→ 'Naive' Bayes is called Naive because it does not take into account the conditional dependencies.

$$P(y/x) = \frac{P(y \wedge x)}{P(x)}$$

$$= \frac{P(x \wedge y)}{P(x)} = \frac{P(x/y) \cdot P(y)}{P(x)}$$

$$\therefore \quad P(y/x) = \frac{P(x/y) \cdot P(y)}{P(x)}.$$

$$P(X/y) = P(x_i/y) \cdot P(x_{i+1}/y) \cdot P(x_{i+2}/y) \ldots P(x_n/y).$$

$$X = \{x_1, x_2 \ldots x_n\}.$$

$$y = \text{argmax}_i \; P(k_i) \, P(X/k_i).$$

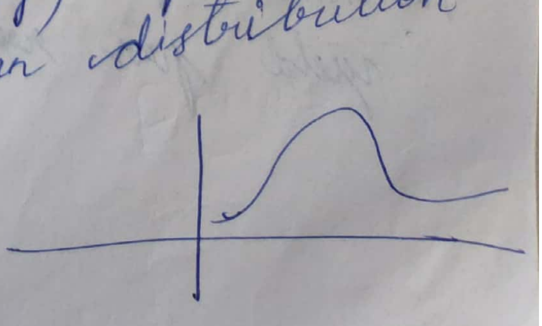
Select the class whichever has the highest probability.

Naive Bayes is divided into following:-

a) <u>multinomial</u>:- Generally used for Spam detection, Sentiment Classification etc.

b) <u>Bernouli</u>:- Where the feature Vectors are Binary (0 or 1) $P(x_i/y)$ is a Binary feature

c) <u>Gaussian</u>:- $P(x_i/y)$ follows a Gaussian distribution



The technique of reinforcement learning is considered with the problem of finding suitable actions to take in a given situation in order to maximize the reward.

Backgammon game = reinforcement learning + machine learning (neural networks).

Input of Neural Network = Board State + Dice throw.

Reward can be calculated only at the End if it is a win or loss.

→ A Net should play against it, thousands of times to learn the game.

The reward must be attributed to all the moves, but there are some good moves and some bad moves, how will you assign the credit to each move is called credit assignment problem.

In reinforcement learning there is a trade off between exploration and Exploitation. Too much focus on any of them will yield poor results.

Gradient descent algorithm

The minimum ~~function~~ value for a function can be obtained using gradient descent algo.

$$f(x) = \sum_{i} \left((\theta_0 + \theta_1 x_i) - y_i \right)^2$$

$$\left\{ \theta_j = \theta_j - \alpha \frac{\partial f(x)}{\partial \theta_j} \quad \forall \theta_j \right.$$

} do until it converges

→ Hypothesis function is one which maps input to output.

$$f(x) = y$$

→ In case of loss functions, if you want to find the parameters for which the loss function is minimum, it is difficult to iterate over all parameters ~~all~~ and find, so we use gradient descent on a randomly iniated parametre values.

Scanned with CamScanner

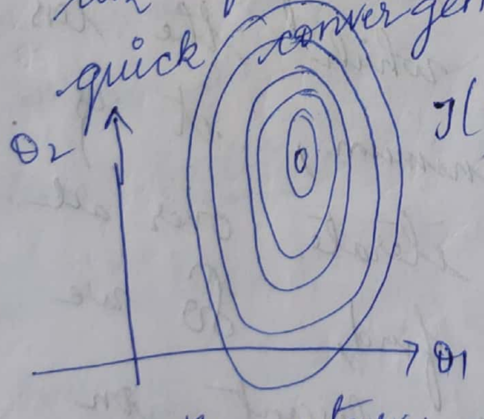
Gradient descent works only for convex functions cause there is only one local (global) minima.

→ In case of non-convex functions multiple local minimas exist, so GD func can't be used. Because it may settle on a local minima than a global minima.

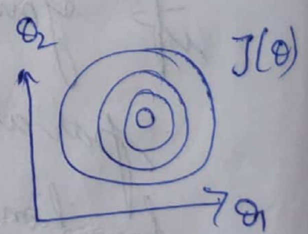
→ Feature Scaling:-

make sure features are on a similar scale.

It decreases the contour size of the loss function, there by helping in quick convergence. ~~to lower~~ &



Big contours.



Small contour

Mean normalization:-

Subtract the mean value of the feature from each point belonging to that feature space before normalizing it, so that the data points are centred around zero.

$$X_i^k = \frac{x_i^k - M^k}{S^k}$$

M^k = mean of feature $\left(M_k = \frac{\sum^m X_i^k}{m}\right)$

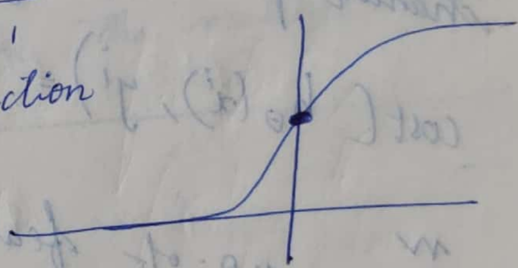
S_k = range of feature value k. (max - min)

Logistic Regression:-

$$f(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots \theta_n x_n$$

$$g(f(x)) = g(z) = \frac{e^{\theta x}}{e^{\theta x} + 1}$$

$g(z)$ is a Sigmoid function



→ The function helps classify into two classes.

$\theta x = 0$ is the decision boundary.

