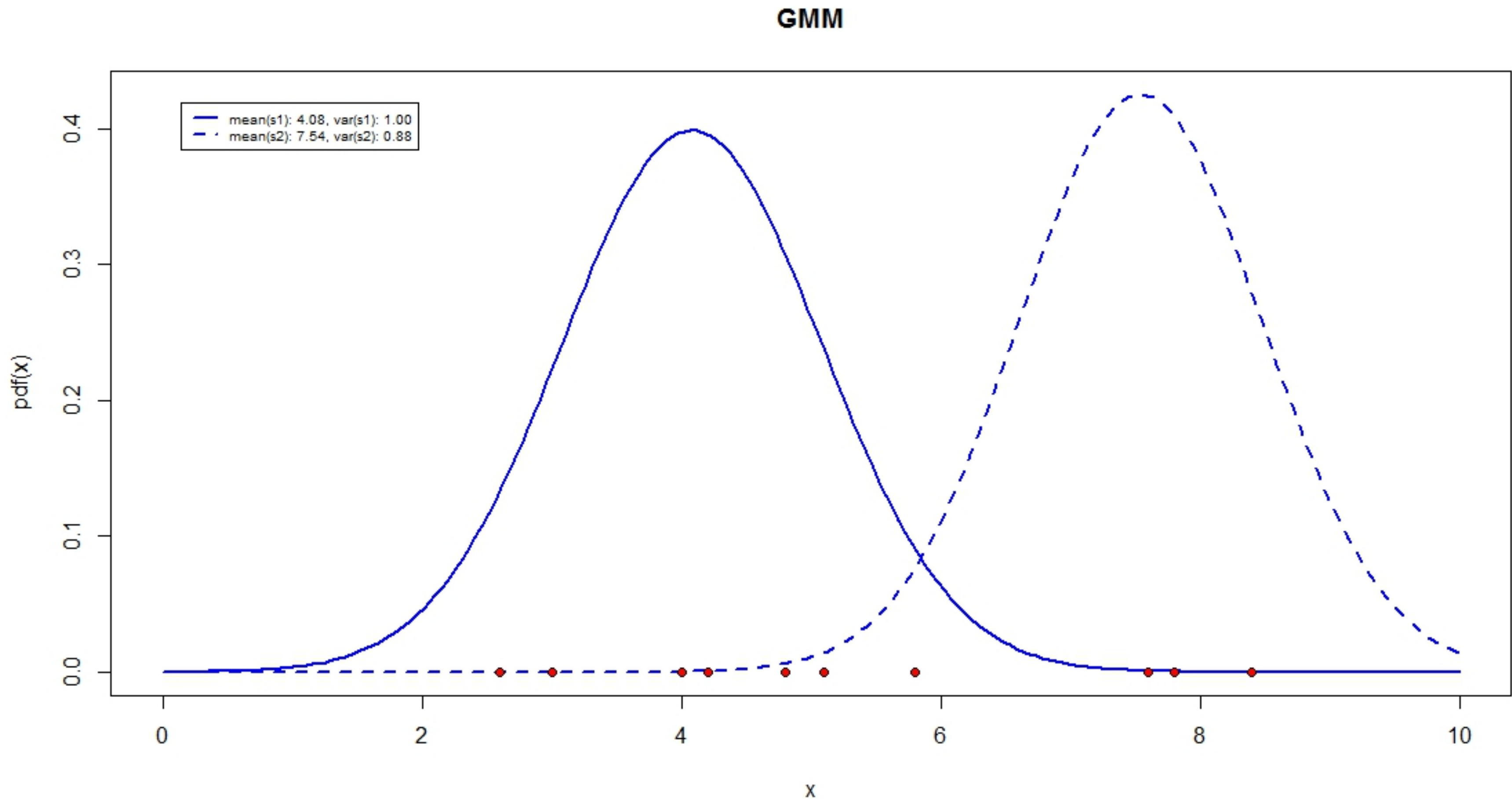
SMU
SINGAPORE MANAGEMENT UNIVERSITY

GMM figure after Iteration 2



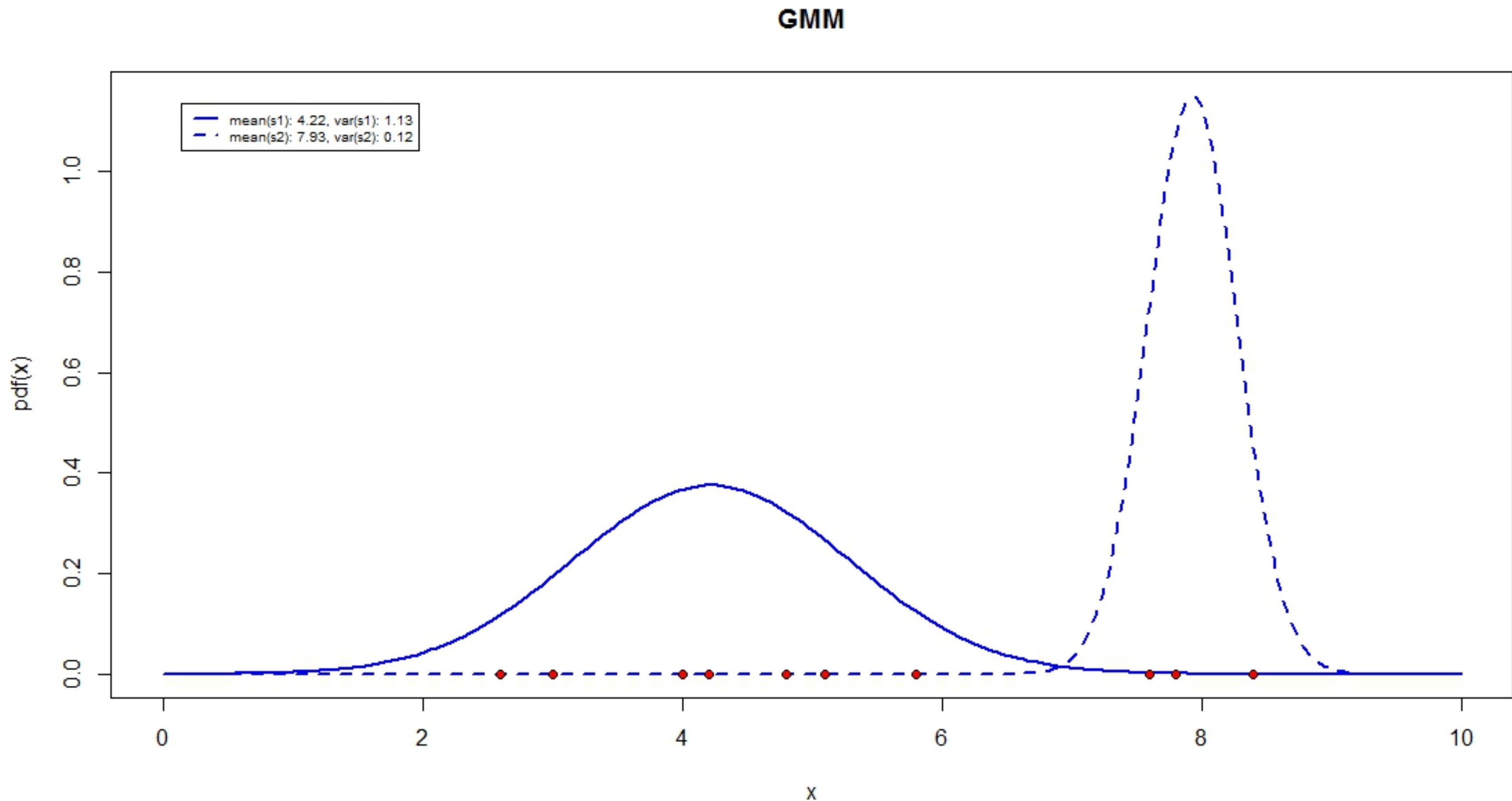
SMU SINGAPORE MANAGEMENT UNIVERSITY

GMM figure after iteration 3



SMU SINGAPORE MANAGEMENT UNIVERSITY

GMM- Final Figure after Iteration 10



MATLAB code to fit the GMM

```
 p1 = 0.6;   p2 = 1-p1;
mu1= mean(x)*1.1 ; mu2 = mean(x) * 0.9;
 sigsqr1=var(x) *1.1;    sigsqr2=var(x)*0.9;
 numpoints=length(x); numiters=10;
 for i=1:numiters
    if i==1
        prob=[p1 p2];    mu= [mu1 mu2];     sigsqr=[sigsqr1 sigsqr2];  sigma_1_sqrd=sigsqr(1);   sigma_2_sqrd=sigsqr(2);
    else
        prob=[ p1_hat(i-1)  p2_hat(i-1)]; mu=[mu1_hat(i-1) mu2_hat(i-1)]; sigma_1_sqrd=sig1sqr_hat(i-1);sigma_2_sqrd=sig2sqr_hat(i-1);
        p1_hat(i-1);
    end
a1 = normpdf(x,mu(1),sqrt( sigma_1_sqrd));
b1 = normpdf(x,mu(2),sqrt( sigma_2_sqrd));
 P1N1_1 = prob(1)*a1  ;    P2N2_1  = prob(2)* b1 ;
  Pxi= P1N1_1+P2N2_1  ;
  prob_1_given_xi = P1N1_1./Pxi ;    prob_2_given_xi = P2N2_1./Pxi    ;
   p1_hat(i)=sum(prob_1_given_xi)/(sum(prob_1_given_xi)+sum(prob_2_given_xi))       ;
  p2_hat(i)= sum(prob_2_given_xi)/(sum(prob_1_given_xi)+sum(prob_2_given_xi))        ;
  xi_prob_1_given_xi=x.*prob_1_given_xi   ;    xi_prob_2_given_xi = x.*prob_2_given_xi ;
          mu1_hat(i)=sum(xi_prob_1_given_xi)/sum(prob_1_given_xi);
    mu2_hat(i)=sum(xi_prob_2_given_xi)/sum(prob_2_given_xi);
    rep_mu1= repmat(mu1_hat(i),numpoints,1);  rep_mu2= repmat(mu2_hat(i),numpoints,1);
    centred_x_sqr_prob_1_given_xi = (x-rep_mu1).^2 .* prob_1_given_xi  ;
    centred_x_sqr_prob_2_given_xi = (x-rep_mu2).^2 .* prob_2_given_xi  ;
        sig1sqr_hat(i)=sum(centred_x_sqr_prob_1_given_xi)/sum(prob_1_given_xi) ;
    sig2sqr_hat(i)=sum(centred_x_sqr_prob_2_given_xi)/sum(prob_2_given_xi) ;
    end
```

SMU
SINGAPORE MANAGEMENT UNIVERSITY

R code to plot the GMM figures

```
g1 = cbind(c(4,1),c(3.98,0.92),c(4.03,0.97),c(4.08,1.00),c(4.22,1.13));
g2 = cbind(c(7,1),c(7.29,1.29),c(7.41,1.12),c(7.54,0.88),c(7.93,0.12));
for (t in 1:ncol(g1))
{
  x<-seq(0,10,length=200)
  s1 = sprintf("mean(s1): %.2f, var(s1): %.2f", g1[1,t],g1[2,t])
