Supervised Learning

Practical Machine Learning

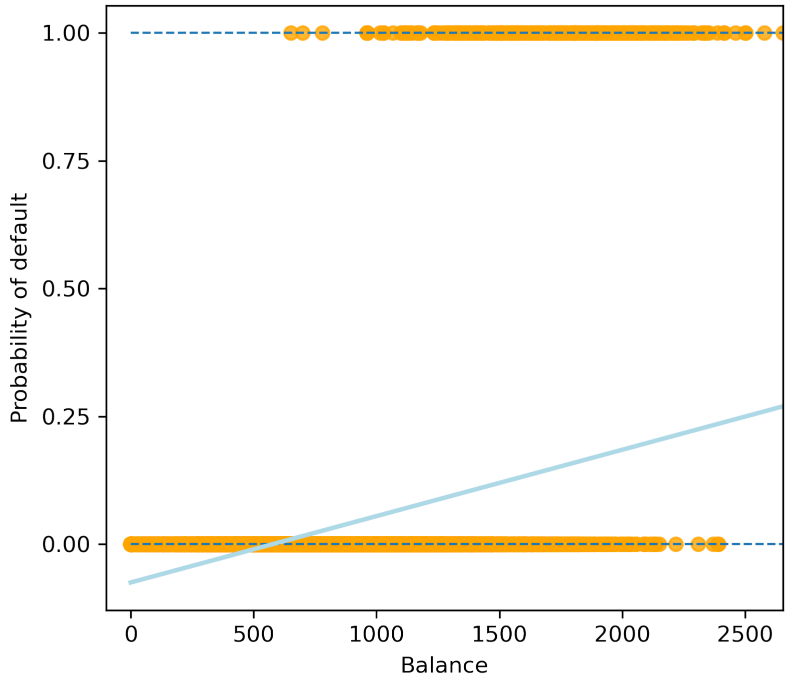
Kannan Singaravelu

May, 2022

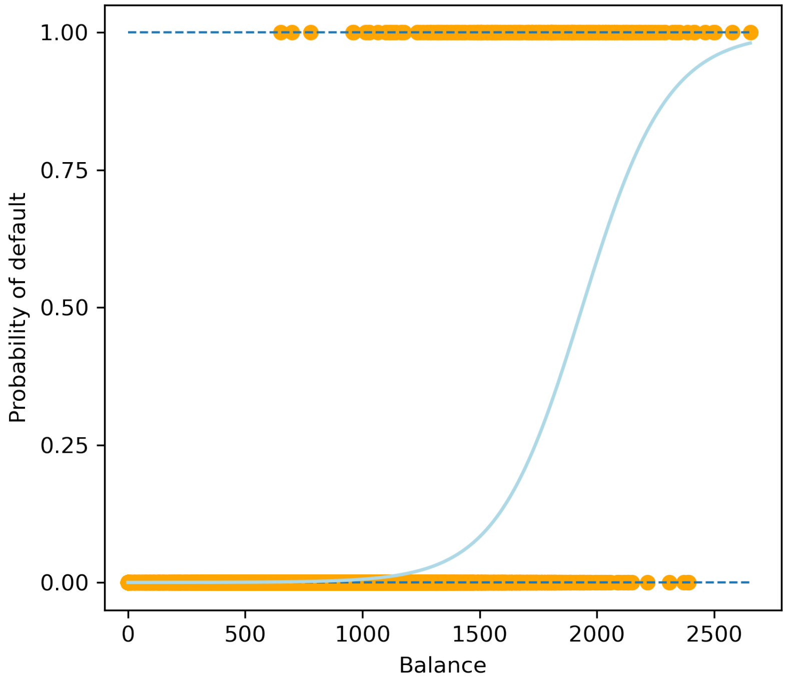
In this lecture…

- Understanding classification setting
- Logit functions & algorithms for classification
- Classification loss functions
- Model assessment
- Support Vector Networks
- Lagrangian Dual Formulation
- Three steps to find nearest neighbours
- Overview of hyperparameters tuning

Why Not Linear Regression?



Why Not Linear Regression?



