Rishabh Indoria

Machine Learning Techniques

December 23, 2023

1 Week 1

1. Paradigms of Machine Learning
   - Broad Paradigms: Supervised Learning, Unsupervised Learning, Sequential Learning
   - Foundations: Linear Algebra for Structure, Probability for Uncertainty, Optimization for Decision
2. Representation Learning
   - Part of Unsupervised Learning
   - **Compression**: Act of finding patterns in data.
     Representation: Some sort of relation within the data point
     Coefficients: Part of the data point, which when written as the representation, we could get the original data point back.
   - Choose Representation and Coefficient such that reconstruction error is minimized.
   - n represents the number of data points and d represents the number of features.
   - Projection of $\mathbf{x}$ onto line $\mathbf{w}$ is $\frac{\mathbf{x}.\mathbf{w}}{||\mathbf{w}||}\frac{\mathbf{w}}{||\mathbf{w}||}$
   - Optimal value of $\mathbf{w}$ is the eigenvector corresponding to the maximum eigenvalue of the covariance matrix. If there are d features, then the Covariance matrix will be a $d \times d$ matrix. The Covariance matrix is $\frac{1}{n}\sum_1^n x_i x_i^T$.
   - We want to minimize $\mathbf{w}C\mathbf{w}.T$.
   - All the residuals $\mathbf{x} - (\mathbf{x}.\mathbf{w})\mathbf{w}$ are perpendicular to the line $\mathbf{w}$
   - **Centred Data**: the Mean value of the dataset is 0. To centre a dataset, you simply subtract the dataset by the mean of the dataset.
3. Principal Component Analysis
   - $\mathbf{w}_1$ is the line which minimizes the reconstruction error of a set of points, and $\mathbf{w}_2$ is the line which minimizes the reconstruction error of the residues. These lines are orthogonal to each other.
   - We find the best fit line $\mathbf{w}_1$, then we find residuals, then using the residuals we find the best fit again $\mathbf{w}_2$. We keep doing this for d iterations. After which, residues will definitely become 0. The final residue will be $\mathbf{x} - (\mathbf{x}.\mathbf{w}_1)\mathbf{w}_1 - (\mathbf{x}.\mathbf{w}_2)\mathbf{w}_2 - ...$
   - The vectors $\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_d$ are orthogonal and span or are the basis of $\mathbb{R}^d$.
   - The best line one can obtain at kth round of the algorithm is the eigenvector corresponding to the kth maximum eigenvalue of the Covariance matrix.

2 Week 2

1. Concerns in PCA
   - Time Complexity: Finding the eigenvalues and eigenvectors, takes about O(d^3). Issue when d is large.
   - It tries to find linear combinations as such, non-linear relationships do not fit well.
2. Time complexity Issue
   - Large d [$d >> n$]; X is $n \times d$
   - Let w_k be the eigenvector corresponding to the kth largest eigenvalue of C(λ_k).
     C$w_k = \lambda_k w_k$
   - $w_k = X\alpha_k$, $\alpha_k X^T X \alpha_k = 1$
   - Non zero eigenvalues of XX^T and $X^T X$ are exactly the same.
   - β_k are eigenvectors corresponding to $n\lambda_k$ of $X^T X$, converting them according to constraint $\alpha_k = \frac{\beta_k}{\sqrt{n\lambda_k}}$

- Compute K = $X^T X$, Compute eigen decomposition, convert eigenvectors according to constraint, then finally $w_k = X\alpha_k$

3. Feature Transformation: Increase the dimensions, such that non-linear relationships are captured, then apply PCA.

4. Kernel function

   - To convert from quadratic to linear, we map the features to $\phi(x) = [1\ f_1^2\ f_2^2\ f_1 f_2\ f_1\ f_2]$
   - Any function k: $\mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ which is a valid map is called a kernel function.
   - A function k is a valid kernel function if there exists a function $\phi : \mathbb{R}^d \to \mathbb{R}^D$ such that $k(x_1, x_2) = \phi(x_1^T)\phi(x_2)$ for all $x_1, x_2 \in \mathbb{R}^d$
   - Kernel is symmetric and positive semi definite. All eigenvalues of k are non-negative.
   - Polynomial Kernel: $k(x, x') = (x^T x' + 1)^2$
   - Radial Basis function kernel or Gaussian Kernel: $k(x, x') = e^{\frac{-||x-x'||^2}{2\sigma^2}}$