  s2 = sprintf("mean(s2): %.2f, var(s2): %.2f", g2[1,t],g2[2,t])
  y<-dnorm(x,mean=g1[1,t], sd=sqrt(g1[2,t]))
  y2<-dnorm(x,mean=g2[1,t], sd=sqrt(g2[2,t]))
  matplot(x, cbind(y,y2),type="l",ylab= "pdf(x)",col=c("blue","blue"),lty=c(1,2), lwd = c(2,2))
  title("GMM")
  par(xpd=TRUE)
  legend("topleft", cex=0.6, inset=.05, legend=c(s1, s2), lwd=c(2.5,2.5),lty=c(1,2), col=c("blue","blue"))
  points(8.4,0, pch=21, bg="red")
  points(7.6,0,pch=21, bg="red")
  points(4.2,0,pch=21, bg="red")
  points(2.6,0,pch=21, bg="red")
  points(5.1,0,pch=21, bg="red")
  points(4.0,0,pch=21, bg="red")
  points(7.8,0,pch=21, bg="red")
  points(3.0,0,pch=21, bg="red")
  points(4.8,0,pch=21, bg="red")
  points(5.8,0,pch=21, bg="red")
 }
```

SMU
SINGAPORE MANAGEMENT UNIVERSITY

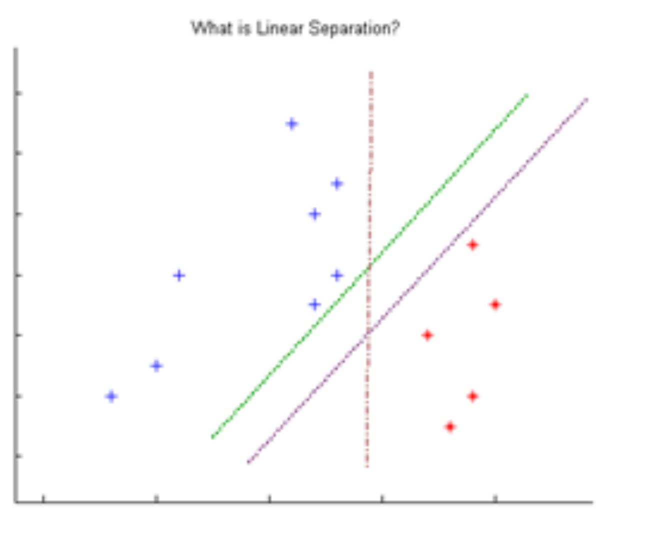
Appendix-3: Support Vector Machines

- The Support Vector Machine (SVM) is a technique for classification and regression. Originally the SVM was devised for binary classification, or classifying data into two types. Generalization when there are more than two classes is relatively straightforward.
- For linearly separable data, SVM finds optimal decision boundary using a linear decision surface. When working with non-linearly separable data in the original space, SVM maps the patterns to a higher dimensional feature space in which the transformed data becomes linearly separable. This conversion can be done using kernel function, and the commonly used kernels functions are listed below:

Name of Kernel Function	Definition
Linear	$K(\mathbf{u},\mathbf{v}) = \mathbf{u}^T\mathbf{v}$
Polynomial of degree d	$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}^T\mathbf{v}+1)^d$
Gaussian Radial Basis Function (RBF)	$K(\mathbf{u},\mathbf{v}) = e^{-\frac{1}{2}[(\mathbf{u}-\mathbf{v})^T\Sigma^{-1}(\mathbf{u}-\mathbf{v})]}$
Sigmoid	$K(\mathbf{u},\mathbf{v}) = \tanh[\mathbf{u}^T\mathbf{v}+b]$

- Solution to SVM can be formulated as a Quadratic Programming Problem. It can be easily implemented by most of the popular statistical languages (MATLAB, R, etc.) or packages (LibSVM).

