NITHISH DIVAKAR

NOTES ON MACHINE LEARNING

Contents

Statistics

PROBABILITY DISTRIBUTIONS

Multivariate Normal Distribution

$$\mathrm{p}(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right]$$

Beta Distribution

Multivariate beta distribution is Dirichlet Distribution

$$\begin{aligned}\mathrm{p}(x;\alpha,\beta) &= Cx^{\alpha-1}(1-x)^{\beta-1}\\ &= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}x^{\alpha-1}(1-x)^{\beta-1}\end{aligned}$$

For positive integer n, $\Gamma(n) = (n-1)!$

Binomial Distribution

A random even have probability of success p. Binomial distribution models the number of success in n trials. Probability of getting k successes in n trials is given by

$$\begin{aligned}Pr(X=k) &= \binom{n}{k}p^k(1-p)^{n-k}\\ \mathbb{E}[X] &= np\end{aligned}$$

Multinomial Distribution

Multinomial Distribution $\mapsto$ think Histogram. The PMF is giving probability of a particular histogram given frequency of each bin.

If we have k possible mutually exclusive outcomes, with corresponding probabilities $p_1,\dots,p_k$, and n independent trials, it is modelled by multinomial distribution.

$$Pr(X_1 = x_1, \dots X_k = x_k) = \begin{cases} \dfrac{n!}{x_1! \cdots x_k!} p_1^{x_1} \times \cdots \times p_k^{x_k} & \sum x_i = n \\ = \dfrac{n!}{\prod x_i!} \prod p_i^{x_i} & \\ 0 & Otherwise \end{cases}$$

LAW OF THE UNCONSCIOUS STATISTICIAN

$$\mathbb{E}_X[g(X)] = \int_x g(x) f_X(x) dx$$
$$\mathbb{E}_X[g(X)] = \sum_x g(x) f_X(x) dx$$

MARGINAL, CONDITIONAL AND JOINT DISTRIBUTION

$$f_{X,Y}(x,y) = f_{Y|X}(y|x) f_X(x)$$

This can also be thought of as chain rule of probability. i.e. $P(A,B) = P(A|B)P(B)$

BAYES RULE

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$
$$posterior = \frac{likelihood \times prior}{marginal}$$

Law Total Probability

$$\mathrm{P}(B) = \sum_i \mathrm{P}(B|A_i)\,\mathrm{P}(A_i)$$

Conditional Probability

$$\mathrm{P}(B|A) = \frac{\mathrm{P}(B \cap A)}{\mathrm{P}(A)}$$

Naive Bayes Classifier

$$\mathrm{P}(class|data) = \frac{\mathrm{P}(data|class)\,\mathrm{P}(class)}{\mathrm{P}(data)}$$
$$\mathrm{P}(y|x) \propto \mathrm{P}(x|y)\,\mathrm{P}(y) = \mathrm{P}(y)\prod_i \mathrm{P}(x_i|y)$$

COVARIANCE MATRIX

Covariance Matrix gives covariances between variables of the population.

$$\Sigma_{ij} = \mathrm{cov}(X_i, X_j)$$

Sample Covariance matrix gives the estimates of covariances.

$$S_{ij} = \frac{1}{N-1}\sum_{i=1}^{N}(x_{ik} - \mu_k)(x_{ij} - \mu_j)$$

$$S = \frac{1}{N-1}\widetilde{X}^T\widetilde{X} \qquad \widetilde{X} = X - \mu$$
$$\mathrm{cov}(X,Y) = \mathbb{E}[(X-\mu_X)(Y-\mu_Y)] \quad \mathrm{Var}(Y) = \mathbb{E}[(Y-\mu_Y)^2]$$

$N-1$ is degrees of freedom. If you have 3 numbers with a mean μ and 2 numbers are a and b, then value of 3rd no is fixed.

CORRELATION COEFFICIENT

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X\sigma_Y} = \frac{\mathbb{E}\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X\sigma_Y}$$

$$\rho_{X,Y} \in (-1,1)$$

Covariance indicates the direction of the linear relationship between variables. Correlation measures both the strength and direction of the linear relationship between two variables.

Independence $\Rightarrow \rho_{X,Y} = 0$ but $\rho_{X,Y} = 0 \nRightarrow$ independence. ex $Y = X^2$

MAXIMUM LIKELIHOOD ESTIMATION

MLE is a procedure to compute parameters such that under the assumed model, the data is most likely.

$$\theta_{MLE} = \arg\max_{\theta} \mathrm{P}(D|\theta)$$

Procedure for computing MLE

◦ For each of the data samples, compute the likelihood function

$$L_i(\theta) = \mathrm{P}(X_i|\theta)$$

◦ Compute the MLE parameter that maximises likelihood

$$\theta_{MLE} = \arg\max_\theta \prod_n L_n(\theta)$$

◦ In practise, it is easier to work with logarithm of likelihood function or log likelihood. $\theta_{MLE} = \arg\max_\theta \sum_n \log L_n(\theta)$

MAXIMUM A POSTERIORI ESTIMATE

MAP gives the parameters of the model which best estimates (explains) the data.

$$\theta_{MAP} = \arg\max_\theta \mathrm{P}(\theta|D)$$

Assume that there is a prior distribution on parameters $p(\theta)$. Then we have

$$\mathrm{P}(\theta|D) = \frac{\mathrm{P}(D|\theta)p(\theta)}{\int \mathrm{P}(D|\epsilon)p(\epsilon)d\epsilon}$$
$$\boxed{\theta_{MAP} = \arg\max_\theta \mathrm{P}(D|\theta)p(\theta)}$$

MAP estimate the mode of posterior distribution. In context of Bayesian inference, MLE is a special case of MAP that assumes a uniform prior distribution of the parameters

CHAIN RULE OF PROBABILITY

$\mathrm{P}(A, B|C) = \mathrm{P}(A|B, C)\,\mathrm{P}(B|C)$

$$\begin{aligned}
\mathrm{P}(A, B) &= \mathrm{P}(A|B)\,\mathrm{P}(B) \\
\mathrm{P}(A, B, C) &= \mathrm{P}(A|B, C)\,\mathrm{P}(B|C)\,\mathrm{P}(C) \\
\mathrm{P}(x_1, \dots, x_n) &= \mathrm{P}(x_n|x_{1:n-1})\,\mathrm{P}(x_{1:n-1}) \\
&= \prod \mathrm{P}(x_i|x_{1:i-1})
\end{aligned}$$

Chain rule is useful when there are some conditional dependence/independence between variables. E.g. graphical models, HMMs etc.

HYPOTHESIS TESTING

In statistical hypothesis testing, we can never confirm a hypothesis. We can only negate or reject it based on lack of evidence. This implies that we can never confirm an empirically observed property of a system with certainty. We can only quantify its absence.

For testing a property of a system, we begin by setting up a null hypothesis. The null hypothesis (H_0) is the default hypothesis under which the property does not exist.

Let say that we have observed a few samples $\mathcal{D}$ from the system. If the null hypothesis were true, then these samples should follow some unknown true distribution π_{H_0}. Hypothesis testing is quantifying if $\mathcal{D}$ is sampled from π_{H_0}. i.e computing $\mathrm{P}(\mathcal{D} \sim \pi_{H_0})$.

Given a few samples, how can we quantify if it came from a distribution? If we had a i.i.d assumption on the samples, then we can use likelihood as a proxy. But we can't reliably make such assumptions without knowing more about the system. The solution is to use test statistics.