The Classification Problem

- Classification models are models which predict a qualitative response.
- Estimate f on training observations $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_1, \ldots, y_n$
- The training error is given as

$$\frac{1}{n}\sum_{j=1}^{n} I(y_i \neq \hat{y}_i)$$

where,

y_i is the predicated class label,

$I(y_i \neq \hat{y}_i)$ is an indicator variable that equals 1 if $y_i \neq \hat{y}_i$ and equals 0 if $y_i = \hat{y}_i$

- The test error rate is then given as $Ave\,(I\,(y_i \neq \hat{y}_i)$

Bayes Classifier

- The test error rate can be minimised if we classify the classes correctly given its predictors.
- Assign test observation with predictor x_0 to the class j for which

  $Pr(Y = j \mid X = x_0)$ is the largest.
- Classifier based on this conditional probability is popularly known as the Bayes Classifier.
- For binary classification of class 1 and class 2, predicting class 1 corresponds to $Pr(Y = 1 \mid X = x_0) > 0.5$ class 2 otherwise.
- Bayes error rate produces lowest possible error rate and is given as

$$1 - E\left(\max_{j} Pr(Y = j \mid X)\right)$$

Logistic Regression

- One of the most widely used algorithms for classification.
- Map quantitative data onto categorial variables.
- Probabilistic model used to assign class labels.
- Two main type of classification problems
    - Binary or Binomial
    - Multi-class or Multinomial
- Also referred as the shallow neural network

Logistic Regression

- Find a function of the predicator variables that relates to 0 and 1 as outcome.
- Instead of y as outcome variable (like in linear regression), we use a function of y called the logit.
- Logit can be modelled as a linear function of the predictor and can be mapped back to a probability which in turn can be mapped to a class.
- Typically use probability with a cutoff value.

Logit

- Logit = log (odds) = $w_0 + w_1x_1 + w_2x_2 + \ldots w_qx_q$
- The odds of an event is defined as

  $Odds = \dfrac{p}{1-p}$, where p is the probability of event

- Probability of an outcome or event can be computed as

  $$p = \frac{Odds}{1 + Odds}$$

- If $p = 0.5$, $Odds = 1$ and $log(Odds) = 0$
- Log odds can range between $-\infty$ to $+\infty$ and the midpoint is when $p = 0.5$

Probability Calculation

Probability Calculation

Probability Calculation

Probability Calculation

Probability Calculation

- Function that converts log-odds to probability is the logistic function

$$log\left(\frac{p}{1-p}\right) = w_0 + w_1x_1 + w_2x_2 + \cdots + w_px_p = z$$

$$\frac{p}{1-p} = e^z$$

$$\frac{1}{p} - 1 = \frac{1}{e^z}$$

$$\frac{1}{p} = \frac{1+e^z}{e^z}$$

$$p = \frac{e^z}{1+e^z} = \frac{1}{1+e^{-z}} = \sigma(z)$$

Probability Calculation

- If response has 3 levels,

$$log\left(\frac{p_1}{p_2}\right) = z_1 \qquad log\left(\frac{p_3}{p_2}\right) = z_2$$

$$\frac{p_1}{p_2} = e^{z_1} \quad (1) \qquad \frac{p_3}{p_2} = e^{z_2} \quad (2) \qquad \frac{p_1 + p_3}{p_2} = e^{z_1} + e^{z_2} \qquad (3)$$

$$p_1 + p_2 + p_3 = 1 \quad \Rightarrow \quad p_1 + p_3 = 1 - p_2$$

$$rewriting\ (3)\ , \frac{1 - p_2}{p_2} = e^{z_1} + e^{z_2} \quad \Rightarrow \frac{1}{p_2} - 1 = e^{z_1} + e^{z_2} \quad \Rightarrow \frac{1}{p_2} = 1 + e^{z_1} + e^{z_2}$$

$$p_2 = \frac{1}{1 + e^{z_1} + e^{z_2}}$$

$$p_1 = \frac{e^{z_1}}{1 + e^{z_1} + e^{z_2}} \qquad from\ (1)$$

$$p_3 = \frac{e^{z_2}}{1 + e^{z_1} + e^{z_2}} \qquad from\ (2)$$

Logistic Regression

- Model the probabilities of the output classes given a function that is linear in x.
- Mathematically,