SVM and Linear Separation



- *Red Asterisk markers and Blue Plus markers are patterns belonging to Class 1 and 2 respectively. Each of the three Straight Lines can separate the test patterns. This is an example of Linear Separation.*

SMU
SINGAPORE MANAGEMENT UNIVERSITY

SVM and Linear Discrimination

- Consider the function $g(\mathbf{x})=\mathbf{w}^T\mathbf{x} + b$.
- If $\mathbf{w}^T\mathbf{x} + b>=0$, classify **x** as belonging to class 1
- If $\mathbf{w}^T\mathbf{x} + b<0$, classify **x** as belonging to class 2
- In the case of two-dimensional **x** and **w**, $\mathbf{w}^T\mathbf{x} + b=0$ defines a straight line. Points on one side of this straight line will be classified as belonging to class 1; points on the other side of this line will be classified as belonging to class 2.
- But there are an infinite number of straight lines that can linearly separate the data points; we can simply vary b to get parallel lines that will do the job. So we need to determine the "best" or optimal **w** and *b*

SMU
SINGAPORE MANAGEMENT UNIVERSITY

SVM and Maximum Margin Separation

- To choose "good" w and b, we measure the distance r(x) of x from the decision surface g(x)=0. The distance r, of a point x from the plane P specified by (w, b) is

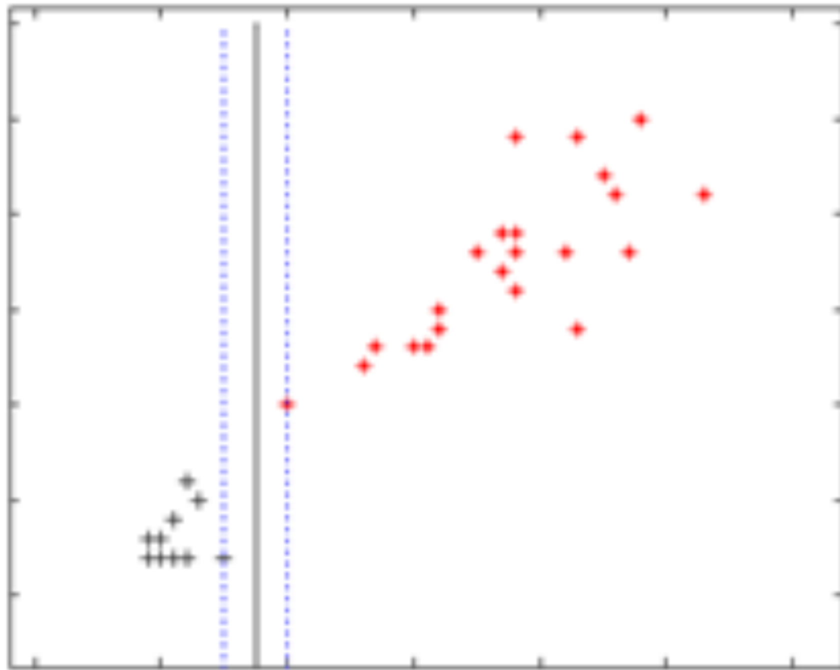
  $$r(\mathbf{x};\mathbf{w},b) = |g(\mathbf{x})|/||\mathbf{w}|| = |\mathbf{w}^T\mathbf{x} + b|/||\mathbf{w}||$$

- When we talk of the distance from a point to a plane we mean the distance from x to the nearest point xp that lies on the plane P. The margin of separation, M, measures the distance between the two classes; $M=2/||\mathbf{w}||$

- The ***optimal separating hyperplane*** separates the two classes and maximizes the distance to the closest point from either class. This provides a unique solution to the separating hyperplane problem. By maximizing the margin between the classes, it leads to better classification.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

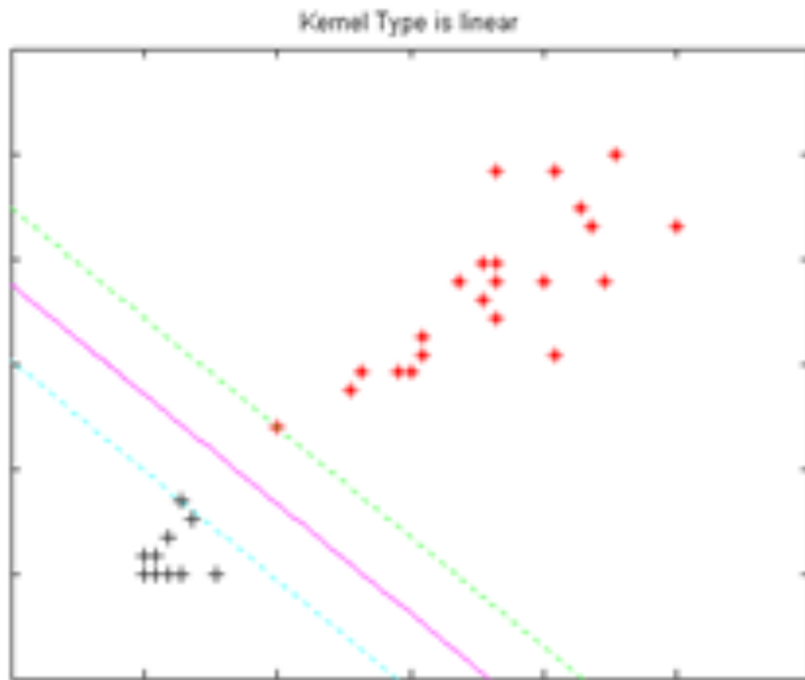
SVM and Maximal Margin illustrated-1



Separating Hyperplane (solid line) with Narrower Margin. Margin is the distance between the dotted lines

SMU
SINGAPORE MANAGEMENT UNIVERSITY

SVM and Maximal Margin Illustrated-2



Separating Hyperplane (solid magenta line) with Wider Margin. Margin is the distance between the dotted lines