A test statistic or summary statistic is a function f which can aggregate a subset of samples T to a numerical value. The distribution of this value indicates how likely the sample came from some distribution.

f is a random variable.

$$\mathrm{P}(f(T)) = \mathrm{P}(T \sim \pi_{H_0})$$

Now we reliably quantify how likely the observed samples came from true distribution with $P(\mathcal{D} \sim \pi_{H_0}) = P(f(\mathcal{D}))$. But P is a p.d.f this probability is essentially 0. Instead, we can measure an equivalent quantity from the c.d.f. A p-value is probability of observing test statistic that is more extreme than the one observed from the observed sample.

$$p\text{-value}(\mathcal{D}) = \mathrm{P}(f(T) \geqslant f(\mathcal{D}) \,|\, T \sim \pi_{H_0})$$

If the p-value is lower than a predefined threshold or significance level (α), we can reject the null hypothesis citing lack of evidence.

We need 3 things to do hypothesis testing. A null hypothesis H_0, significance level α and test statistic f. The steps for hypothesis testing are.

Hypothesis Testing

- ○ Set up H_0 and H_A
- ○ Decide on significance level α
- ○ Collect data and calculate the test statistics
- ○ Calculate the p-value
- ○ If $p < \alpha$ reject H_0; otherwise fail to reject H_0

One and 2 tailed tests

$$H_A :\neq \implies \text{2 tailed test} \qquad H_A :\geqslant \implies \text{1 tailed test}$$

$$\begin{aligned}
&p = \mathrm{P}(T \geqslant t|H_0) \text{ one sided right tail test}\\
&p = \mathrm{P}(T \leqslant t|H_0) \text{ one sided left tail test}\\
&p = \mathrm{P}(|T| \geqslant |t|\ |H_0) \text{ if distribution is symmetric about 0}\\
&p = 2 \cdot \min\{\mathrm{P}(T \geqslant t|H_0), \mathrm{P}(T \geqslant t|H_0)\}\\
&\qquad \text{when distribution is not symmetric}
\end{aligned}$$

Test statistics is used to summarise data. It converts all the observation into a single scalar value.

Test	Typical use
Z-test	mean of normal distribution when variance is known
Student T-test	mean of normal distribution when variance is unknown
F-test	identify the model that best fits the population from which the data were sampled

	Test		
Z-Test	sample mean is μ	Variance is known	large sample
T-Test	sample mean is μ	Variance is not known	small sample
F-Test	variance of 2 samples is same or not	compare diff linear model trained on same data	
ANNOVA	is there significant difference in means of groups	Testing if an attribute is useful for classifying data	

s

Z-statistics

Z-statistics is used to test claims related to population means when population variance is known and number of samples are large ($n > 30$).

For example, if the population has mean μ and variance σ^2 and we have a sample $\{x_1, \dots x_n\}$ and we want to test if the sample was drawn from the population. We can calculate the z-statistics as

$$Z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

Through this claim, we are saying that both distribution are same and so, variance must be same (and hence known)

where $\bar{x} = \frac{1}{n}\sum x_i$ is the sample mean.

If $Z < t_\alpha$, where α is the significance level, we can reject null hypothesis or reject the claim that sample was drawn from the population.

If we are testing wether 2 samples of size n_1 and n_2 have the the same mean or not; the z-statistics is

$$Z = \frac{\bar{x} - \mu}{\sigma\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$$

T-statistics

T-statistics is used to test claims related to population mean when population variance is not known and number of samples are very less ($n < 30$).

For example, there is a claim that population mean is μ. To test the claim, we can gather some samples of the data $\{x_1, \dots, x_n\}$ and calculate t-statistics as

$$T = \frac{\bar{x} - \mu}{S/\sqrt{n}}$$

where $\bar{x} = \frac{1}{n}\sum x_i$ is the sample mean, $S^2 = \frac{1}{n-1}\sum(x_i - \bar{x})^2$ is Bessel-corrected sample variance.

Now if $T < t_{\alpha,\nu}$ we can reject the null hypothesis that population mean is μ. α is the level of significance and $\nu = n - 1$ is the degrees of freedom for t-statistics.

Here, our $H_0 : \bar{x} = \mu$ and we are doing a 2 tailed test. In case of $H_0 : \bar{x} \leqslant \mu$, we need to do right tailed test.

F-statistics

F-statistics are used to test if variances of two samples which are normally distributed are same or not.

$$F_{(n_A-1,n_B-1)} = \frac{S_A^2}{S_B^2}$$

Numerator has $n_A - 1$ degrees of freedom while denominator has $n_B - 1$

F-test can be used to compare fit of different linear models trained on same data. The sample variances corresponds to squared residual sum of each model. $RSS_i = \sum_i (y_j - f_i(x_j))^2$.

F-test can be used to test significance of a feature. We phase this as testing variances of two model, one model being intercept-only model whose prediction is average value of the feature and the other model being the feature itself. Here we are testing if the feature is able to reduce the variance. So $H_0 : \sigma_A^2 \geqslant \sigma_B^2$

Analysis of Variance (ANOVA)

ANOVA is used to test if there is a significance difference in means among groups. Lets say there are k groups. n_k is the size and $\bar{x}_i$ is the sample mean of group k.

The null hypothesis of ANOVA is that the means are equal i.e. $H_0 : \mu_1 = \mu_2 = \cdots \mu_k$. The test statistics is computed as

$$MS_{between} = \frac{1}{k-1} \sum n_j (\bar{x}_i - \bar{x})^2 \quad MS_{within} = \frac{1}{N-k} \sum_k \sum_i (x_{ik} - \bar{x}_k)^2$$

If difference in means is low and variance of all groups is large, its as if the groups are not different from each other and may even be part of same groups.

$$F_{k-1,N-k} = \frac{MS_{between}}{MS_{within}}$$

where $\bar{x}$ is overall sample mean and $N = \sum n_k$.

ANOVA assumes that the data is normally distributed and variance are homogeneous (same variance for all groups).

Use: Testing if an attribute is useful for classifying data. We can construct a group for each value of the attribute and compute F-statistics. Significance implies that the groups are distinguishable.

Chi-Square Test

If we have a categorical distribution with c classes and we know the expected frequencies E_i of each class, chi-square test allow us to test if a sample with observed frequencies O_i belongs to the same distribution.

The test statistics with $c - 1$ degrees of freedom is given by

$$\chi^2_{c-1} = \sum_i^c \frac{(O_i - E_i)^2}{E_i}$$

Chi-Squared test is used as a goodness-of-fit test to check if a sample fits a expected distribution. E.g. check if a dice is fair, check randomness of pseudo-random number generator. When used as a goodness-of-fit test, chi-squared test is a right tailed test.

Grubbs's test

Grubb's test is a statistical test to detect outliers in a univariate dataset assumed to come from a normally distributed population. It is also known as the maximum normalized residual test or extreme studentized deviate test.

Grubbs's test is defined for the hypothesis

H_0 : There are no outliers in the data set

H_A : There is exactly one outlier in the data set

Test statistics with $\bar{Y}$ as sample mean and s is sample deviation is given by

$$G = \max_i \frac{|Y_i - \bar{Y}|}{s}$$

This is for 2 tailed test. One tailed test is defined as a test if min/max value is an outlier.

$$G = \frac{|\bar{Y} - Y_{min}|}{s} \quad G = \frac{|Y_{max} - \bar{Y}|}{s}$$

Optimisation

UNCONSTRAINED OPTIMISATION

Gradient indicates the direction of largest change

$$\textit{Gradient} \qquad \nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix}$$

$$\textit{Hessian} \qquad \nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \dots & \frac{\partial^2 f(x)}{\partial x_1 x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n x_1} & \dots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix} = H(x)$$

Necessary and sufficient condition for local minima

A point x^* is said to be local minima iff $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semi-definite.