5. Kernel PCA

   - Compute Kernel matrix using a kernel function
   - Center the Kernel using the formula: $K^C = K - 1_n K - K 1_n + 1_n K 1_n = (I - 1_n)K(I - 1_n)$, where 1_n is a matrix with all elements $\frac{1}{n}$
   - Compute eigenvalues $\{n\lambda_1, n\lambda_2, ...\}$ and eigenvectors $\{\beta_1, \beta_2, ...\}$ of K and normalize the eigenvectors, $\alpha_u = \frac{\beta_u}{\sqrt{n\lambda_u}}$.
   - Compute the transformed data points, $\phi(x_i)^T w = [\sum \alpha_{1j} K_{ij}^C, \sum \alpha_{1j} K_{ij}^C, ...]$
   - We cannot 'recompute' the eigenvectors of the covariance matrix as they would require computing, ϕ which would defeat the whole purpose.
   - But we can still compute a "compressed" representation.

3 Week 3

1. Intro to Clustering

   - Goal: Partition the given data into k different clusters.
   - Data points: $\{x_1, x_2, ...\}$
   - Cluster Indicator: $\{z_1, z_2, ...\}$
   - Performance Matrix: $F(z_1, ..., z_n) = \sum_{i=1}^n ||x_i - \mu_{z_i}||_2^2$, where μ_{z_i} is the mean/average of z_i cluster.
   - Goal is to minimize Performance matrix

2. K-means Clustering

   - Also known as Lloyd's Algorithm
   - Step 1, Initialization: We define some assignment to clusters $z_1^0, z_2^0, ..., z_n^0$
   - Then until convergence we
   - Step 2, Compute means: $\mu_k^t = \frac{\sum x_i 1(z_i^t = k)}{\sum 1(z_i^t = k)}$
   - Step 3, Reassignment: $z_i^{t+1} = \arg\min_k ||x_i - \mu_k^t||_2^2$
   - FACT: LLOYD'S ALGORITHM CONVERGES, but converged solution may not be "optimal". But produces "reasonable" cluster in practice.

3. Convergence of K-means

   - FACT 1: Let $x_1, x_2, ..., x_l$, $v^* = \arg\min_v \sum ||x_i - v||^2$. $v^* = \frac{\sum x_i}{l}$
   - In every iteration, the objective function strictly reduces, which implies that no partition repeats.
   - There are only "FINITE" number of partitions as such algorithm must converge.

4. Nature of Clusters

   - For a cluster with mean μ_1, all x assigned to it will satisfy $x^T(\mu_n - \mu_1) \leq \frac{||\mu_n||^2 - ||\mu_1||^2}{2}$, for all $n \neq 1$
   - VORONOI region: intersection of half spaces. Cluster regions are voronoi regions.

- K-means can not efficiently cluster data points that are not linearly separable. Kernel K-means is used to cluster data points that are not linearly separable. Spectral Clustering can also be used.

5. Initialization of centroids

   - Pick K-means uniformly at random from the dataset.
   - Means should be far apart.
   - K-means++: Choose first mean μ_1^0 uniformly at random from the dataset. For $l = 2, 3, ..., k$ choose μ_l^0 probabilistically proportional to score.
   - Score $S(x) = \min_j ||x - \mu_j^0||^2$

6. Choice of K

   - We want K to be not as small and not as large. Penalize large values of K.
   - Value of K is where Objective function + Penalty function is the smallest.
   - Akaike Information Criterion: $2K - 2\log(L(\theta^*))$
   - Bayesian Information Criterion: $K\log(n) - 2\log(L(\theta^*))$

