Linear Algebra Basics

Vector Norm

For vector X $[X_1, \dots, X_n]$

$$|X| = \sqrt{\sum_{i=1}^{n} x_i^2}$$

Dot Product

For vectors X and Y
$X \cdot Y = \sum_{i=1}^{n} X_i Y_i = |X||Y| \cos(\theta)$,
Where θ is the angle between X and Y
So dot product is equal to 0 when $X \perp Y$

Plain Equation

$$X \cdot \Theta - \Theta_0 = 0$$

Projections of a on b

Vector

$$proj_b a = \frac{a \cdot b}{|b|^2} b$$

Scalar/Component

$$|proj_b a| = \frac{a \cdot b}{|b|}$$

The magnitude of the projected vector or distance from a point to a vector/hyperplane

Geometric Series

Finite

$$S_n = \sum_{i=1}^{n} ar^i = \frac{a(1-r^n)}{1-r}$$

Infinite

$$\sum_{i=1}^{\infty} ar^i = \frac{a}{1-r}$$

Classifiers

Training Error

$$\epsilon_n = \frac{1}{n} \sum_{i=1}^{n} [h(x^{(i)}) \neq y^{(i)}]$$

Where $h(x)$ is a classifier

Generalisation – how well the classifier generalises on test set

Hypothesis Space – the set of all possible hypothesis/classifiers/models

Linear Classifier

$$h(X; \theta + \theta_0) = sign(\theta \cdot X + \theta_0),$$

Decision Boundary Equation

$X \cdot \Theta + \Theta_0 = 0$,
So Θ is orthogonal to the decision boundary and it divides space into two parts if angle between X and θ is acute then we get positive result and vice versa

Linear Separation

If $y^{(i)}(\hat{\theta} X^{(i)} + \theta_0) > 0$, for all points then training examples are linearly separable

Perceptron Algorithm

$$\epsilon_n(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} [y^{(i)}(\theta X^{(i)} + \theta_0) \leq 0]$$

If $y^{(i)}(\theta X^{(i)} + \theta_0) \leq 0$, then
$\theta = \theta + y^{(i)} X^{(i)}$
$\theta_0 = \theta_0 + y^{(i)}$

Loss Function minimises the amount of misclassified points

Hinge Loss
if $y^{(i)}(\theta X^{(i)} + \theta_0) = z$
$\begin{cases} 0 \; if \; z \geq 1 \\ 1 - z \; if \; < 1 \end{cases}$

Signed Distance From Margin Boundary to Decision Boundary

$$\gamma(\theta, \theta_0) = \frac{y^{(i)}(\theta X^{(i)} + \theta_0)}{||\theta||}$$

Regularisation goal is to maximise $||\gamma(\theta, \theta_0)||$ the distance between margin and decision boundaries

Objective Function

$$J(\theta, \theta_0) = \frac{1}{n} \sum_{i=1}^{n} [y^{(i)}(\theta X^{(i)} + \theta_0) \leq 0] + \frac{\lambda}{2} ||\theta||^2$$

Gradient Descent

$\theta - \eta \nabla J(\theta)$
η -learning rate

Stochastic Gradient Descent

Instead of optimising the whole function at once we take some training points at random and optimise them

If loss is >0, then the gradient of the loss is $-y^{(i)} x^{(i)}$

Support Vector Machine

Doesn't allow for any point to be within the margin boundary, so it stretches the margin boundaries until a point is exactly on the margin boundary and others are outside, thus it finds the **maximum margin linear separator**.

Non-Linear Classifier

$h(x; \Theta; \theta_0) = \Theta\phi(x) + \theta_0$
where $\Theta\phi(x)$ is a **feature vector** that transforms original points

Kernel Function
$K(X,X') = \phi(X) \cdot \phi(X')$ an inner product of two feature vectors

Kernel Perceptron

$$\Theta \cdot \phi(X) = \sum_{j=1}^{n} y^{(i)} \alpha_j y^{(j)} K(X^{(j)}, X^{(i)})$$