Matrix A is said to be p.s.d if $\forall x \in \mathbb{R}^n, x \neq 0, x^\intercal A x \geqslant 0$. If the last condition is a strict in-equality, then A is said to be positive definite

Newton's Method

To derive newton's method, we simply have to find the optimum point from second order taylor series expansion of $f(x)$

$$x_{k+1} = x_k - [H(x_k)]^{-1} \nabla f(x_k)^\intercal$$

Derivation: From a point x_k, we want to compute the best possible move $x_k + s$ to minimise f. Using taylor series expansion, we have

$$f(x_k + s) = f(x_k) + s\nabla f(x_k) + \frac{s^2}{2!} H(x_k) = g(s)$$

Since this is just a function of single variable, the optimum point is given by $\nabla_s g(s) = 0$

$$\begin{aligned} 0 = \nabla_s g(s) &= \nabla f(x_k) + sH(x_k) \\ s &= -H(x_k)^{-1}\nabla f(x_k)^\intercal \end{aligned}$$

Quasi Newton's Method

$$x_{k+1} = x_k - \alpha_k S_k \nabla f(x_k)^\intercal$$

If S_k is inverse of Hessian, then method is Newton's iteration; if $S_k = I$, then it is steepest descent

BFGS

B_k is an approximate to Hessian. The search direction p_k is determined by solving

$$B_k p_k = -\nabla f(x_k)$$

A line search is performed in this search direction to find next point x_{k+1} by minimising $f(x_k + \gamma p_k)$. The approximation to hessian is then updated as

$$\begin{aligned} B_{k+1} &= B_k + \alpha_k u_k u_k^\intercal + \beta_k v_k v_k^\intercal \\ u_k &= \nabla f(x_{k+1}) - \nabla f(x_k) \\ \alpha_k &= \frac{1}{\alpha u_k^\intercal p_k} \\ v_k &= B_k p_k \\ \beta_k &= \frac{-1}{p_k^\intercal B_k p_k} \end{aligned}$$

CONSTRAINED OPTIMISATION

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ s.t.\ &h_i(x) = 0 \\ &g_j(x) \leqslant 0 \end{aligned}$$

Karush-Kuhn-Tucker conditions

If the negative of the gradient has any component along an equality constraint $h(x) = 0$, then we can take small steps along this surface to reduce $f(x)$.

Since $\nabla h(x)$ the gradient of the equality constraint is always perpendicular to the constraint surface $h(x) = 0$, at optimum, $-\nabla f(x)$ should be either parallel or anti-parallel to $\nabla h(x)$

$$-\nabla f(x) = \mu \nabla h(x)$$

A similar argument can be made for inequality constraints. These form KKT conditions. So at an optimum point x^* we have,

$$\begin{aligned} h_i(x^*) &= 0 & g_j(x^*) &\leqslant 0 \\ \lambda_j g_j(x^*) &= 0 & \lambda_j &\geqslant 0 \end{aligned}$$

$$\nabla f(x^*) + \sum_i \mu_i \nabla h(x^*) + \sum_j \lambda_j \nabla g_j(x^*) = 0$$

Lagrange multipliers

The method of Lagrange multipliers relies on KKT conditions. For a constrained optimisation problem, we introduce a Lagrange function

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu^\intercal h(x) + \lambda^\intercal g(x)$$

Stationary points are points where derivative of the function is zero

The stationary points of Lagrange function satisfies all KKT conditions. Hence we can solve for $\nabla_x \mathcal{L} = 0$ to find the optimum point of $f(x)$. $\nabla_\mu \mathcal{L} = 0$ and $\nabla_\lambda \mathcal{L} = 0$ gives us the constraints.

SIMULATED ANNEALING

Simulated annealing is a gradient less optimisation which allows exploring the space in order to find a global optima. Each state has an energy $E(s)$.

SIMULATED ANNEALING

```
○ init $s = s_0$
○ For $step \in 1 \dots total$
    ○ $T = \text{temperature}\left(1 - \frac{step}{total}\right)$
    ○ $s^* = \text{neighbour}(s)$
    ○ If $\mathrm{P}(E(s), E(s^*), T) > \mathcal{U}(0,1)$
        ○ $s \leftarrow s^*$
```

The goal of simulated annealing is to find a state s such that the energy of the state $E(s)$ is minimised. But a simple gradient descent algorithm may get stuck in local minimal. SA overcomes this difficulty by encouraging explorations.

At every step, it is probabilistically decided weather to move to a neighbouring state s^* or not. The neighbouring states are produced by some conservative alteration of current state to encourage progressive improvement.

P is positive even when $E(s^*) > E(s)$. This prevent method from getting stuck in local minima. P $\rightarrow 0$ as $T \rightarrow 0$ to encourage only 'down-hill' moves towards the end of the procedure.

An example for $\mathrm{P}(e, e', T) = \begin{cases} 1 & e' < e \\ exp^{-(\frac{e'-e}{T})} & e' \geqslant e \end{cases}$

Annealing schedule. The algorithm starts with a high value of T which is gradually reduced at every step. It will end with $T = 0$ towards the end of allotted budget $(\frac{step}{total} \rightarrow 0)$

Simulated annealing can be combined with "restarts" where we can decide to go back to an older more optimum solution rather than continue exploration from current state. This is called Simulated annealing with restarts.

Machine Learning

PRIOR AND POSTERIOR DISTRIBUTION IN MACHINE LEARNING

In the context of machine learning, both prior and posterior distribution refers to **distribution of parameters**. Prior is the initial distribution of parameters and Posterior is the distribution of parameter when evidence (or data) becomes available.

Prior	$p(\theta)$
Posterior	$p(\theta\|D)$
Likelihood	$p(D\|\theta)$
Marginal	$p(D)$

REGULARISATION

L1 Regularisation

$$loss = error(f(x;\theta), y) + \|\theta\|_1$$

L2 Regularisation

$$loss = error(f(x;\theta), y) + \|\theta\|_2^2$$

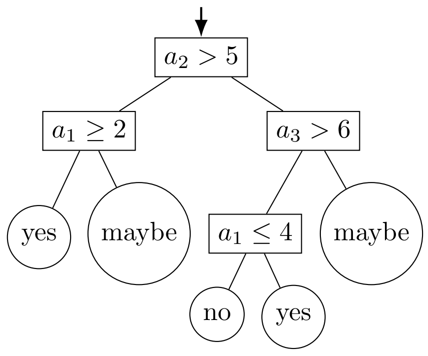
$$\frac{\partial\|\theta\|_2^2}{\partial\theta_i} = 2\theta_i \qquad \frac{\partial\|\theta\|_1}{\partial\theta_i} = \frac{|\theta_i|}{\theta_i} = \begin{cases} 1 & \theta_i > 0 \\ -1 & otherwise \end{cases}$$

Why does L1 regulariser induces sparsity?
Updates to the parameters during gradient descent are proportional to the gradient. For L2 regulariser, the magnitude of gradient diminishes as $\theta_i \to 0$ (gradient $= \theta_i$). But for L1 regulariser, the magnitde of gradient is constant. Hence for L2, its unlikely the values will ever become 0.

DECISION TREES

A decision tree is a tree based model which has 2 kinds of nodes.

- An **internal node** represents a test on an input attribute to split input set.
- A **leaf node** contains an outcome or prediction of the model



Construction: *Recursive Splitting*

- ◦ **1.** Select the best attribute and design a test on the attribute that partitions the data into disjoint subsets.
- ◦ **2.** For each outcome of the test, create a new child node with the subset
- ◦ **3.** Repeat for each child node until all samples in a node has same class/outcome or all attributes are used.

Decision trees tend to over-fit the data. They produce models which are **low bias** and **high variance**

Decision Tree Attribute Selection

Information Gain ↑

Information Gain is defined as amount of information gained about a random variable (outcome) from observing another (attribute). We can quantify information gain with difference in entropy when random variable is observed.

$$IG(T, A) = H(T) - H(T|A)$$

$$H(T) = -\sum_{c \in C} \mathrm{p}_c \log_2 \mathrm{p}_c$$

← entropy of T

$$H(T|A) = \sum_{a \in A} \mathrm{p}_a \, H(T_a)$$

← convex sum

The $\mathrm{p}_a = \frac{|T_a|}{|T|}$ or probability of finding a sample with attribute $A = a$.

$T_a = \{t \in T : t_A = a\}$

GINI Impurity ↓

GINI Impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of an attribute in the set.