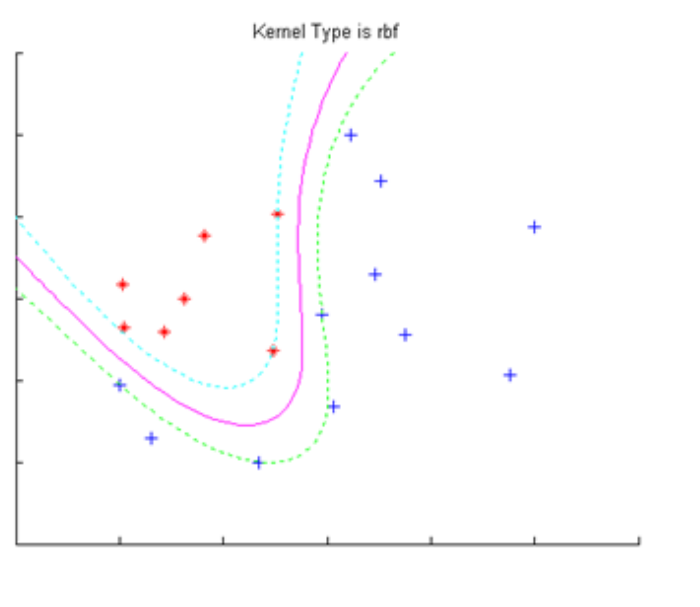
SMU
SINGAPORE MANAGEMENT UNIVERSITY

SVM and the Kernel Trick

- SVM's will use a non-linear function to map the training vectors or data points into a higher dimensional space.
- This higher dimensional space is called the Feature Space.
- We can then find and use a linear decision surface in the feature space, and this allows for non-linear separation in the original space.
- In our example, we used the mapping $\boldsymbol{\Psi}$ to map the two dimensional pattern space to a three dimensional feature space. $\Psi(u, v) \mapsto (u, v, uv)$

SMU
SINGAPORE MANAGEMENT UNIVERSITY

SVM and non-linear separation using kernel to map to higher dimension



Mapping to Higher Dimensional Feature Space Using RBFs Permits Non-Linear Separation.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

MATLAB code for SVM (previous example)

```
data=load('nonlin.txt')
x=data(:,1:2);
y=data(:,3);
kerneltype='rbf';
sigma=1;
[W0,b0,alpha]=swquad(x,y,kerneltype,sigma );
swPlot(x,y,'rbf',alpha, W0,b0 ,sigma);
hold on
pattern1= [-0.5 ,  -0.3];
scatter(pattern1(1),pattern1(2),401,'c.')

 pattern2= [0.3,  -0.8];
scatter(pattern2(1),pattern2(2),401,'k.')

test=[pattern1;pattern2];
classification=swSVMclassify(alpha,b0,x,y,test,kerneltype,sigma)
%%%%%%%%%%%%%%%%%%%%%%%%%
```

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Multi-class Classification by SVM's

- The two simple approaches are
  - One vs All (OVA) : Build K "one vs all" classifiers and choose the class which classifies the test datum with greatest margin. One vs Rest would have been a more appropriate name.
  - All vs All (AVA) : Build K(K-1)/2 pairwise binary classifiers and choose the class that is selected by the most classifiers. One vs One would have been a more fitting name.
  - AVA is often faster even though it has $O(\mathbf{N}^2)$ classifier
  - Another approach is a structural SVM which will not be described today.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Appendix 4:Informal Description of Hidden Markov Models

- Suppose we have a set of ***N*** urns, each with balls of ***M*** colours in different proportions. According to some random process, I randomly choose an urn and then select a ball from this urn with replacement. You get to see the colour of the ball but not which urn the ball comes from. Then, I choose another urn and select another ball. The process is repeated.

- This is a Hidden Markov Model. The urns are the "***hidden states***" and the colours of the balls are the "***observed signals***". A ***Markov chain*** governs the successive choices of the urns, i.e., the transition matrix of the Markov chain dictates the choice of the urns.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Informal Description of Hidden Markov Models (2)

The main characteristics of an HMM are

- The ***N*** hidden states.
- The ***M*** distinct types of observations, which are the colours of the balls in this example. We could have had continuous valued observations of course.
- The state transition probability matrix ***A*** = $\{a_{jk}\}$ giving the conditional probability that we are in state *k* at time (*t+1*) given that we were in state *j* at time *t*, i.e.,
  $a_{jk} = P[Q_{t+1} = S_k \mid Q_t = S_j]$,
  where S and Q are both used for denoting the states; $S_1, S_2, \ldots, S_N$ are the *N* states and Q_t is the state at time *t*.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Informal Description of Hidden Markov Models (3)

- The probability distribution of the observations, conditional on a state. In our example, this is the conditional probability of choosing a ball of a given colour, after the urn has been selected. $\boldsymbol{B_{jv}}$ = probability of observing a ball of *v-th* colour from the *j-th* urn.

- The initial state distribution **Π**, which is the probability distribution governing the initial choice of the states. The probability that the j-th urn is the first urn to be selected is Π_j.

- Together, these parameters are denoted by $\boldsymbol{\lambda}$ in the literature.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

The Three Basic Problems for HMM

- 2.1 **Evaluation Problem**. Given the parameters of an HMM, i.e. given ***λ**,* calculate the probability of realisation of a sequence of observations ***O***. (**Forward-Backward algorithms**)
  - That is, compute $P[O|\lambda]$
- 2.2 **Decoding or Classification Problem**. Given an observation sequence ***O***, find the sequence of hidden states most likely to have occurred. That is, compute
  - *argmax* P[***Q*/*O***]
  - *Q where* ***Q*** = $q_1, q_2, \ldots, q_T$ is the series of states indexed by time. (**Viterbi Algorithm**)
- 2.3 **Estimation or Training Problem.** Given a sequence of observations, fit the HMM. That is estimate **λ** *=* ***(A, B, Π)*** the parameters of the HMM, that maximize $P[O|\lambda]$. (**Baum-Welch Algorithm,** a type of EM algorithm**)**

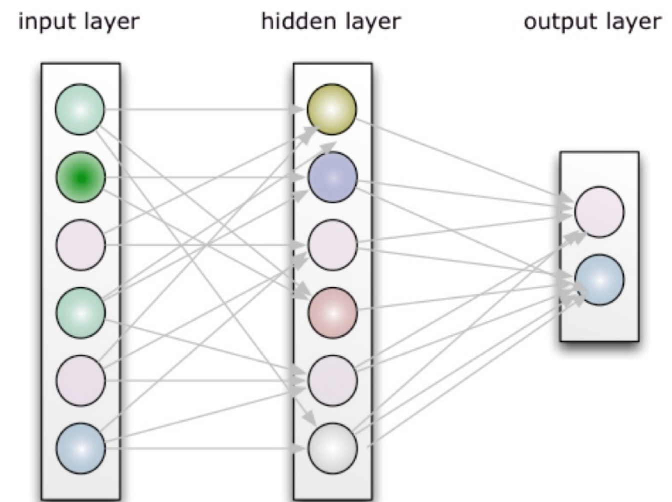
SMU
SINGAPORE MANAGEMENT UNIVERSITY

Appendix 5: What is a Neural Network (NN)?

- Traditional NN uses a feedforward network structure and usually has only one layer. Compared with Deep Neural Network, its structure is simpler and the training is less computationally intensive.