Where α_j is the number of mistakes

Update

$\theta = \theta + y^{(i)} \phi(x^{(i)})$
$\alpha_i = \alpha_i + 1$

Kernel Composition Rules

1. $K(X, X') = 1$ is a kernel
2. Let $f(x)$ be a scalar function, then $f(X)K(X, X')f(X')$ is a kernel
3. If $K_1(X, X')$ and $K_2(X, X')$ are both kernels then their sum is a kernel
4. If $K_1(X, X')$ and $K_2(X, X')$ are both kernels then their product is a kernel

Radial Basis Kernel

$$K(X, X') = \exp\left(-\frac{1}{2} ||X - X'||^2\right)$$

General Formula For Number of Polynomial Terms

$$\binom{n}{1}\binom{n+1}{2} \dots \binom{n+p_i-1}{p_i}$$

Where p_i is the degree of a polynomial

Linear Regression

Empirical Risk

$$R_n(\theta) = \frac{1}{n} \sum_{t=1}^{n} (y^{(t)} - \Theta X^{(t)})^2$$

LR Mistakes

Structural linear mapping is simply not good enough
Estimation not enough training samples

Gradient Based Approach

Initialise at $\theta = 0$
Randomly pick t = {1,..,n}
$\theta = \theta + \eta(y^{(t)} - \theta^{(t)}) X^{(t)}$
$\eta_k = \frac{1}{1+k}$ possible way of defining learning rate

Closed Form Minimisation Solution

$b + A^{-1}\,\widehat{\Theta} = 0$

Ridge Regression

The full update

$\theta = (1 - \eta\lambda)\Theta + \eta\left(y^{(t)} - \Theta X^{(t)}\right)X^{(t)}$

Recommender Systems

K-Nearest Neighbours

$\widehat{Y_{ai}} = \frac{\sum_{b \in KNN(a)} sim(a,b)Y_{bi}}{\sum_{b \in KNN(a)} sim(a,b)}$

Where $sim(a,b)Y_{ai}$ is any distance function between the feature vectors X_a *and* X_b

Collaborative Filtering Algorithm

- Initialise V vector values randomly and multiply U^T
- Plug in the computed numbers into U matrix to calculate the Vs

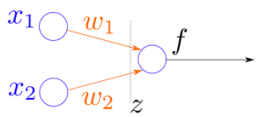
Alternating Minimasation

$$\frac{\partial}{\partial U}\sum_{(a,i)\in D}\frac{(Y_{ai} - U_a V_i)^2}{2} + \frac{\lambda}{2}\sum_a U_a$$

Where Y_{ai} is the value of the given matrix of other users, U *and* V are vectors such as UV^T have same dimensions as Y, U is the vector we are optimising and V is fixed

Neural Networks

Feedforward Neural Network



$z = X \cdot w + w_0$

f is the **activation/transfer function**

Hyperbolic Tangent Function

$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

Rectified Linear Function (ReLU)

$\mathrm{ReLU}(z) = \max(0, z)$

Deep Feedforward Neural Network also contains **hidden layers**

Recurring Neural Network

$S_t = \tanh\left(W^{s,s}s_{t-1} + W^{s,x}x_t\right)$

S_t – new state

x_t – new information

s_{t-1} – existing state

$W^{s,x}$ – weights/parameters taking into the account new information

$W^{s,s}$ – deciding what part of the previous information to keep

Difference between RNN and Feed Forward

- Input is received at each layer
- The number of layers varies depending on encoding
- Parameters of each layer are shared

Problem of RNN

The gradients can vanish or explode

Gating Network

$S_t = (1 - g_t)\odot S_{t-1} + g_t \tanh\left(W^{s,s}s_{t-1} + W^{s,x}x_t\right)$

Where $g_t = Sigmoid(W^{g,s}s_{t-1} + W^{g,x}x_t)$

$$Sigmoid = \frac{e^x}{e^x + 1}$$

Here first part of the equation retains the old information and the second is receiving the new one

Long Short Term Memory Network

$f_t = Sigmoid(W^{f,h}h_{t-1} + W^{f,x}x_t)$ – **Forget Gate**

$i_t = Sigmoid(W^{i,h} + W^{i,x}x_t)$ – **Input Gate**

$o_t = Sigmoid(W^{o,h} + W^{o,x}x_t)$ – **Output Gate**

$c_t = f_t \odot c_{t-1} + i_t \odot \tanh\left(W^{c,h} + W^{c,x}x_t\right)$ – **Memory Cell**