4 Week 4

1. Introduction to Estimation

   - Estimation: There is some probabilistic mechanism that generates the data. About which we don't know "something".
   - Goal: Observe data and "Assume" a probabilistic model that generates the data.
   - Assumption: Observations are **Independent** and **Identically Distributed**

2. Maximum Likelihood Estimation

   - Fisher's Principle of Maximum Likelihood: Write the likelihood function
     $L(p, \{x_1, x_2, ...x_n\}) = P(x_1, x_2, ..., x_n, p)$
     $= P(x_1, p).P(x_2, p)...P(x_n, p) = \Pi_{i=1}^n p^{x_i}(1-p)^{1-x_i}$ [Independence]
   - Estimator: $\hat{p}_{ML} = \arg\max_p \Pi_{i=1}^n p^{x_i}(1-p)^{1-x_i}$
     We take logarithm to simplify
     $= \arg\max_p \sum_{i=1}^n x_i \log(p) + (1-x_i)\log(1-p)$ [log is monotonically increasing]
     Taking Derivative and setting to 0 we get,
     $\hat{p}_{ML} = \frac{\sum_{i=1}^n x_i}{n}$
   - Fisher's Proposal: $L(\mu, \sigma^2, \{x_1, x_2, ...x_n\}) = f_{x_1, x_2, ...x_n}(x_1, x_2, ...x_n, \mu, \sigma^2) = \Pi f_{x_i}(x_i, \mu, \sigma^2)$
     This is done because for continuous functions, Probability only exists for intervals and probability at any particular point would be 0.

3. Bayesian Estimation

   - Goal: Incorporate "Hunch" about parameters into the estimation procedure.
   - Approach: Think of the parameter to estimate as a "random" variable.
   - Hunch: codified using a probabilistic distribution over θ
   - After looking at data, move to
     Updated Hunch: Codified using another probabilistic distribution.
   - $P(\theta|\{x_1, x_2, ...x_n\}) = \frac{P(\{x_1, x_2, ...x_n\}|\theta)P(\theta)}{P(\{x_1, x_2, ...x_n\})}$
   - BETA PRIOR: $f(p, \alpha, \beta) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{z}$
   - BETA POSTERIOR$(\alpha + n_h, \beta + n_t)$
   - One possible guess $= \frac{\alpha + n_h}{\alpha + \beta + n}$

4. Gaussian Mixture Models

   - STEP 1: Pick which mixture a data point comes from.
   - STEP 2: Generate data point from that mixture.
   - Generate a mixture component among $\{1, ..., k\}$, $z_i \epsilon \{1, ..., k\}$
     $P(z_i = l) = \pi_l$

- Generate $x_i = N(\mu_{z_i}, \sigma^2_{z_i})$
- Observed: $\{x_1, x_2, ..., x_n\}$
  Unobserved: $\{z_1, z_2, ..., z_n\}$
  Parameters: $\pi = [\pi_1, \pi_2, ..., \pi_k]$, and for each k (μ_k, σ^2_k)

5. Likelihood of GMM

   - $L(Parameters, \{x_1, x_2, ..., x_n\}) = \Pi^n_{i=1} f(x_i; Parameters)$
     $= \Pi^n_{i=1}[\sum^k_{k=1} \pi_k f(x_i; \mu_k, \sigma^2_k)]$
     $L(Parameters) = \Pi^n_{i=1}[\sum^k_{k=1} \pi_k \frac{e^{\frac{-(x_i-\mu_k)^2}{2\sigma^2_k}}}{\sqrt{2\pi}\sigma_k}]$
   - $\log(L(Parameters)) = \sum^n_{i=1} \log(\sum^k_{k=1} \pi_k \frac{e^{\frac{-(x_i-\mu_k)^2}{2\sigma^2_k}}}{\sqrt{2\pi}\sigma_k})$