Let say we partition the input set T according to the values of attribute A such that $T = \bigcup_{a \in A} T_a$. The split would be ideal if each of the partitions would have only a single class.

← different subset can have same class.

GINI impurity quantifies having multiple classes in same partition.

$$G(T_a) = \sum_{c \in C} \mathrm{p}_{a,c} \left(\textstyle\sum_{k \neq c} \mathrm{p}_{a,k} \right) = \sum_{c \in C} \mathrm{p}_{a,c}(1 - \mathrm{p}_{a,c}) = 1 - \sum_{c \in C} \mathrm{p}_{a,c}^2$$

← the attribute partition should have either all samples or none of it. Having only half samples implies it was a bad attribute to partition with.

Overall GINI Impurity score of partitioning T according to A is

$$G(A) = \sum_{a \in A} \mathrm{p}_a \, G(T_a) \quad = \sum_{a \in A} \mathrm{p}_a \left(1 - \sum_{c \in C} \mathrm{p}_{a,c}^2 \right)$$

p_a fraction of elements which has attribute a
$\mathrm{p}_{a,c}$ fraction of elements in class c and has attribute a

For continuous variable input, best decision threshold is calculated by computing IG/GINI at different levels (using binning or exhaustive search) of the range.

Variance Reduction

Variance reduction is used when target variable is continuous (tree is a regression tree). If the set T is being partitioned into T_L and T_R, the reduction in variance is given by

$$V(T) = \text{Var}(T) - \left(\frac{|T_L|}{|T|} \text{Var}(T_L) + \frac{|T_R|}{|T|} \text{Var}(T_R) \right)$$

$$\text{Var}(S) = \frac{1}{|S|^2} \sum_{i,j \in S} (y_i - y_j)^2$$

Computing variance without explicitly calculating mean

Random Forest

Decision Trees: **Low bias, high variance**
tends to over-fit data

While predicting, a single tree maybe highly sensitive to noise, but average of many trees are not if the **trees are uncorrelated**. Random forest is made by combining multitudes of trees trained on same data. The individual trees are trained using following techniques.

Reducing Variance

Bagging Select a subset of samples with replacement from training set. Bagging reduces variance while not effecting bias.

Feature Bagging At each candidate split, train the tree only on a subset of features. Without feature bagging, if a feature is strong predictor, it will get selected in many trees. This will result in correlation between trees.

← Also called random subspace method

In practise, a random forest will have $[100 - 1000]$ trees. The outputs of the trees are combined to arrive at the final prediction

- Continuous target $\mapsto$ average of predictions
- Categorical target $\mapsto$ majority vote

As a validation step, a part of training data is not used for training and the trees are evaluated on it. The error is called out-of-bag error.

BOOTSTRAP AGGREGATION OR BAGGING

Bagging is an ensemble procedure that reduces variance. It can be used with any method.

Given a training set D of size n, bagging creates many training sets each of size m by **uniform sampling with replacement** from D. Such a sample is called a bootstrap sample.

Typically $m = n$.

A model is trained on each of these training sets and during inference, the outputs from these models are combined (ensemble); averaging for regression and majority voting for classification.

advantages

- Reduces variance for learners which have low bias-high variance
- Ensemble of weak learners typically outperform a single learner trained on entire set

disadvantages

- Bagging will carry the bias if the original learner had high bias
- Computationally expensive

BOOSTING

Boosting is an ensembling algorithm which primarily reduces bias (and also variance). It combines many weak learners to make a strong learner.

The main idea of boosting is to learn a ensemble of trees whose output can be summed to get the final score.

$$F(x) = \sum_t f_t(x)$$

With $p_i \triangleq F(x_i)$, the objective function is

$$\arg\min_F l(y_i, p_i)$$

This objective includes functions as parameters and cannot be computed using traditional methods. Hence trees are learnt in a iterative (greedy) manner.

$$\mathcal{L}_t = \arg\min_{f_t} \sum_i l(y_i, p_i^{t-1} + f_t(x_i))$$

Boosting works by iteratively learning weak learners and adding it to the strong learner. After the latest weak learner is added, the data samples are **re-weighted** to reflect misclassified and correctly classified samples. The weights of misclassified samples are increased and correctly classified samples are diminished to allows future weak learner to concentrate on misclassified samples.

AdaBoost

Adaptive Boosting or AdaBoost is used for classification problems where the labels $y_i \in \{-1, 1\}$.

AdaBoost

```
○ $D = \{(x_i, y_i)\}$, $y_i \in \{-1, 1\}$, error function $E_i(f) = e^{-y_i f(x_i)}$
○ **initialise** $w_{1,1} \dots w_{n,1} \leftarrow \frac{1}{n}$
○ For $t \in 1 \dots T$
    ○ **select a weak learner** $f_t$ which minimises misclassification error rate $\epsilon_t$
        $$\epsilon_t = \sum_{i:f_t(x_i) \neq y_i} w_{t,i}$$
    ○ **add to ensemble**
        $$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$
        $$F_t = F_{t-1} + \alpha_t f_t$$
    ○ **reweighting** $w_{i,t+1} = w_{i,t} e^{-y_i \alpha_t f_t(x_i)}$ and normalise $w_{:,t+1}$ to 1
```

Gradient Boosting

The general framework of Boosting where learners are added in greedy manner is

$$F_t(x) = F_{t-1}(x) + f_t(x)$$

At the t^{th} step, we are interested in learning the function f which minimised the loss. The value of loss function at this point is given by

$$\begin{aligned} L &= l(y, p + f(x)) \\ &= l(y, p) + \nabla_p l(y, p) f(x) \end{aligned}$$

First order taylor series approximation.
$f(x) = f(a) + (x - a)f'(a)$

We have trying find f which minimises the loss L. So, we should move in negative gradient direction in function space.

$$F_t \leftarrow F_{t-1} - \gamma_t \frac{\partial L}{\partial f}$$
$$\frac{\partial L}{\partial f} = \nabla_p l(y, p) = -r$$

What this entails is at step t, we find the learner which best approximates the pseudo-residual r

Gradient Boosting

◦ **input** $D = \{(x_i, y_i)\}$, loss function $L(y, F(x))$
◦ **initialise** the model with a constant (typically mean $\mathbb{E}\, y_i$)

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$

◦ For $t \in 1 \dots T$
  ◦ **Compute pseudo residuals**

$$r_{it} = -\frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}$$

For MSE loss, $r_{it} = (y_i - F_{t-1}(x_i))$

  ◦ **Fit a weak learner on residuals**

$$f_t \leftrightsquigarrow \{(x_i, r_{it})\}$$

  ◦ **Compute multiplier** by solving the 1D optimisation problem

$$\gamma_t = \arg\min_{\gamma} L(y, F_{t-1}(x) + \gamma f_t(x))$$

  ◦ **Update model**

$$F_t = F_{t-1} + \gamma_t f_t$$

XGBoost or Extreme Gradient Boosting

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that follows the gradient boosting framework.

$$F_t(x) = F_{t-1}(x) + f_t(x)$$

One key difference is it uses second order taylor approximation for the loss function

$$\begin{aligned} L &= l(y, p + f(x)) \\ &= l(y,p) + \nabla_p l(y,p) f(x) + \nabla_p^2 l(y,p) \frac{f(x)^2}{2} \\ \frac{\partial L}{\partial f} &= \nabla_p l(y,p) + \nabla_p^2 l(y,p) f(x) = \nabla_p^2 \left(\frac{\nabla_p}{\nabla_p^2} + f(x) \right) \end{aligned}$$

XGBoost

- ◦ **input** $D = \{(x_i, y_i)\}$, loss function $L(y, F(x))$
- ◦ **initialise** the model with a constant (typically mean $\mathbb{E}\, y_i$)

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$

- ◦ For $t \in 1 \dots T$
  - ◦ **Compute gradients and hessians**

$$\nabla_F = \frac{\partial L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)} \quad \nabla_F^2 = \frac{\partial^2 L(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)^2}$$

  - ◦ **Fit a weak learner on residuals**

$$f_t \leftrightsquigarrow \left\{ \left(x_i, -\frac{\nabla_F}{\nabla_F^2} \right) \right\}$$

$$f_t = \arg\min_f \sum_i \frac{1}{2} \nabla_F^2 \left(-\frac{\nabla_F}{\nabla_F^2} - f(x_i) \right)^2$$

  - ◦ **Update model**

$$F_t = F_{t-1} + \gamma_t f_t$$

LINEAR REGRESSION

Regression problems are those where we have response variable y which depends on features x. Linear regression chooses to model this relationship as a linear transform.

affine transform to be exact

$$y = xW + b$$

where $\Theta = \{W, b\}$ are parameters of the linear regression model.