$h_t = o_t \tanh \odot (c_t)$ – **Visible State**

Markov Model

Prediction of the word is the product of the conditional probabilities

$P(w_1 \dots w_c) = P(w_1|w_0) \dots P(w_n|w_{n-1})$

$$\hat{p}(w'|w) = \frac{count(w,w')}{\sum_w count(w,\tilde{w})}$$

Softmax Function

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}, for\ i = \{1, \dots, k\}$$

Advantages of Feed Forward vs Markov Models

Extendable – we can easily add vectors representing k preceding words

We can look at **more complex** combinations of preceding words by introducing hidden layers

Markov model complexity of k-th order is $O(|v|^k)$ where *v* is the size of the vocabulary.

Feedforward network complexity $O((k-1)|v|^{k-1})$

RNN for Sequencing

We can use RNN to update probability distributions for of the states

So if $S_t = \tanh\left(W^{s,s}_{s_{t-1}} + W^{s,x}x_t\right)$, then probability distribution is $P_t = softmax(W^0 S_t)$

Convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(-\tau + t)\,d\tau$$

g' is called **convolution filter**

In order to apply convolution filter g' we need to flip g first and then apply it to f.

Cross-Correlation

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(\tau + t)\,d\tau$$

Same as convolution but without flipping **g**

Pooling

Separates what is in the image from where it is in the image. We take a patch and store its maximum value not to forget the location of the target object.

Image Quantization

Restricting the amount of colours we can use in the image

Clustering

Hard Clustering

Every element belongs strictly to one partition/cluster

Soft Clustering

Every point belongs to a every cluster just with different probabilities

Euclidian Distance

$dist\left(x^{(i)}, x^{(j)}\right) = \left|\left|x^{(i)} - x^{(j)}\right|\right|^2$

Cosine Similarity Measure

$$\cos(\theta) = \frac{X_a X_b}{||X_a||\ ||X_b||}$$

Unlike Euclidian distance it is insensitive to the size of the vectors

Cost Function

$$Cost(c_1 \ \dots \ c_k; z^{(1)} \dots \ z^{(k)})$$
$$= \sum_{j=1}^{k} \sum_{c \in c_j} \left|\left| x^c - z^j \right|\right|^2$$

In words a **cluster cost function** is the sum of the squared distance between the centre and all points in the cluster and total cost is the sum of all cluster costs

Clustering

K-means Clustering Algorithm

1. We randomly assign representatives and calculate distances of the point
2. Then we assign the points to each representatives according to distance
3. We move representatives in the right direction and repeat by using $z^j = \frac{\sum x_i}{|c_j|}$, where z_j is a new representative, C_j is the cluster cost.

K-means Limitations

- Depends heavily on the initialisation
- Can only work with the Euclidian square distance
- Manual choice of K
- Not robust to outliers
- Doesn't scale well with increasing number of dimensions

K-means Complexity

$O(nkd)$
n - number of points
k – number of clusters
d – dimensionality of a cluster

K-Medoids Algortihm

It is almost the same as the K-means, but we choose representatives from the existing points and we can use any function to measure the distance between points and representatives.

K-medoids Complexity
$O(n^2kd)$

We can check how far the point is from the centroids of all clusters, so we can use supervised learning to find optimal amount of clusters *K*. In general the number *K* is a trade off between the number of clusters and the size of the cost.

Mixture Models

Discriminative Model
Learns the boundary that separates each class

Generative Model
Model the probability distribution of each class

Multinomial Likelihood

$$p(D|\theta) = \prod_{i=1}^{n} \theta_{wi}$$
$$\log\left(\frac{P(y=+|D)}{P(y=-|D)}\right) = \sum count(w)\,\theta_w + \hat{\theta}_0$$
$$MLE = \frac{\sum count(w)}{\sum count(W)}$$

Multinomial Prediction

$\log\left(\frac{p(D|\theta^+)}{p(D|\theta^-)}\right)$ **Class conditional distribution**