6. Convex Functions and Jensen's inequality

   - Convex Functions: $f(\frac{a+b}{2}) \leq \frac{f(a)+f(b)}{2}$ $\forall a, b$
   - Jensen's inequality: $f(\lambda_1 a_1 + ... + \lambda_k a_k) \leq \lambda_1 f(a_1) + ... + \lambda_k f(a_k)$
     $f(\sum \lambda_k a_k) \leq \sum \lambda_k f(a_k)$, $\sum \lambda_k = 1$
   - Concave Functions: $f(\frac{a+b}{2}) \geq \frac{f(a)+f(b)}{2}$ $\forall a, b$
   - Jensen's inequality: $f(\lambda_1 a_1 + ... + \lambda_k a_k) \geq \lambda_1 f(a_1) + ... + \lambda_k f(a_k)$
     $f(\sum \lambda_k a_k) \geq \sum \lambda_k f(a_k)$, $\sum \lambda_k = 1$
   - Linear Functions are both Concave and Convex.
   - Log is a concave function

7. Estimating the parameters

   - $\log(L(Parameters)) = \sum^n_{i=1} \log(\sum^k_{k=1} \lambda^i_k (\pi_k \frac{e^{\frac{-(x_i-\mu_k)^2}{2\sigma^2_k}}}{\lambda^i_k \sqrt{2\pi}\sigma_k}))$
     Apply Jensen's inequality, Fix λ and then take derivative and set to 0
   - $\hat{\mu_k} = \frac{\sum^n_{i=1} \lambda^i_k x_i}{\sum^n_{i=1} \lambda^i_k}$, $\hat{\sigma^2_k} = \frac{\sum^n_{i=1} \lambda^i_k (x_i - \hat{\mu_k})^2}{\sum^n_{i=1} \lambda^i_k}$, $\hat{\pi_k} = \frac{\sum^n_{i=1} \lambda^i_k}{n}$
   - Fixing all parameters and maximizing with respect to λ
     $\hat{\lambda^i_k} = \frac{\frac{1}{\sqrt{2\pi}\sigma_k} e^{\frac{-(x_i-\mu_k)^2}{2\sigma^2_k}} . \pi_k}{\sum^k_{l=1} \frac{1}{\sqrt{2\pi}\sigma_l} e^{\frac{-(x_i-\mu_l)^2}{2\sigma^2_l}} . \pi_l} = \frac{P(x_i|z_i=k)P(k)}{P(x_i)}$
     $\forall i$, $\sum^K_{k=1} \lambda^i_k = 1$

8. Expectation Maximization Algorithm

   - Initialize Parameters
   - Then until convergence
     Expectation Step: Calculate λ^{t+1} using $Parameters^t$
     Maximization Step: Calculate $Parameters^{t+1}$ using λ^{t+1}

5 Week 5

1. Supervised Learning

   - Input is features/attributes $\{x_1, ...x_n\}$ and labels $\{y_1, ..., y_n\}$.
   - If labels only take two values, then the problem is called binary classification problem. If labels take n values, then the problem is called multi class classification. If labels take any value in real number, then the problem is called a regression problem.

2. Linear Regression

   - Goal is to learn a function f which maps a given feature to its correct label.
   - Error$(f) = \sum (f(x_i) - y_i)^2$
   - For linear regression we take $f(x) = w^T x$, and minimize Error(f).

- $\begin{bmatrix} - & - & - x_1 & - & - \\ - & - & - x_2 & - & - \\ - & - & - x_n & - & - \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_n \end{bmatrix}$, we want to minimize $(x^T w - y)^2$ or $(x^T w - y)^T (x^T w - y)$.
- Simply take gradient and set to 0, we get $(xx^T)w^* = xy$.
- Since w^* is a solution of an unconstrained optimization problem, we can apply gradient descent $w^{t+1} = w^t - \eta^t \nabla f(w^t)$, where η is the step size.
  $w^{t+1} = w^t - \eta^t 2(xx^T w^t - xy)$
- In case number of data points is large, we use
  Stochastic Gradient Descent: For t iterations, sample a bunch(k) of data points uniformly at random from the set of all points. Pretend this sample is the entire dataset and take a gradient step w.r.t it. After all rounds, we use $w^T_{SGD} = \frac{1}{T} \sum w^t$.
- **Kernel Regression** is similar to kernel PCA, we take $w^* = x\alpha^*$ and $K = x^T x$. $\alpha^* = K^{-1} y$.
  To make a prediction $w^* \phi(x_{test}) = \sum_{i=1}^n \alpha_i^* K(x_i, x_{test})$
- **Probabilistic Regression**, assume labels are generated as $w^t x + \epsilon$, where ϵ is noise generated from a Gaussian distribution. Using Maximum Likelihood function we simply arrive at
  $\hat{w_{ML}} = \min \sum (w^T x - y)^2$.