Cost function:-

$-\log(g(z))$ if $y = 1$

$-\log(1 - g(z))$ if $y = 0$

combined Eq^n:-

$$\text{cost}(g(z), y) = -y \log(g(z)) - (1-y)\log(1 - g(z))$$

principle of maximum likelihood Estimation:-

This is used to estimate the parameters of a model

→ Regularisation:-

Underfit - Has "high bias" irrespective of the evidence.

Overfit - too many features, the function cannot generalize over new examples → "High Variance".

→ In order to ~~over~~ prevent overfit, we can make the values of parameters small, resulting in simpler hypothesis and less chances of overfitting.

$$cost\left(h_\theta(x^i), y^i\right) = \frac{\left(h_\theta(x^i) - y^i\right)^2}{2} + \lambda \sum_{i=1}^{n} \theta_i^2$$

λ ↓ reg. parameter

n - no. of features

→ When you try to minimize the cost function, it will assign less value to parameters "θ".

$$\sum_{i=1}^{m} \theta_i^2 = W^T W$$

Regularisation can be done in gradient descent and also "finding parameters by making gradient equal to zero using matrix inversion method".

$X\theta = Y$

$\theta = (X^T X)^{-1} X^T y$

"if" $m < n$
(# examples) (# features)

then $(X^T X)^{-1}$ is
Singular / Non-invertible

if $\lambda > 0$

$$\theta = \Big(\underbrace{X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}}_{\text{Invertible}}\Big)^{-1} X^T y$$

Adding regularisation, makes the matrix invertible

Generative modelling :-

In Generative modelling we learn the probability distribution of "x" i.e $p(x)$ or "x and y" i.e $p(x \wedge y)$.

Discriminative models :- In discriminative modelling we learn the $P(y/x)$.

→ An Example of $p(x \wedge y)$ is in the case of CRF's. where you try to Estimate the probability of $(x \wedge y)$ using some feature functions.

In CRF's $p(y/x) = \dfrac{P(x \wedge y)}{P(x)}$

where $P(x). = \sum_y p(x \wedge y)$.

So, in CRF's you are kind of doing generative modelling followed by discrimination ;) . (Refer more to NOW Publishers).

→ CRF's are undirected probabilistic models.

→ Bayesian networks (directed models)

→ Markov property

SVD:- Given a matrix, decompose (factorize) it into its Eigen Vectors.

$$A = U \Sigma V$$
$$[m \times n] \quad [m \times r][r \times r][r \times n].$$

r is the number of Topics.

'U' and 'V' are orthogonal matrices. Meaning each of its column vectors are orthogonal to each other

$$\text{Dot\_product}(C_i, C_j)_{i \neq j} = 0.$$

Its like ~~finding~~ the all the axes of the co-ordinate system, which are mutually perpendicular to each other.

'Σ' is a diagonal matrix; only the index (i,j) $(i=j)$ are non-zero.

These non-zero elements are called 'Singular Values' and 'U' is called 'left-Singular Matrix', whereas 'V' is called 'Right-Singular Matrix'.

→ The Row Vectors in matrix $V_{[r \times n]}$ can be treated as 'Eigen Vectors'.

(fbpca library) helps you decompose a given matrix 'A' into U, Σ, V matrices.

→ The input matrix 'A' can be represented using 'tf_idf' or 'count_vectorizer'.

Gaussian process:- predict the class of new point by calculating its relation to existing known datapoints ((x,y) points).

→ Non-parametric, high-inference time.

Normalising flows:- These help you to move you from a Simple distribution to a more complex distribution.

→ These are/can be used in variational auto-encoders to model the latent Space (Z-Space).

→ Variational auto-encoders differ from auto-encoders in a way that, auto-encoders predict the latent variable Z, variation A-E's predict the 'mean' and 'variance' of Z. A data point is sampled from that distribution using re-parametrisation trick, so that the gradients can be back-propogated.

Bayesian Optimization:- Done when the number of (x,y) points we can obtain is costly. (Digging of oil wells in oceans). Given few points, you approximate the function; you infer the value of new 'x'. This 'x' selection is based upon 'exploitation/Exploration trade off. [Not Sure] → This is also called acqusition function.

Scanned with CamScanner

→ In variational auto-encoders, the main idea from a probabilistic perspective is used to minimize the KL Divergence of $p(z/x)$ and $P_\lambda(z/x)$, where P_λ is the assumed probability distribution which you want to get it closer to the original probability $p(z/x)$ distribution.

→ In this process $p(x)$ appears, which is intractable. So, they calculate ELBO and try to minimize it.

→ This is the intuition for building the NN architecture of VAE's.

$$KL\left(P(z/x) \,/\, P_\lambda(z/x) \right).$$

$$\Rightarrow KL = \log P(z/x) - \log\left(P_\lambda(z/x)\right).$$

$$= \log\left(P(z \wedge x)\right) - \log(x) - \log\left(P_\lambda(z/x)\right)$$

$$= \log p(x/z) + \log(z) - \log(x) - \log P_\lambda(z/x).$$

$$KL = \underbrace{\log P(x/z) - \left[\log P_\lambda(z/x) - \log(z)\right]}_{(ELBO)} + \log(x)$$

→ $\log p(x/z) \rightarrow$ likelihood.

$\left[\log P_\lambda(z/x) - \log(z)\right] \Rightarrow$ minimise the diffe. between P_λ and Z.

→ In the above KL is written opposite [wrong def/sym].