Gaussian

$$\mu^{MLE} = \frac{\sum x^{(t)}}{n}$$
$$\sigma^{2\,MLE} = \frac{\sum \left|\left| x^{(t)} - \mu^{(t)} \right|\right|^2}{nd}$$

d – number of dimension

Expectation Maximisation Algorithm

Randomly initialize clusters, means and variances

E-step

$$p(\theta_j | x^{(i)}) = p(j|i) = \frac{p_j N(x^{(i)}, \mu^{(j)}, \sigma_j^2 I)}{p(x^{(i)}|\theta)}$$

Finding the likelihood of each point belonging to a certain cluster

M-step

$\widehat{n_j} = \sum_{i=1}^{n} p(j|i)$ – size of the cluster
$\hat{p}_j = \frac{\sum_{i=1}^{n} p(j|i)}{n}$ – mixture weight of cluster j
$$\hat{\mu}^{(j)} = \frac{\sum_{i=1}^{n} p(j|i) x^{(i)}}{\sum_{i=1}^{n} p(j|i)}$$
$$\widehat{\sigma^2}^{(j)} = \frac{\sum_{i=1}^{n} p(j|i) \left|\left| x^{(i)} - \widehat{\mu^{(j)}} \right|\right|^2}{d \sum_{i=1}^{n} p(j|i)}$$

Reinforcement Learning

Markov Decision Process

$MDP = < S, A, T, R >$
S= state
A = action
T= transition
R = reward

The Markov property transitions depend only on current state and actions and disregard the past.

Utility Function
Finite Horizon

$$U([S_0 \dots S_n]) = \sum_{i=1}^{n} R(S_i)$$

Infinite Discounted Reward UF

$$U([S_0, S_1, \dots]) = \sum_{t=0}^{\infty} \gamma^t R(S_t)$$

Where γ is the discount factor, so the reward in the future is less valuable than at present

Optimal policy
Action in a given state that maximises expected utility

Bellman Equations

Optimal value function
$V^*(s,a) = \max_a Q(s,a)$

Optimal policy function
$Q^*(s,a) = T(s,a,s')(R(s,a,s') + \gamma V^*(s'))$

Value Iteration Algorithm
Initialize $V_0^*(S) = 0$
Iterate until $V_k^*(s) \approx V_{k+1}^*(S)$, so when the value function converges
Compute V^* at every step {k,k-1…0}

Model Based Learning
Collecting probability estimates for all transition states

Model Free Learning
Estimating the probabilities along the way
$Q_{i+1}(s,a) = \alpha * sample + (1 - \alpha) Q_i(s,a)$
$sample = R(s,a,s') + \gamma \max_a Q(s',a')$

Exponential Running Average
$X_n = \alpha X_n + (1-\alpha)\,\bar{X}_{n-1}$

Q- Value Iteration by Sampling Algorithm
Initialize $Q(s,a) = 0$
Iterate until $Q_k^*(s,a) \approx Q_{k+1}^*(s,a)$, so when the Q-value function converges
Collect sample: s,a,s' and R(s,a,s')
$Q_{i+1}(s,a) = \alpha * sample + (1 - \alpha) Q_i(s,a)$

If
$Q_{i+1}(s,a) = Q_i(s,a) + \alpha(R(s,a,s') + \gamma \max_a Q_i(s',a') - Q_i(s,a))$,
Then algorithm guaranteed to converge

$\epsilon - greedy\ method$

ϵ
$- tells\ the\ likelihood\ of\ taking\ the\ action\ randomly$

Average Treatment Effect

$$ATE = \frac{1}{N_t}\sum_{i:W_i=1} Y_i^{obs} - \frac{1}{N_c}\sum_{i:W_i=0} Y_i^{obs}$$

Variance of the Treatment Effect

$$V(\hat{\tau}) = \frac{S_t^2}{N_t - 1} + \frac{S_c^2}{N_c - 1}$$

First Order Stochastic Dominance (FOSD)

A is FOSD over B if CDF of A is always smaller or equal to CDF of B and somewhere it should be smaller.