6 Week 6

1. Goodness of Maximum Likelihood Estimator
   - To understand how good $\hat{w}_{ML}$ is in estimating w, $E[||\hat{w}_{ML} - w||^2] = \sigma^2 trace((xx^T)^{-1})$
   - $trace((xx^T)^{-1}) = \sum \frac{1}{\lambda_i}$, where λ_i are the eigenvalues of xx^T
   - $\hat{w}_{new} = (xx^T + \lambda I)^{-1} xy$, $trace((xx^T + \lambda I)^{-1}) = \sum \frac{1}{\lambda_i + \lambda}$
   - **Existence Theorem**, There exists some λ such that $\hat{w}_{new}$ has lesser mean squared error than $\hat{w}_{ML}$. We find this λ by cross validation.
   - Split the training set into train set and validation set. Train on the train set and check for error on validation set. Pick λ that gives the least error.
   - **K-fold Cross Validation**, Split dataset into K folds, train on K-1 folds and validate on the last fold. Pick λ that gives the least average error.
   - **Leave One Out Cross Validation**, train on $n-1$ data points and validate on the last point.
2. Bayesian Modelling for Linear Regression
   - Need a Prior on w, a choice of prior $N(0, \gamma^2 I)$
   - Posterior is proportional to $P(dataset|w)P(w) = (\prod e^{\frac{-(y_i - w^T x_i)^2}{2}}) e^{\frac{-||w||^2}{2\gamma^2}}$
   - $\hat{w}_{MAP} = \min \frac{1}{2} \sum (y_i - w^T x_i)^2 + \frac{1}{2\gamma^2} ||w||^2$, taking gradient and setting to 0 we get
     $\hat{w}_{MAP} = (xx^T + \frac{1}{\gamma^2} I)^{-1} xy$
3. Ridge Regression
   - $\hat{w}_R = \arg\min \sum (w^T x_i - y_i)^2 + \lambda ||w||^2$, where the added term is called regularization term.
   - Ridge pushes weight values towards 0 but does not necessarily make it 0.
4. Lasso Regression
   - An alternate way is to regularize would be using L1 norm(Summation of absolute values instead of squared values).
   - Much more likely to make some weight values 0. But it does not have a closed form solution.
   - Sub gradients methods are usually used to solve LASSO.

7 Week 7

1. Binary Classification
   - Labels belong to the set $\{0, 1\}$ or the set $\{-1, 1\}$
   - Loss$(h) = \frac{1}{n} \sum 1(h(x_i) \neq y_i)$
   - $h(x) = sign(w^T x)$

2. K Nearest Neighbours

   - Given a test point x_{test}, find the closest point x' to x_{test} in the training set. Predict $y_{test} = y'$.
   - Can get affected by outliers, Ask more neighbours and predict the majority.
   - Problems: Choosing a distance function, Prediction is computationally expensive, No model is learnt.

3. Decision Trees

   - A question is a (feature, value) pair. Is feature $\leq$ value ?
   - Need a measure of "Impurity" for a set of labels to determine how good a question is.
   - Entropy function $= -(p\log(p) + (1-p)\log(1-p))$, where p is the fraction of 1's, and log(0) is considered 0.
   - Information Gain(feature, value) = Entropy(D) - [γEntropy(D_{yes}) + (1-γ)Entropy(D_{no})], where $\gamma = \frac{|D_yes|}{|D|}$
   - Discretize each feature in [min, max] range. Pick the question that has the largest Information Gain. Repeat the procedure for D_{yes}, D*no*.
   - Can stop growing a tree if a node becomes "Sufficiently" Pure. Depth of the tree is a hyperparameter. There are alternate measures for "goodness" of a question.
   - Gini Index function is another popular function to measure impurity.