Linear Regression model aim to reduce the squared distance between its predictions and true values. If we think of the feature vector x as $(1, x_1, x_2, \dots x_n)$ then we can write the original transform as simply xW.

If X contains all the features in the dataset and Y has all the response variables stacked, we can rewrite linear regression as

$$\arg\min_W \frac{1}{2}\|Y - XW\|_2^2$$

Closed Form Solution

The error/loss incurred by the model is $l = \frac{1}{2}\|Y - XW\|_2^2$. If we take the partial derivative of the loss w.r.to the W and set it to zero, we get

$$\begin{aligned}\frac{\partial l}{\partial W} = 0 &\implies -(Y - XW)X = 0\\ &\implies XW = Y\\ W = (X^TX)^{-1}X^TY\end{aligned}$$

$(X^TX)^{-1}$ is called pseudo inverse of X

Gradient update form

Forward computation of the network is

$$l = \frac{1}{2}(Y_t - P_t)^2 \qquad\qquad P_t = X_{tj}W_j$$

Backward gradient of the network is

$$\begin{aligned}\frac{\partial l}{\partial P_a} &= P_a - Y_a \quad \frac{\partial P_a}{\partial W_b} = X_{ab}\\ \frac{\partial l}{\partial W_b} &= \frac{\partial l}{\partial P_a}\frac{\partial P_a}{\partial W_b} = (P_a - Y_a)X_{ab}\end{aligned}$$

```
P      = np.einsum('ab,b->a',X,W)
grad_W = np.einsum('a,ab->b',P-Y,X)
```

MLE for Linear Regression

Give a dataset $\{(x_i, y_i)\}$, we assume a the data is generated by $y = xW + \epsilon$. The noise in measurement ϵ is normally

MLE finds a model which maximises likelihood of data i.e $\arg\max_\theta \log L$

$$\mathcal{N}(y_i; x_iW, \Sigma) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(y_i - x_iW)^T\Sigma^{-1}(y_i - x_iW)\right)$$

$$\mathcal{L}(W; x, y) = \mathrm{p}(y|x; W)$$

distributed with zero mean and variance σ^2 in all dimensions. So we have,

$$\mathrm{p}(y|x, W) \sim \mathcal{N}(xW, \sigma^2 I)$$

Likelihood of *data*|*model* is

$$\begin{aligned}\mathcal{L}(W; x, y) &= \prod_i \mathcal{N}(y_i; x_i W, \Sigma)\\ \log \mathcal{L} &= -\frac{1}{2}\log\left((2\pi)^d \sigma^2\right) - \frac{1}{2\sigma^2}\sum_i (y_i - x_i W)^T (y_i - x_i W)\end{aligned}$$

Since $\sum_i a_i^2 = \|a\|_2^2$, we have

$$\arg\max_W \log \mathcal{L} \implies \arg\min_W \|Y - XW\|_2^2$$

←This is Linear Regression.

MAP for Linear Regression

We have a population which is generated by $y = xw + \epsilon$. The noise in measurement $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. We have a sample of the population $\mathcal{D} = \{(x_i, y_i)\}$.

Now, if we assume a prior distribution on the parameters $\mathrm{p}(w)$, the posterior probability is given by

$$\mathrm{p}(w|\mathcal{D}) = \frac{\mathrm{p}(w)\,\mathrm{p}(\mathcal{D}\,|w)}{\mathrm{p}(\mathcal{D})}$$

MAP estimation is computing $\arg\max_w \mathrm{p}(w|\mathcal{D})$. So, we have

$$\begin{aligned}\arg\max_w \mathrm{p}(w|\mathcal{D}) &= \arg\max_w \left[\log \mathrm{p}(w) + \log \mathrm{p}(\mathcal{D}\,|w) - \log \mathrm{p}(\mathcal{D})\right]\\ &= \arg\max_w \left[\log \mathrm{p}(w) + \log \mathrm{p}(\mathcal{D}\,|w)\right]\end{aligned}$$

Gaussian Prior Assuming a Gaussian prior for parameters in linear regression model would result in the parameter being mostly small values close to zero.

$$\begin{aligned}\mathrm{p}(w) &\sim \mathcal{N}(0, \lambda^{-1} I)\\ w^* &= \arg\min_w \frac{1}{2\sigma^2}\sum_i (y_i - x_i w)^T (y_i - x_i w) + \frac{\lambda}{2} w^T w\\ \mathbf{w_{MAP}} &= \arg\min_{\mathbf{w}} \|\mathbf{Y} - \mathbf{Xw}\|_2^2 + \mathbf{c}\|\mathbf{w}\|_2^2\end{aligned}$$

Least Squares with L2 regulariser or **Ridge regression**

Laplacian Prior $p(w) \sim Laplace(0, b)$ where

$$\begin{aligned} \text{p}(w) &\sim Laplace(\mu, b) = \frac{1}{2b}\exp\left(-\sum_i \frac{|x_i-\mu|}{b}\right) \\ w^* &= \arg\min_w \frac{1}{2\sigma^2}\sum_i (y_i - x_i w)^T(y_i - x_i w) + \frac{1}{b}\sum_i |w_i| \\ \mathbf{w_{MAP}} &= \arg\min_{\mathbf{w}} \|\mathbf{Y} - \mathbf{Xw}\|_\mathbf{1} + \mathbf{c}\|\mathbf{w}\|_\mathbf{1} \end{aligned}$$

Least Squares with L1 regulariser or **LASSO**

Hetroscedacity

Error The deviation of the observed value from the (unobservable) true value

Residual The difference between the observed value and the estimated (predicted) value

Scedastic function Conditional Variance is also called as scedastic function

Consider a regression equation $y_i = x_i\beta + \epsilon_i$. If the variance of the error term is a constant, $\text{Var}(\epsilon_i|X) = \sigma$, the data is said to be **homoscedastic**; Otherwise it is **hetroscedastic**. An example of hetroscedastic data is when variance is proportional to magnitude of observation. i.e $\text{Var}(\epsilon_i|X) = x_i\sigma$.

Example of hetroscedastic data might be a data which spreads proportional to value of x

Generalised Least Squares

Given sample form population in the form of $y = (y_1, \ldots, y_n)^\intercal$ and design matrix $X = [x_1, \ldots x_n]^\intercal$ where $x_i = (1, x_{i2}, \ldots x_{ik})$, GLS models conditional relationship $y|X$ as linear function of X.

$$y = X\beta + \epsilon \qquad \mathbb{E}(\epsilon|X) = 0 \qquad \text{Cov}(\epsilon|X) = \Omega$$

The residual of the model is $y - X\hat{\beta}$. The best estimate of β is computed by minimising Mahalanobis distance.

$$\begin{aligned} \hat{\beta} &= \arg\min_b (y - Xb)^\intercal\Omega^{-1}(y - Xb) \\ &= \arg\min_b y^\intercal\Omega^{-1}y + b^\intercal X^\intercal\Omega^{-1}Xb - 2b^\intercal X^\intercal\Omega^{-1}y \\ \hat{\beta} &= (X^\intercal\Omega^{-1}X)^{-1}X^\intercal\Omega^{-1}y \end{aligned}$$

Quadratic in b. Take gradient and equate to 0.