Linear Regression Extended

Ordinary Least Squares Assumptions

- Minimum Variance
- MLE under normality assumption
- Estimates are consistent and asymptotically normal

Coefficients

$\widehat{\beta_0} = \bar{Y} - \widehat{\beta_1}\,\bar{X}$

$E[\widehat{\beta_0}] = \beta_0$

$Var(\beta_0) = \frac{\sigma^2 \bar{X}^2}{n\,\widehat{\sigma_x^2}} + \frac{\sigma^2}{n}$

$\widehat{\beta_1} = \frac{1}{n}\frac{Cov(X,Y)}{Var(X)}$

$E[\widehat{\beta_1}] = \beta_1$

$Var(\beta_1) = \frac{\sigma^2}{n\,\widehat{\sigma_x^2}} - error\ variance$

$Cov(\beta_0, \beta_1) = -\frac{\sigma^2 \bar{X}}{n\,\widehat{\sigma_x^2}}$

$\widehat{\sigma^2} = \frac{\sum_{i=1}^{n} \widehat{\epsilon_i}^2}{n-2}$

ANOVA

Residual

$\epsilon = y_i - \widehat{y_i}$

Sum of Squared Residuals

$$SSR = \sum_{i=1}^{n} \widehat{\epsilon_i}^2$$

Total Sum of Squares

$$SST = \sum (Y_i - \bar{Y})^2$$

R-squared

$$0 \le R^2 = 1 - \frac{SSR}{SST} \le 1$$

Adjusted R-squared

$$\frac{n-k}{k-1}\frac{R^2}{1-R^2}$$

n – number of samples
k – number of parameters in β

Other Measure of Goodness of Fit

$$SSM = \sum \left(\hat{Y}_i - \bar{Y}\right)^2$$

F-Statistic

$$T = \frac{\frac{SSR - SSR_u}{r}}{\frac{SSR_u}{n-(k+1)}} \sim F_{r,n-(k+1)}$$

k – number of parameters in β
r – number of restrictions

Multivariate T-test

$$T = \frac{R\,\hat{\beta} - c}{SE(R\,\hat{\beta})}$$

$$SE\left(R\,\hat{\beta}\right) = (\sigma^2 R(X^T X)^{-1} R^T)^{\frac{1}{2}}$$

For one coefficient t-test = sqrt(F-test)

Neyman is more accurate in calculating small sample size **SE** than **OLS**

Dummy(Categorical/Indicator) Variables

Help us isolate certain effects in the regression, i.e. group is treated or not. They affect the intercept height of the line, if dummy variable is multiplied by continuous one, then it affects the slope of the line.

Synthetic Control Group Method

When you take the weighted average of value of groups similar to the one you study.

Series Regression

Method where we add polynomial terms until squared error is minimised

Piecewise Linear Regression

Splits continuous data of X into a number of bins and assigns a dummy variable to each of the bins.

Local Linear Regression

Same as piecewise linear regression but we also multiply dummies by continuous variables to create lines

Regression Discontinuity Design

Occurs when there is a discontinuous shift in our data connected to some other variable

Wald's Estimator

$$\hat{\beta} = \frac{E[Y_i|Z_i = 1] - E[Y_i|Z_i = 0]}{E[X_i|Z_i = 1] - E[X_i|Z_i = 0]}$$

Two stage Least Square

$\hat{\beta} = (Z'X)^{-1}Z'Y$

$Var(IV) = \sigma^2(Z'X)^{-1}Z'Z(Z'X)^{-1}$

Experiment Design

Endogeneity Problem

Occurs when dependent variable and explanatory variable affect each other both ways

Valid Instrumental Variable Conditions

- Randomly assigned
- Satisfies the exclusion restriction, i.e. No direct effect on Y
- Affects the endogenous regressor, i.e. affects X

Stratification

Split people into the groups(stratas) where individuals have the same traits and randomise within stratas

Clustering

Split people into groups and treat each group as a single observation

Phase-In

When you switch treatment and control groups during the progress of the experiment

Randomisation Around the Cutoff (Bubble)

Randomise people below and above the certain threshold

Encouragement Design

Introduces the variable that increases the probability that the individual is treated

SUTVA outcome of treatment of one person doesn't affect the others treated

Counterfactuals events that happen if the treatment measure isn't taken