4. Types of Modelling

   - Generative Model: $P(x, y)$
   - Discriminative Model: $P(y|x)$

8 Week 8

1. Generative Model based Algorithm

   - Data: $\{(x_1, y_1), ..., (x_n, y_n)\}$, where $x\epsilon\{0,1\}^d$ and $y\epsilon\{0,1\}$.
   - Step 1: Decide the labels by tossing a coin with $P(y_i = 1) = p$
   - Step 2: Determine the features using the labels obtained in Step 1 through the conditional probability $P(x_i|y_i)$.
   - The parameters in generative modelling are defined as $\hat{p}$ to decide the label, $2^d - 1$ parameters for $P(x|y = 1)$ and $2^d - 1$ parameters for $P(x|y = 0)$. Where d is the number of features.
   - Too many parameters, could lead to overfitting and the model may not be practically viable.

2. Alternate Generative Model

   - Class conditional independence: This assumption states that the features of an object are conditionally independent given its class label.
   - Step 1 remains the same.
   - Step 2: Determine the features for x given y using the following conditional probability, $P(x = [f_1, f_2, ...f_n]|y) = \prod(p_i^{y_i})^{f_i}(1 - p_i^{y_i})^{f_i}$.
   - The parameters in generative modelling are defined as $\hat{p}$ to decide the label, d parameters for $P(x|y = 1)$ and d parameters for $P(x|y = 0)$. Where d is the number of features.
   - Parameters are estimated using Maximum Likelihood Estimator.

3. Naive Bayes Algorithm

   - The model is given by: $P(x = [f_1, f_2, ...f_n]|y) = \prod(p_i^{y_i})^{f_i}(1 - p_i^{y_i})^{f_i}$.
   - The parameters estimated are p, $\{p_1^0, p_2^0, ..., p_d^0\}$, and $\{p_1^1, p_2^1, ..., p_d^1\}$.
   - The estimates are $\hat{p} = \frac{1}{n}\sum y$, and $\hat{p}_j^y = \frac{\sum_{i=1}^n 1(f_j^i=1, y_i=y)}{\sum_{i=1}^n 1(y_i=y)}$
   - Given $x^{test}\epsilon\{0,1\}^d$, the prediction of $\hat{y}^{test}$ is done using the inequality $P(\hat{y}^{test} = 1|x^{test}) \geq P(\hat{y}^{test} = 0|x^{test})$.
   - Can express $P(\hat{y}^{test} = t|x^{test}) = \frac{P(x^{test}|\hat{y}^{test}=t)P(\hat{y}^{test}=t)}{P(x^{test})}$, since we are only comparing, there is no need to calculate x^{test}.

- One prominent issue with Naive Bayes is that if a feature is not observed in the training set, but present in the testing set, the prediction probabilities for both classes become zero.
  Laplace smoothing: A popular remedy for this issue is to introduce two "pseudo" data points with labels 1 and 0, respectively, into the dataset, where all their features are set to 1.
- The decision function of Naive Bayes is linear, and the boundary is given by
  $\{x = P(y = 0|x) = P(y = 1|x)\}$

4. Gaussian Naive Bayes

   - Assumes that features in the dataset follow a normal distribution and computes the likelihood of a class for a given set of feature values by estimating the mean and variance of the feature values within each class.
   - $P(x|y = 0) = N(\mu_0, \Sigma)$ and $P(x|y = 1) = N(\mu_1, \Sigma)$
   - $\hat{\mu}_t = \frac{\sum 1(y_i=t)x_i}{\sum 1(y_i=t)}$, $\hat{\Sigma} = \frac{1}{n}\sum(x_i - \hat{\mu}_{y_i})(x_i - \hat{\mu}_{y_i})^T$.
   - If the covariance matrices are equal then the decision boundary is linear, if they are unequal then the decision boundary is quadratic.
   - For unequal covariance matrix $\hat{\Sigma}_t = \frac{\sum(1(y_i=t)x_i-\hat{\mu}_t)(1(y_i=t)x_i-\hat{\mu}_t)^T}{\sum 1(y_i=t)}$.