- NN is useful when we have abundance of labeled data but without the knowledge of the underlying mapping function that generates the output. It also shines when data sets are noisy or containing missing variables.

- To train a NN, we first acquire labeled inputs (as high-dimensional vector) and outputs. We then design the structure of the network, such as number of layers and number of neurons in each layer. The formal training process starts with random initialization and feedforward and backpropgation.



SMU
SINGAPORE MANAGEMENT UNIVERSITY

What is a Neural Network *REALLY* ?

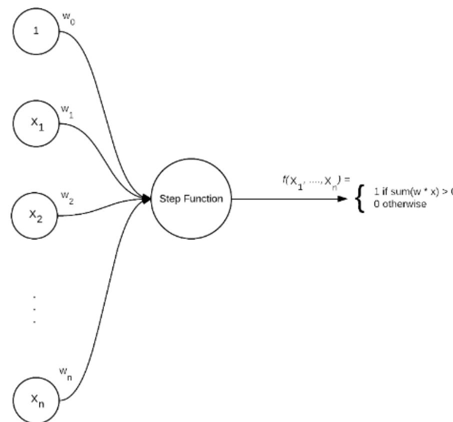
- There has been lot of hype surrounding Neural Networks, including exaggerated comparisons to human brain. This led to very high expectations which were not met, leading to disappointment with Neural Networks and AI for decades.

- Think of a Neural Network simply as a two stage, non-linear statistical model used for classification or regression.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

The Simplest Neural Network: Perceptron

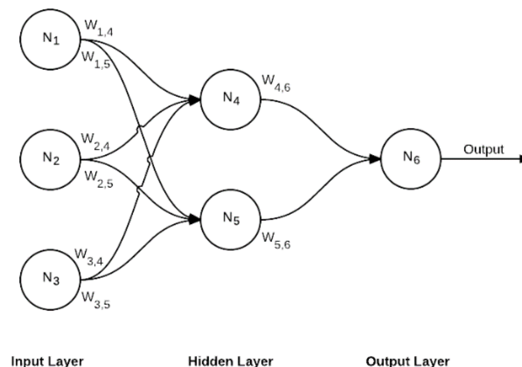
- The simplest neural network is described as a single hidden layer back-propagation network. There are *N* input nodes, one for each entry in the input feature vector, followed by only ***one layer*** in the network with just a ***single node*** in that layer. There exist connections and their corresponding weights, from the input 's to the single output node in the network. This node then takes the weighted sum of inputs and applies a *step function* to determine the output class label. The Perceptron outputs either a *0* or a *1* — *0* for class #1 and *1* for class #2; thus, in its original form, the Perceptron is simply a binary, two-class classifier. Perceptron is a *linear classifier*; it cannot solve non-linear problems such as XOR.



SMU
SINGAPORE MANAGEMENT UNIVERSITY

Multilayer Feed Forward Neural Network

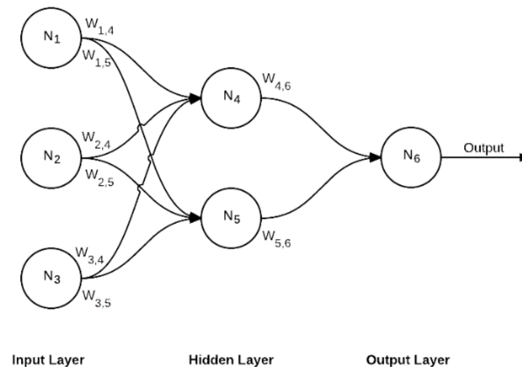
- In order to obtain non-linear separability, we can use ***multi-layer*** feedforward networks with ***non-linear activation functions***.
- A multi-layer feedforward network consists of multiple layers: 1 input layer, *N* hidden layers, and 1 output layer; in our figure we have one input layer, a hidden layer and an output



SMU
SINGAPORE MANAGEMENT UNIVERSITY

Neural Network and Softmax Regression

- In order to obtain non-linear separability, we can use ***multi-layer*** feedforward networks with ***non-linear activation functions.***
- The output of the hidden layer is fed to a Softmax function, as in a multinomial logistic regression function.



SMU SINGAPORE MANAGEMENT UNIVERSITY

Gradient Descent and Neural Network Training

- The bottom of the bowl is the minimum loss, or the best set of model parameters or in case of a neural network, the “weights
- The objective is to reach the bottom, taking as few steps as possible

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Stochastic Gradient Descent

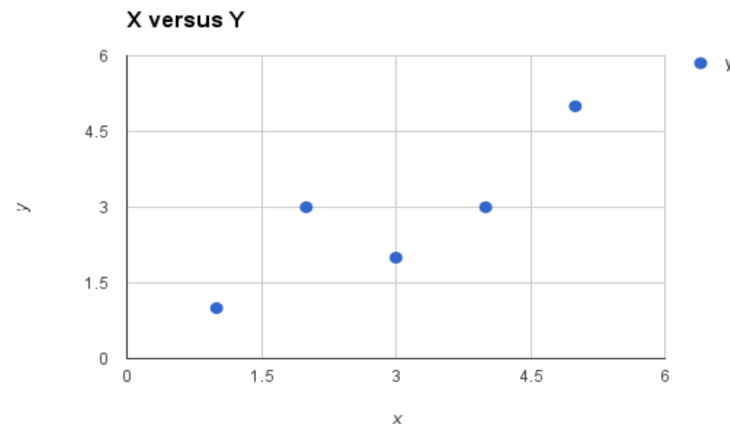
- In the original Gradient Descent scheme, all training examples are shown, the error calculated, and then we find gradients with respect to weights.
- This is called "batch" gradient descent but is very slow.
- Another approach is to show one example, compute error and gradients, update weights… this faster but noisy
- Stochastic Gradient – randomly choose a subset of the training examples (called "mini-batch") for each epoch. This is better than choosing one example each epoch

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Illustrating Gradient Descent for Training a Simple Model

- We can work through a simple example of how gradient descent can be used to train a linear regression model.
- Although the use Gradient Descent is not necessary here, it serves as an illustration
- Here are the data

x	y
1	1
2	3
4	3
3	2
5	5



SMU SINGAPORE MANAGEMENT UNIVERSITY

Illustrating Gradient Descent -2

- In a simple linear regression, the model is $y = b_0 + b_1 x$
- Initialise both model parameters to 0.
- The model becomes y = 0.0 + 0.0 * x
- Calculate the predicted value for y using our starting point coefficients for the first training instance: i=1, data(i) is x=1, y=1

- Prediction, p(i) = $b_0 + b_1 x(1) = 0$
- Error(i) = p(i)- y(i) = $b_0 + b_1 x(1) - 1 = 0 - 1 = -1$
- The loss function is $L = \frac{1}{2} * (\text{prediction - target})^2$
- Partial derivative of Error w.r.t. b_0 is 1 and w.r.t. b_1, it is x