Converting GLS to OLS: GLS is equivalent to applying ordinary least squares to a linearly transformed version of the data. To see this, factor $\Omega = CC^\intercal$ and we get $\text{Cov}(\epsilon|C^{-1}X) =$ I.

$\Omega = CC^\intercal$ is Cholesky decomposition where C is Lower triangular matrix

Weighted Least Squares: When all off diagonal entries of Ω is zero. If diagonals of Ω are not the same value, the distribution is hetroscedastic .

R^2-coefficient

Coefficient of determination denotes proportion of variance in dependent variable that can be predicted by the independent variable.

For a linear regression model h,

$$R^2 = 1 - \sum \frac{y_i - p_i}{y_i - \bar{y}}$$

$$\begin{aligned} R^2 &= 1 - \frac{SS_{res}}{SR_{tot}} \\ SS_{res} &= \sum_i (y_i - h(x_i))^2 \\ SS_{tot} &= \sum_i (y_i - \bar{y})^2 \end{aligned}$$

For a base model which simply predicts $\bar{y}$, R^2-coefficient is 0. A model which is worse than predicting $\bar{y}$ would have negative score. An ideal model would get a score is 1.

The score can be treated as a measure of goodness of fit of the model.

LOGISTIC REGRESSION

Say we have a population $T = \{(x_i, y_i)\}$ where each sample belongs to 2 classes denoted by $y_i \in \{0, 1\}$. Let the probability of a sample being the true class be $p = \text{p}(y = 1|x)$.

Logistic regression assumes that log-odds is a linear function of features.

$$\log \frac{p}{1-p} = w^T x \quad \implies \quad p = \frac{e^{w^T x}}{1 + e^{w^T x}}$$

$$\text{p}(y = 1|x, w) = \frac{1}{1 + e^{-w^T x}} = \text{sigmoid}(w^T x)$$

MLE for Logistic Regression

Let $p_i \triangleq \mathrm{p}(y_i = 1|x_i, w) = \text{sigmoid}(w^T x_i)$

$$\begin{aligned}
\mathcal{L}(w) &= \prod_i p_i^{y_i}(1-p_i)^{1-y_i} \\
-\log \mathcal{L}(w) &= -\sum_i y_i \log(p_i) + (1-y_i)\log(1-p_i) \\
-\nabla_w \log \mathcal{L}(w) &= -\sum_i \frac{y_i}{p_i} p_i(1-p_i)x_i + \frac{1-y_i}{1-p_i}(1-p_i)p_i(-x_i) \\
&= -\sum_i (y_i - p_i)x_i \\
\nabla w &= (p-y)x
\end{aligned}$$

The likelihood for each sample is a Bernoulli distribution. $\mathrm{p}(x,y) = \prod_k \mathrm{p}(y=k|x)^{I[y=k]}$ Multinomial distribution

$\nabla\sigma = \sigma(1-\sigma)$, $1-\sigma(a) = \sigma(-a)$

This gradient can be used in gradient update equation which will minimise negative log likelihood. Note that negative log likelihood for logistic regression is cross entropy loss. Hence, logistic regression model minimises cross entropy loss between labels and predictions.

$$\min_w \sum_i \text{cross entropy}(y_i, w^\intercal x_i)$$

PRINCIPAL COMPONENT ANALYSIS

Principle Component Analysis seeks to find a linear transform which preserves variance of the data while reducing its dimensions.

Computing PCA

- Construct mean subtracted data matrix

$$X_{ji} = x_{ij} - u_j$$

- Compute partial SVD

$$X \approx U_k \Sigma_k V_k^T$$

- XV_k is the k dimensional projection of data with minimal reconstruction error

V has the right singular vectors of X or eigen vectors of $X^T X$

$$X = \begin{bmatrix} (x_1 - \mu)^T \\ \cdots \\ (x_m - \mu)^T \end{bmatrix}$$

Reconstruction Error of PCA

PCA has the least reconstruction error of all linear projections.

Let $\{v_1, \dots v_n\}$ be an orthonormal basis. Component of a data x along the basis vector v_j is $p_j = ({v_j}^\intercal x) v_j$

Since the basis is orthonormal and complete, we also have $x = \sum_j ({v_j}^\intercal x) v_j$. Total error of projecting data to only first k dimension to this orthonormal basis is given by

$$error = \sum_{i=1}^{m} \left\| x_i - \textstyle\sum_{j=1}^{k} (v_j^T x_i) v_j \right\|_2^2 \quad = \quad \sum_{i=1}^{m} \left\| \textstyle\sum_{j=k+1}^{m} (v_j^T x_i) v_j \right\|_2^2$$

To simplify this error term, we can use 2 identities. $\| \sum a_i \|_2^2 = (a_1 + \dots)^2 = \sum_{ij} {a_i}^\intercal a_j$ and ${v_p}^\intercal v_q = 0$ when $p \neq q$ and 1 otherwise.

$$error = \sum_{i=1}^{m} \left(\sum_{j=k+1}^{n} (v_j^T x_i)^2 \right) \quad = \quad \sum_{j=k+1}^{m} v_j^T (X^T X) v_j$$

This error is minimised when $\{v_{k+1}, \dots, v_n\}$ are the last $n-k$ eigen vectors of $X^T X$. Thus, the minimum error projection is computed by PCA.

Maximum variance of PCA

PCA maximises the variance of projected data.

Let assume we found a direction v to project data into. Variance of data point in this one dimensional projection is given by

Var$[x] = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$. In this case, we are assuming mean centered data. So second term is 0

$$\begin{aligned} \mathrm{Var}[v^\intercal x] &= \frac{1}{m} \sum_i (v^\intercal x_i)^2 = \frac{1}{m} \|Xv\|_2^2 \\ &= \frac{1}{m} v^\intercal X^\intercal X v \quad \propto \quad v^\intercal S v \end{aligned}$$

We need this variance to be maximised or $\max v^\intercal S v$, but then we could just select some arbitrarily large v. So to normalise all directions, we have

$$\arg\max_v v^\intercal S v \;\; s.t. \;\; \|v\|_2 = 1$$

[0]SVD of X, Eigenvalue Decomposition of $X^T X$ are all PCA

Since this is a constrained optimisation, we can write Lagrangian of the problem

$$L(v,\lambda) = -v^\intercal S v + \lambda(v^\intercal v - 1)$$

Which gives us

$$Sv = \lambda v$$

Thus v must be the eigen vector of S, moreover, since $\mathrm{Var}[vx] = v^\intercal S v = \lambda v^\intercal v = \lambda$, to maximise variance, λ must be the largest eigen value and v must be the corresponding eigen vector.

To find the second direction to project data point and maximise variance, we need a direction u such that $\mathrm{Var}[ux]$ is maximised and also, u must be orthogonal to v

$$\begin{aligned} &\arg\max_u u^\intercal S u \\ &s.t.\ u^\intercal u = 1; u^\intercal v = 0 \end{aligned}$$

This gives us that u is the second eigen vector of S. We can apply this argument iteratively and see that the first k dimensions are the first k eigen vectors of S. Hence of all linear projections to k dimensions, PCA maximises variance.

FISHER'S LINEAR DISCRIMINANT

FLD is used in dimensionality reduction when class labels are available. Projection found by PCA (direction of maximum variance) might not result in optimal separation of classes.

Given a data with K classes, FLD computes directions which will has large between class variance and small within class variance.

$$\begin{aligned} S_W &= \sum_k \sum_{x_n \in C_k} (x_n - \mu_k)(x_n - \mu_k)^\intercal \\ S_B &= \sum_k N_k(\mu_k - \mu)(\mu_k - \mu)^\intercal \\ S_B v &= \lambda S_W v \end{aligned}$$

Generalised Eigen value problem. S_W might not be invertible.