9 Week 9

1. Perceptron Learning Algorithm

   - Widely employed for binary classification, focuses on modelling the boundary between each class.
   - Objective function, $\sum 1(h(x_i) \neq y_i)$
   - Until convergence, select a pair (x_i, y_i), if, $sign(w^T x_i) \neq y_i$ then update the weight vector $w^{t+1} = w^t + x_i y_i$.
   - $l^2\gamma^2 \leq ||w_{l+1}||^2 \leq lR^2$
   - Uber bound on number of mistakes
     $\#mistakes \leq \frac{R^2}{\gamma^2}$

2. Logistic Regression

   - Objective is to estimate the probability that the dependant variable belongs to one of two possible values.
   - Let $z = w^T x_i$, we define the $P(y = 1|x) = g(z) = \frac{1}{1+e^{-z}}$, where the function $g(z)$ is called the sigmoid function.
   - Objective is to maximize the log likelihood or minimize the negative log likelihood.
     $y_i \log(g(z)) + (1 - y_i)\log(1 - g(z))$
   - For gradient descent, we get the gradient as $x_i(y_i - g(z))$

10 Week 10

1. Support Vector Machines

   - category of supervised learning algorithms designed for classification and regression analysis. SVMs aim to identify the optimal hyperplane that maximizes the margin between data points from different classes.
   - Hard Margin SVMs: applicable only when the dataset is linearly separable
     Direct or Kernelized Calculation of Q: Compute the matrix $Q = X^T X$ directly or using a kernel, based on the dataset.
     Gradient Descent: Employ the gradient of the dual formula, $\alpha^T 1 - \frac{1}{2}\alpha^T Y^T Q Y \alpha$, in a gradient descent algorithm to iteratively find a satisfactory set of Lagrange multipliers α.
     $\text{label}(x_{test}) = \text{sign}(w^T x_{test}) = \text{sign}(\sum \alpha_i y_i (x_i^T x_{test}))$
     $\text{label}(x_{test}) = \text{sign}(w^T \phi(x_{test})) = \text{sign}(\sum \alpha_i y_i k(x_i^T x_{test}))$
   - Soft Margin SVMs: extends the standard SVM algorithm to accommodate some misclassifications in the training data. This extension is particularly useful when dealing with non-linearly separable data. It introduces a regularization parameter (C) to control the balance between maximizing the margin and allowing for misclassifications.
     $\min \frac{1}{2}||w||_2^2 + C\sum \epsilon_i$ such that $(w^T x_i)y_i + \epsilon_i \geq 1$, $\epsilon_i \geq 0$

11 Week 11

1. Bagging
   - Simply Distribute the dataset into m smaller datasets, them make m different models. For prediction, predict using each of the models, average out the prediction, then use the same function on the averaged prediction.
   - Can also use feature bagging.
2. Boosting
   - Input: S = $\{(x_1, y_1), ..., (x_n, y_n)\}$
   - Initialize $D_0(i) = \frac{1}{n}$
   - For t = 1 to T
     h_t = Input S to a weak Learner
     $\tilde{D}_{t+1}(i) = D_t(i)e^{\alpha_t}$ if $h_t(x_i) \neq y_i$ else $D_t(i)e^{-\alpha_t}$
     $D_{t+1}(i) = \frac{\tilde{D}_{t+1}(i)}{\sum \tilde{D}_{t+1}(i)}$
   - $\alpha_t = \log(\sqrt{\frac{1-error(h_t)}{error(h_t)}})$
   - $h^*(x) = \text{sign}(\sum \alpha_t h_t(x))$

12 Week 12

1. Activation Functions
   - Sigmoid function: $\frac{1}{1+e^{-z}}$
   - Rectified Linear Unit: max(0,z)