SMU
SINGAPORE MANAGEMENT UNIVERSITY

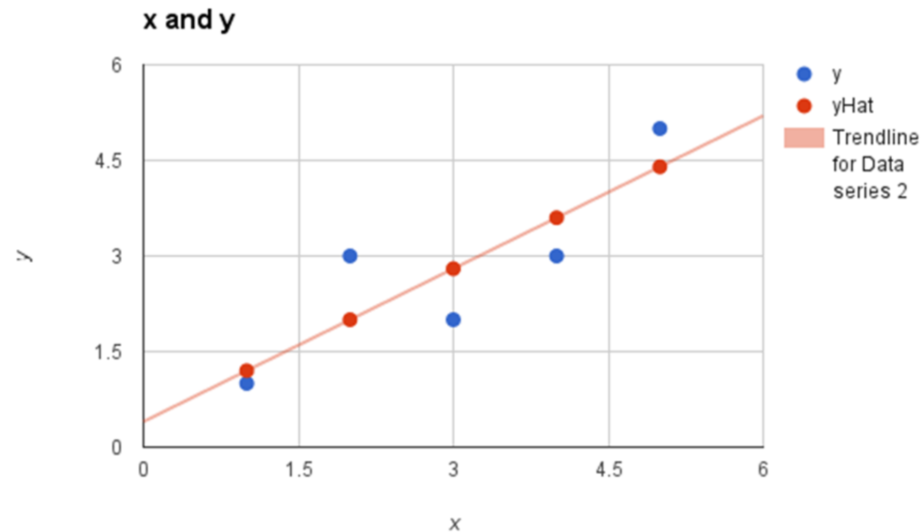
Illustrating Gradient Descent -3

- Keeping in mind what we just derived, and deciding to use a parameter α to control how big a step we want to take in the direction of the gradient, we see that
- $b_0(t+1) = b_0(t) – α * error$
- and
- $b_1(t+1) = b_1(t) – α * error * x$
- b_0(t+1) = 0.0 – 0.01 * -1.0 = 0.01
- b_1 (t+1) = 0.0 – 0.01 * -1 * 1=0.01
- We finished one epoch. After 20 iterations, we have b_0 as 0.23 and b_1 as 0.79 (actual values are 0.4 and 0.8)

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Example of Gradient Descent in Action -4

- When we use these parameters to make pass the values of x through the model and output the predicted values, here is what we get.

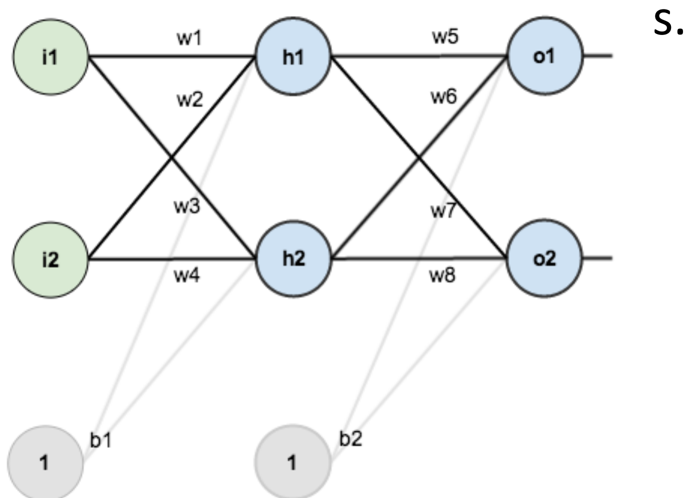


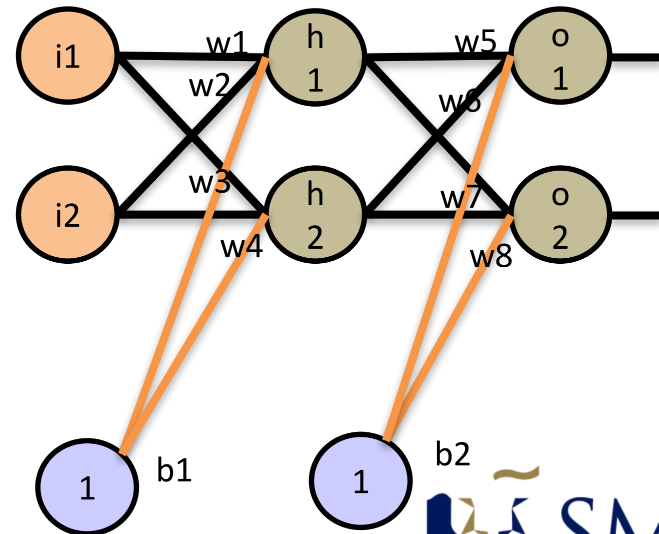
Appendix 6: Backpropagation Algorithm explained in-depth

- The backpropagation algorithm is a common method for training a neural network

- **Overview:** Let use a neural network with two inputs, two hidden neurons, two output neurons. Additionally, the hidden and output

s.





SMU
SINGAPORE MANAGEMENT UNIVERSITY

The Backpropagation Algorithm Illustrated-2

- In order to have some numbers to work with, here are the initial weights, the biases, and training inputs/outputs:

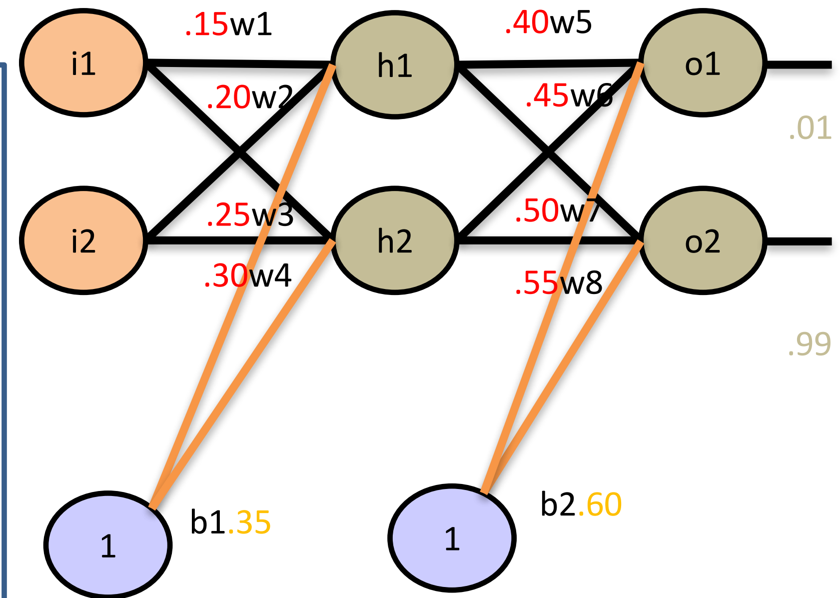
The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.

For the rest of this presentation we're going to work with a single training set:

- Given inputs 0.05 and 0.10
- We want the neural network to output 0.01 and 0.99.

The Forward Pass

- To do this we'll feed those inputs forward though the network.
- We figure out the *total net input* to each hidden layer neuron, *squash* the total net input using an *activation function* (here we use the *logistic function*),
- Repeat the process with the output layer neurons.