The direction of optimal projection are the eigen vectors of $S_W^{-1} S_B$. FLD can reduce dimensions to at most $K-1$ as maximum rank of S_B is $K-1$.

FLD fails when there are class means which coincide or the original data has large overlap between classes.

LDA

Model the generative process of the population as The samples have a Normal distribution within a class. i.e

$$\mathrm{p}(x|y) \sim \mathcal{N}(\mu_k, \Sigma_k)$$

LDA also makes **homoscedasticity** assumption ie. all class covariances are equal.

$$\Sigma_i = \Sigma$$

During Inference, it computes

$$\arg\max_k \mathrm{p}(y_k|x)$$

`WIP`

EXPECTATION MAXIMISATION ALGORITHM

EM algorithm is a technique to compute maximum likelihood estimate of parameters of a model with latent variables.

Compute posterior of latent variable given parameters and data

$$\theta_{MLE} = \arg\max_\theta L(\theta; X, Z)$$

But attempt to find analytical solution for such cases results in interlocking equations between parameters (θ) and latent variables (Z). Also, the marginal likelihood is often intractable

$$L(\theta; X) = P(X|\theta) = \int_Z P(X, Z|\theta)\, dZ$$

EM algorithm solves this situation by initially assuming some values for parameters and solving for latent variables. These computed values are used to get better estimates of parameters and the process is repeated until convergence.

Note: Since our aim is to estimate parameters, in practise, we don't estimate latent variable.

EM Algorithm

- ○ **Initialise** Assume values for parameter $\theta^{[0]}$
- ○ **Iterate**
  - ○ Given X and an estimate for $\theta = \theta^{[i]}$, we can compute Z

$$L(\theta; X, Z) = \mathrm{P}(X, Z|\theta)$$
$$Q(\theta|\theta^{[i]}) = \mathbb{E}_{Z|X,\theta^{[i]}}\left[\log L(\theta; X, Z)\right]$$

◦ Improve the estimate of θ

$$\theta^{[i+1]} = \arg\max Q(\theta|\theta^{[i]})$$

K-MEANS ALGORITHM

K-means algorithm assigns each data point to one of K clusters such that each point is closest to its cluster center. K-means also computes the cluster centers from the samples.

Clusters are represented by cluster centers $\{u_k\}$ and cluster assignment by binary variables r_{nk}. If x_n is assigned to cluster k, then $r_{nk} = 1$. The objective of the algorithm is to compute

$$\arg\min_{r,u} \sum_n \sum_k r_{nk} \|x_n - u_k\|^2$$

We can think of $\{u_k\}$ as parameter of the model and $\{r_{nk}\}$ as latent variables. This gives a EM algorithm formulation to compute the unknowns.

K-Means

◦ **Initialise** Assume values for cluster centers $\{u_k\}$
◦ **Iterate**
  ◦ **E-step** Assign points to cluster whose center is closest

$$\boldsymbol{r_{nk}} = \begin{cases} \mathbf{1} & \boldsymbol{k} = \mathbf{arg\,min}_{\boldsymbol{j}} \|\boldsymbol{x_n} - \boldsymbol{u_j}\|^{\mathbf{2}} \\ \mathbf{0} & \text{Otherwise} \end{cases}$$

  ◦ **M-step** Improve the estimate of $\{u_k\}$ by setting it as center of all points in corresponding cluster.

$$\boldsymbol{u_k} = \frac{\sum \boldsymbol{r_{nk} x_n}}{\sum \boldsymbol{r_{nk}}}$$

Formula for u_k is derived by setting $\nabla_{u_k} J = 0$

GAUSSIAN MIXTURE MODELS

GMMs are density estimation model which uses mixture of Gaussians to represent the data density.

$$\mathrm{p}(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k)$$

[0]K-means is equivalent to EM algorithm with $\{u_k\}$ as parameters and r_{nk} as latent variables.

Lets assume a latent variable z which indicates which mixture component is responsible for a sample x. If we use one-hot representation for z, then $z_k \in \{0,1\}$ and $\sum_k z_k = 1$. The marginalised density function is given by

$$\mathrm{p}(x) = \sum_{k=1}^{K} \mathrm{p}(z_k = 1)\,\mathrm{p}(x|z_k = 1)$$

$\mathrm{p}(x) = \sum_z \mathrm{p}(x, z)$

Since $\mathrm{p}(x|z_k = 1) = \mathcal{N}(\mu_k, \Sigma_k)$, we can see that $\mathrm{p}(z_k = 1) = \pi_k$. Also,

$\mathrm{p}(z_k|x_n) = \frac{\mathrm{p}(x_n|z_k)\,\mathrm{p}(z_k)}{\sum_t \mathrm{p}(x_n|z_t)\,\mathrm{p}(z_t)}$

$$\mathrm{p}(z_k = 1|x_n) = \boxed{r_{nk} \triangleq \frac{\pi_k p(x_n|\mu_k, \Sigma_k)}{\sum_t \pi_t\,\mathrm{p}(x_n|\mu_t, \Sigma_t)}}$$

GMM Algorithm

- **Initialise** Assume values for parameters $\{\mu_k, \Sigma_k, \pi_k\}$
- **Iterate**
  - **E-step** Compute r_{nk} with current values of $\{\pi_k, \mu_k, \Sigma_k\}$
  - **M-step** Improve the estimate of parameter.

$$\mu_k = \frac{1}{N_k}\sum_n r_{nk}x_n$$

$$\Sigma_k = \frac{1}{N_k}\sum_n r_{nk}(x_n - \mu_k)(x_n - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N}\ ;\ N_k = \sum_n r_{nk}$$

Convergence is tested by evaluating log likelihood

NAIVE BAYES CLASSIFIER

Naive Bayes assumption: Given class labels, features are conditionally independent. i.e $x_i \perp x_j|y$

Bayes Rule gives us

$$\mathrm{p}(y|x) \propto \mathrm{p}(y)\,\mathrm{p}(x|y)$$

Naive bayes assumption allows us to write class conditional density as

$$\mathrm{p}(x|y_k) = \prod_i \mathrm{p}(x_i|y_k)$$

[0]Equation of $\{\pi_k, \mu_k, \Sigma_k\}$ are derived by setting $\nabla_{\square} \log p(X|\theta) = 0$

Applying Bayes rule we get $\mathrm{p}(y_k|x) \propto \mathrm{p}(y_k)\prod_i \mathrm{p}(x_i|y_k)$. The quantities $\mathrm{p}(y_k)$ and parameters of $\mathrm{p}(x_i|y_k)$ can be estimated from data. To classify a new sample, simply compute

$$k = \arg\max_k \mathrm{p}(y_k) \prod_i \mathrm{p}(x_i|y_k)$$

Bernoulli Naive Bayes When the features of the samples are only binary, i.e $x_i \in \{0,1\}$. We can define $p_{ki} \triangleq \mathrm{p}(x_i = 1|y = y_k)$ is the frequency of $x_k = 1$ in class y_k. Then $1 - p_{ki} \triangleq \mathrm{p}(x_i = 0|y = y_k)$. This gives us

$$\mathrm{p}(x_i|y_k) = (p_{ki})^{x_i}(1 - p_{ki})^{(1-x_i)}$$

Gaussian Naive Bayes When x_i is has continuous value we assume it has a normal distribution within each class. We estimate the mean μ_{ik} and variance σ_{ik}^2 of the feature for that class. During inference, when we observe x_i to have value v_i, we compute

$$\mathrm{p}(x_i = v_i|y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} \exp\left[-\frac{(v_i - \mu_{ik})^2}{2\sigma_{ik}^2}\right]$$

Multinomial Naive Bayes In this case, the feature is occurrence of some N different events a total of M times. x_i denotes counts of event i. $x_i \in \{0, 1, \dots M\}$ and $\sum_i^N x_i = M$. $\mathrm{p}(x|y_k) = [\dots, \mathrm{p}(x_i|y_k), \dots]$ is the normalised histogram representing a multinomial distribution. We can estimate the quantities p_{ki} as frequency of occurrence of event i in class k. During inference for classifying a sample x, we compute

Think of a document with different word. event i is word i

$$\mathrm{p}(x|y_k) = \frac{(\sum x_i)!}{\prod x_i!} \prod p_{ki}^{x_i}$$

$\log p(y_k|x)$ evaluates to a $wx + b$ form $\implies$ linear classifier with word count frequencies.