SMU
SINGAPORE MANAGEMENT UNIVERSITY

The Backpropagation Algorithm-The Forward Pass (Equation)

- Calculating the total net input for h_1

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15*0.05+0.2*0.1+0.35*1=0.3775$$

- We then squash it using the logistic function to get the output of h_1

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}} = \frac{1}{1+e^{-0.3775}} = 0.593269992$$

- Carrying out the same process for h_2

$$out_{h2} = 0.596884378$$

- We repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.
- Here's the output for o_1

$$net_{01} = 0.4*0.593269992+0.45*0.596884378+0.6*1=1.105905967$$

$$net_{h1} = w_5 * out_{h1} + w_6 * out_{h2} + b_2 * 1$$

$$out_{01} = \frac{1}{1+e^{-net_{01}}} = \frac{1}{1+e^{-1.105905967}} = 0.75136507$$

- And carrying out the same process for o_2 we get

out_{02} = 0.772928465

SMU SINGAPORE MANAGEMENT UNIVERSITY

Backpropagation: Calculating the Total Error-contd.

- we can now calculate the error for each output neuron using the squared error function and sum them to get the total error:
- $E_{total}=\sum \frac{1}{2}(target - output)^2$
- The is included so that exponent is cancelled when we differentitate later on. The result is eventually multiplied by a learning rate anyway so it doesn't matter that we introduce the constant here.
- For example, the target output for o_1 is 0.01but the neural network output 0.75136507, therefore its error is :
- $E_{o1}=\frac{1}{2}(target_{01}-out_{01})^2=\frac{1}{2}(0.01-0.75136507)^2=0.274811083$
- Repeating this process for o2 (remembering that the target is 0.99) we get
- $E_{o2}=0.023560026$
- The total error for the neural network is the sum of these errors:
- $E_{total}=E_{01}+E_{02}=0.274811083+0.023560026=0.298371109$

SMU SINGAPORE MANAGEMENT UNIVERSITY

The Backpropagation Algorithm :The Backwards Pass

- Goal with backpropogation is to update each of the weights in the network so that they cause the actual output to be the closer the target output.
- Thereby minimizing the for each output neuron and the network as a whole.

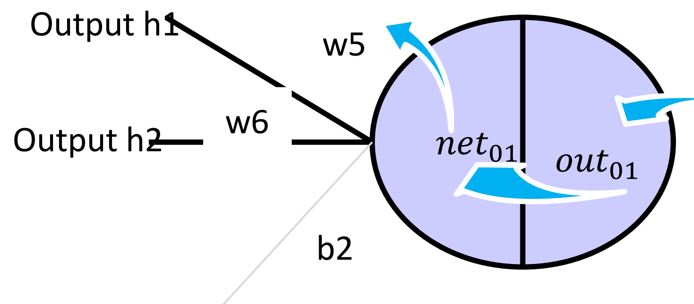
Output layer

- Consider w_5. How much a change in w_5 affects the total error aka $\frac{\partial E_{total}}{\partial w_5}$.
- $\frac{\partial E_{total}}{\partial w_5}$ is read as " the partial derivative of E_{total} with respect to w_5. You can also say that " the gradient with respect to w_5."

By applying the chain rule we know that:

$$\frac{\partial net_{01}}{\partial w_5} * \frac{\partial out_{01}}{\partial net_{01}} * \frac{\partial E_{total}}{\partial out_{01}} = \frac{\partial E_{total}}{\partial w_5}$$

- ➢ Visually, here's what we're doing



$$E_{01} = 1/2(target_{01} - out_{01})^2$$

$$E_{total} = E_{01} + E_{02}$$

SMU SINGAPORE MANAGEMENT UNIVERSITY

The Backwards Pass: Understanding the each piece of Equation

- First, how much does the total error change with respect to the output?
- $E_{total} = \frac{1}{2}(target_{01} - out_{01})^2 + \frac{1}{2}(target_{02} - out_{02})^2$
- $\frac{\partial E_{total}}{\partial out_{01}} = 2 * \frac{1}{2}(target_{01} - out_{01})^{2-1} * -1 + 0$
- $\frac{\partial E_{total}}{\partial out_{01}} = -(target_{01} - out_{01}) = -(0.01 - 0.75136507) = 0.74136507$
- -(target-out) is sometimes expressed as out-target
- When we take the partial derivative of the total error with respect to out_{01},the quantity $\frac{1}{2}(target_{02} - out_{02})^2$ becomes zero because out_{01} does not affect it which means we're taking the partial derivative of a constant which is zero.
- Next, how much does the output of o1 change with the respect to its total net output ?

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Understanding the each piece of Equation (contd..)

- The partial derivative of the logistic function is the output multiplied by 1 minus the output:
- $Out_{o1} = \frac{1}{1+e^{-neto1}}$
- $\frac{\partial out\ o\ 1}{\partial neto1}$= out_{o1} (1-out_{o1})=0.7513650(1-0.75136507)=0.186815602

- Finally how much does the total net input of change with respect to w_5?

SMU SINGAPORE MANAGEMENT UNIVERSITY

Understanding the each piece of Equation (contd..)

- Putting it all together
- $\frac{\partial E_{total}}{\partial w_5}$= 0.74136507*0.186815602*0.593269992=0.082167041
- You'll often see this calculation combined in the form of the delta form:
- $\frac{\partial E_{total}}{\partial w_5}$= -$(target_{01} - out_{01})$*$out_{01}$ (1-out_{01})*out_{h1}
- Alternatively, we have $\frac{\partial E_{total}}{\partial out_{01}}$ and $\frac{\partial out_{01}}{\partial net_{01}}$ which can be written as $\frac{\partial E_{total}}{\partial net_{01}}$ aka δ_{01}.
- (the Greek letter delta) aka the node delta. So we can use this to rewrite the calculation above:
- δ_{o1}=-($target_{o1}$-out_{o1})*out_{o1} (1-out_{o1})*out_{h1}
- Therefore,
- $\frac{\partial E_{total}}{\partial w_5}$=$\delta_{o1} out_{h1}$
- Some sources extract the negative sign from δ so it would be written as:
- $\frac{\partial E_{total}}{\partial w_5}$=-$\delta_{o1} out_{h1}$

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Understanding the each piece of Equation (contd..)

- To decrease the error we then subtract this value from this current weight (optionally multiplied by some learning rate data ,eta, which we'll set to 0.5):

$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$

- Some sources use **α** (alpha) to represent the learning rate, others use **η** (eta) and others even use **E** (epsilon).

w_6^+=0.408666816

w_7^+=0.511301270

w_8^+=0.0561370121

We perform the actual updates in the neural network after w have the new weights leading into the hidden layer neurons (i.e. we use the original weights, not the updated updates , when we continue the backpropogation algorithm below).