RECOMMENDER SYSTEMS

Utility matrix consists of the rating (or preference) for each user-item pair

Content-based filtering

Content-based recommendations finds items which are similar to highly rated items by customers. Content based filtering works as follows.

Item Profile First step is to build a profile of the items, which is a vector representing the features of the item. The features can be important words of a document (tf-idf), or properties of a movie like genre or actors.

User profile A user profile is built from the item profiles of the items the user has rated (explicitly or implicitly). A user profile is some sort of aggregation of all the item profiles and gives a likeness preference of the user.

Recommending similar Item Given profile of a user x and and items profile z, the affinity of the user towards item can be calculated using **cosine similarity** between x and z. Items with high similarity can be recommended.

Content based filtering does not suffer from cold start problem on the item side. But the main difficulty is in finding the right features for building item profiles.

It does however suffer from *cold start* on the users. Its hard to build a profiles of a user who has not rated anything.

Collaborative Filtering

Collaborative filtering finds users whose ratings of items are similar. Then it recommends an item rated highly by a user to the other who has not rated it. The main issue in finding similar users from their ratings of items is that the utility matrix is very sparse. Following measures are used to measure similarity between users.

Jaccard Distance completely ignores the rating value of the items. It computes IoU scores where I is the set of all items both users have rated and U is set of all items at least one of them have rated.

$$J = \frac{|I|}{|U|}$$

Cosine Similarity Simply find the cosine distance between the vector of ratings by both users. Cosine similarity treats missing values as 0. This results in treating no rating as strong dislike.

Pearson Correlation coefficient is computed only between items both users have rated.

For computing rating of an unrated item z by user x, collaborative filtering find n users who is similar to the x and averages their ratings for z. In the averaging, only those rating are considered which are not missing (i.e remove 0's).

For handling the cases where different users differ in their general rating strategy (giving all high ratings of low ratings), the averaging step can be modified to first subtract the average of a user's rating from the rating for z before being aggregated. The value is then added to average rating of user x for arrive at ratings for z by x.

User-User and Item-Item The utility matrix has a symmetry w.r.to the similarity. Similar to computing similarity of 2 users, we can compute similarity of 2 items also from ratings. The item similarity can be used to reduce the problem of having only sparse ratings. User rating for an item can be thought of as user rating for the entire cluster of similar items.

Collaborative filtering also sufferers from cold start problems. New item have no ratings and new users have no history.

Latent Factor Models

We can model the utility matrix as a product of 2 thin long matrices to have a low rank approximation. This is UV decomposition.

$$M = UV$$

$$\arg\min_{U,V} \|M - UV\|_2$$

We can use gradient descent to compute the best U, V for the problem. We can also use Stochastic gradient descent which uses only a small batch of data at a time.

However, directly using this results in bias due to sparseness of M. A huge number of entries are treated as 0 where the ratings which are simply unavailable. So we rewrite the problem as.

$$\arg\min_{u,v} \sum_{ij \in S} (m_{ij} - {u_i}^\intercal v_j)^2$$

Can use l2 regularisation on u and v

The utility matrix also suffers from ratings distribution of users. So before decomposition, the ratings are normalised by subtracting average of a user ratings from all the user's ratings. Latent Factor models also have cold start problem.

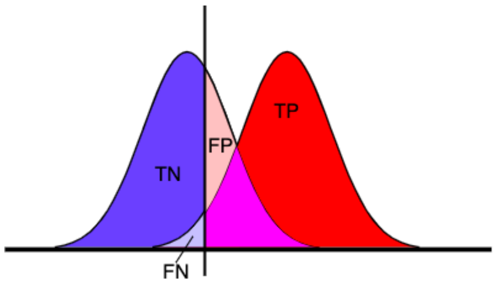
CROSS VALIDATION

Cross validation is a technique for assessing how much a model generalises is on unknown dataset. The procedure involves splitting the dataset into complementary subsets, training the model on on of the set(train set) and evaluating performance on the other (validation set)

Leave-one-out Cross Validation The data is partitioned. One of the partition is selected and the model is trained all the other partitions combined. The model is then evaluated on the selected partition. The process is repeated for all the partition iteratively and results is combined.

MEASURES

Precision and Recall



- Precision $= \dfrac{\text{correct predictions}}{\text{outputs}} = \dfrac{TP}{TP + FP}$
- Recall $= \dfrac{\text{correct predictions}}{\text{true samples}} = \dfrac{TP}{TP + FN}$

*ROC and AUC

Receiver Operating Characteristics is a plot between false positive rate and true positive rate.

TPR and recall are the same

$$TPR = \frac{TP}{P} \qquad\qquad FPR = \frac{FP}{N}$$

TPR can be thought of fraction of samples in output which are *correctly* marked as True

FPR can be thought of fraction of samples in output which are *in-correctly* marked as True

AUC stands for Area Under the Curve and it is the total area under ROC curve. AUC represent the probability of the model to rank a randomly chosen positive higher than a randomly chosen negative.

*PR curve

Plot between precision and recall.

*Macro and micro precision

A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric

F-measure

F-measure is a family of measure which parameterise relative importance to recall over precision. It is defined as

$$F_\beta = (1+\beta^2)\frac{\text{precision} \cdot \text{recall}}{\beta \cdot \text{precision} + \text{recall}}$$

LEARNING TO RANK

Approaches

Point wise approach Each query-document has an ordinal score. The problem is regression problem which requires the score to be predicted.

Pairwise approach In this approach, the problem is treated as a binary classification problem which given 2 document (A, B), predicts which one is more relevant.

List-wise approach Tries to find the best order of a list of document based on relevance.

Measures

DCG Discounted cumulative gain measures the usefulness, or gain, of a document based on its position in the result list.

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}$$

Value of relevance can be binary $rel_i \in \{0, 1\}$ or any other real numbers. Binary is more popular.

NDCG Normalised DCG. Comparing different search queries cannot be accommodated in DGC because length of search queries may vary. So the scores must be normalised by sorting all possible document in the corpus by their relevance to obtain max possible DCG with p positions (Ideal DCG).

$$nDCG_p = \frac{DCG_p}{IDCG_p}$$
$$IDCG_p = \sum_{i=1}^{|REL_p|} \frac{2^{rel_i=1} - 1}{\log_2(i+1)}$$

REL_p represents list of relevant document up to position p in the corpus. $nDCG_p \in [0, 1]$.

nDCG is harder to calculate because ideal ordering of relevant document are usually not available.

mAP@N

$$\begin{aligned} precision &= \frac{\text{relevant recommendations}}{\text{total recommendations}} \\ P@k &= \text{Precision for top k recommendations} \\ rel(k) &= \begin{cases} 1 & k^{th} \text{ recommendation is relevant} \\ 0 & \text{Otherwise} \end{cases} \\ AP@N &= \frac{1}{M} \sum_{k=1}^{N} P@k \cdot rel(k) \end{aligned}$$

AP automatically weights placement. Relevant recommendation in early place gets higher score. To compute mAP, simply compute mean of AP for all users.

mAR@N

$$\begin{aligned} recall &= \frac{\text{relevant recommendations}}{\text{possible relevant items}} \\ R@k &= \text{Recall for top k recommendations} \\ AR@N &= \frac{1}{M} \sum_{k=1}^{N} R@k \cdot rel(k) \end{aligned}$$

AP automatically weights placement. Relevant recommendation in early place gets higher score. To compute mAP, simply compute mean of AP for all users.