SMU
SINGAPORE MANAGEMENT UNIVERSITY

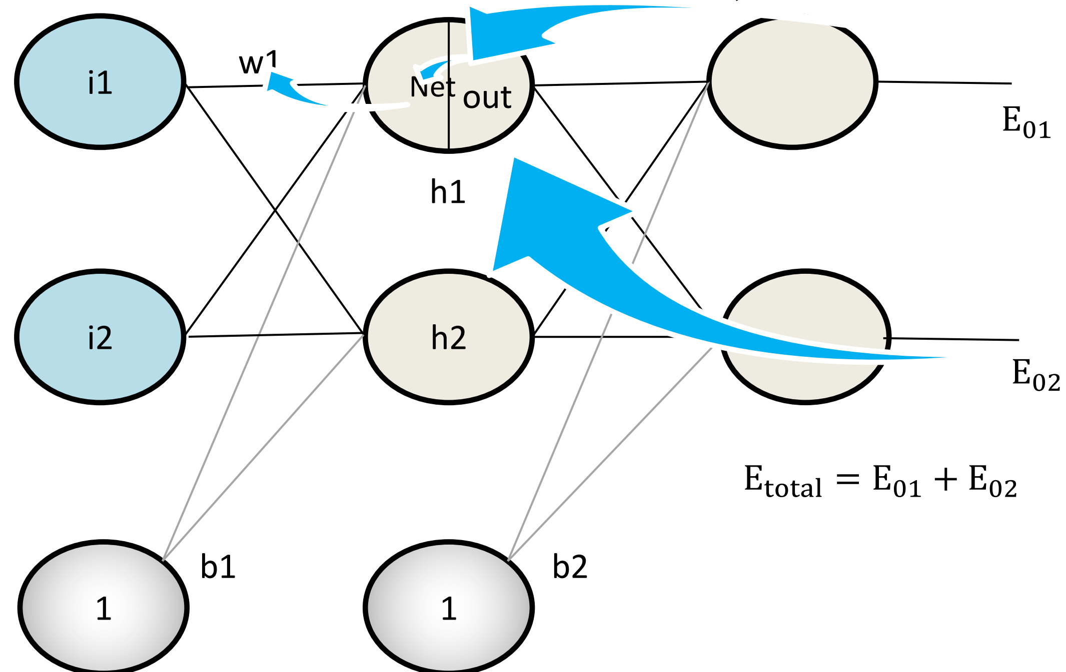
Understanding the each piece of Equation (contd..)

Hidden Layer

Next, we'll continue the backwards pass by calculating new valued for w_1,$w_{2,}$ w_3 and w_4.

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{01}}{\partial out_{h1}} + \frac{\partial E_{02}}{\partial out_{h1}}$$



SMU
SINGAPORE MANAGEMENT UNIVERSITY

Understanding the each piece of Equation (contd..)

We're going to use a similar process as we did for the outer layer but slightly different to account for the fact that the output each hidden layer neuron contributes to the output (therefore error)multiple output neurons. We know that out_{h1} affects both out_{01} and out_{02} and therefore the $\frac{\partial E_{total}}{\partial out_{h1}}$ needs to take into consideration its effects on the both output neurons:

- Starting with $\frac{\partial E_{o1}}{\partial out_{h1}}$:
- We can calculate $\frac{\partial E_{o1}}{\partial net_{h1}}$ using values we calculated earlier:
- And $\frac{\partial net_{o1}}{\partial out_{h1}}$ is equal to w_5:
- net_{01}=w_5*out_{h1}+w_6*out_{h2}+b_2*1
- $\frac{\partial net_{o1}}{\partial out_{h1}}$= w_5=0.40
- Plugging them in:
- Following the same process for $\frac{\partial E_{o2}}{\partial out_{o1}}$ we get
- $\frac{\partial E_{o2}}{\partial out_{h1}}$=0.019049119

SMU SINGAPORE MANAGEMENT UNIVERSITY

Understanding the each piece of Equation (contd..)

Therefore,

- Now that we have $\frac{\partial E_{total}}{\partial out_{h1}}$ we need to figure out $\frac{\partial out_{h1}}{\partial net_{h1}}$ and then $\frac{\partial net_{h1}}{\partial w}$ for each weight.
- $\frac{\partial out_{h1}}{\partial net}$=$out_{h1}$ (1-out_{h1}) $= 0.59326999(1-0.59326999)=0.241300709$
- We calculate the partial derivative oh the total net input h1 with the respect to w_1 the same as we did for the output neuron:
- $net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$
- $\frac{\partial net_{h1}}{\partial w_1}$=$i_1$=0.05
- Putting it all together
- $\frac{\partial E_{total}}{\partial w_1}$=0.036350306*0.241300709*0.05=0.000438568

SMU SINGAPORE MANAGEMENT UNIVERSITY

Understanding the each piece of Equation (Conclusion)

We can now update w_1:

- w_1^+=w_1-η*$\frac{\partial Etotal}{\partial w1}$=0.15-0.5=0.000438568=0.149780716
- Repeating this for w_2,w_3 and w_4:
- w_2^+=0.19956143
- w_3^+=0.24975114
- w_4^+=0.29950229
- Finally, we've updated all our weights!
- When we fed forward the 0.05and 0.1 inputs originally, the error on the network on the 0.298371109. after this first round of backpropogation, the total error is now down 0.2910227924. it might not seem like much, but after repeating this process for 10,000 times, for example, the error plummets 0.000035085. At this time when we feed forward 0.05and 0.1, the two outputs neurons generate 0.015912196(vs 0.01 target) and 0.984065734(vs 0.99 target).

SMU
SINGAPORE MANAGEMENT UNIVERSITY

The Backpropagation Algorithm-summary

A brief representation of the back propagation algorithm:

1: Initialize all weights w with random values

2: Until convergence:

- 2.1: For each feature vector x and expected output y:
  - 2.1.1: Pass x through the network and obtain the output.
  - 2.1.2: Calculate the error of the output nodes.
  - 2.1.3: Calculate the error at hidden nodes.
  - 2.1.4: Use the errors to compute Delta w_{i,j}
  - 2.1.5: Accumulate the errors for each Delta w_{i,j}
- 2.2: Perform learning by adding the accumulated Delta w_{i,j} to w_{i, j}

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Application of Hidden Markov Models

- The majority of the state-of-the art Automatic Speech Recognition systems employ HMM's. More recently, HMM's have been used in the prediction of gene sequences. We first describe the attempts to use HMM's to model phenomena other than Speech.
- HMM's have been used in modelling daily rainfall in a city. It is quite common to use the rainfall data for each day of the year going back several years to compile rainfall summary statistics like average rainfall and its variability. However, some questions regarding rainfall cannot be answered unless there is a stochastic model for the rainfall-generating process.
- E.g., we may ask the question "***What is the probability that there is more than X cm of rainfall in a given week and the longest dry run is no longer than k days***?"
- So to answer a question like the one we posed, one would fit a stochastic model to the time series, and then simulate generate different sample paths and get the quantities of interest using Monte Carlo methods. A good model will capture the serial relationship between successive observations.
- The time series of wet-dry days can often exhibit persistence or anti-persistence. After ***two dry days in Singapore, it is quite likely we get rain on the third day***. This type of modelling can have straightforward and useful parallels in the financial markets.
- Researchers have also fit HMM's to a time series of the Old Faithful geyser's ***waiting times between successive eruptions*** and duration of the eruptions.

SMU
SINGAPORE MANAGEMENT UNIVERSITY

END OF Bonus Material

- Thanks for your patience

SMU
SINGAPORE MANAGEMENT UNIVERSITY