$$P_j = g\left(w_0 + \sum_{j=1}^{p} x_{ij} w_j\right) = g\left(w^T x\right)$$

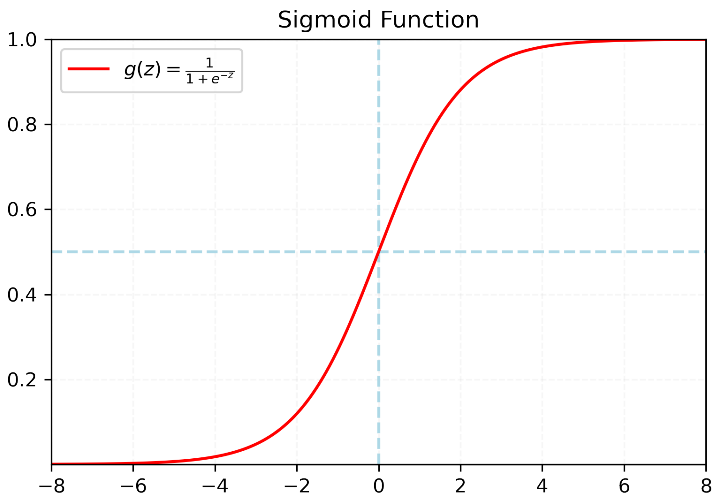
where, $g(z) = \dfrac{1}{1 + e^{-w^T x}} = \dfrac{1}{1 + e^{-z}}$

- Sigmoid function maps values that range from positive to negative infinity to values that only range from 0 to 1.
- Use probability with a threshold value for predictive classification.

Logistic Regression

- Prediction of the probability is then given as

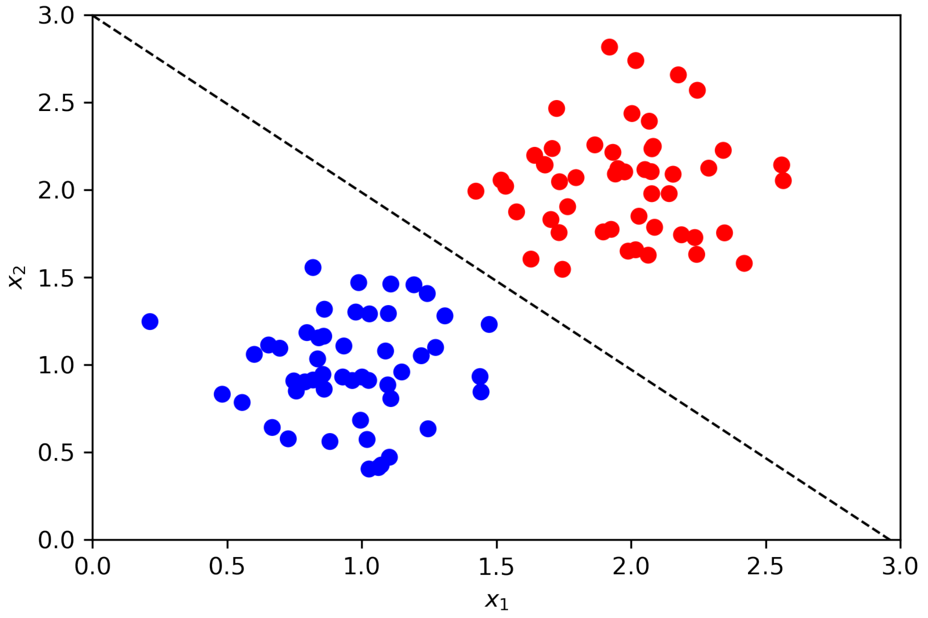
$$\hat{y} = \begin{cases} 0 & \text{if } \hat{p} < 0.5 \\ 1 & \text{if } \hat{p} \geq 0.5 \end{cases} \quad \text{where, } \hat{p} = g(z)$$



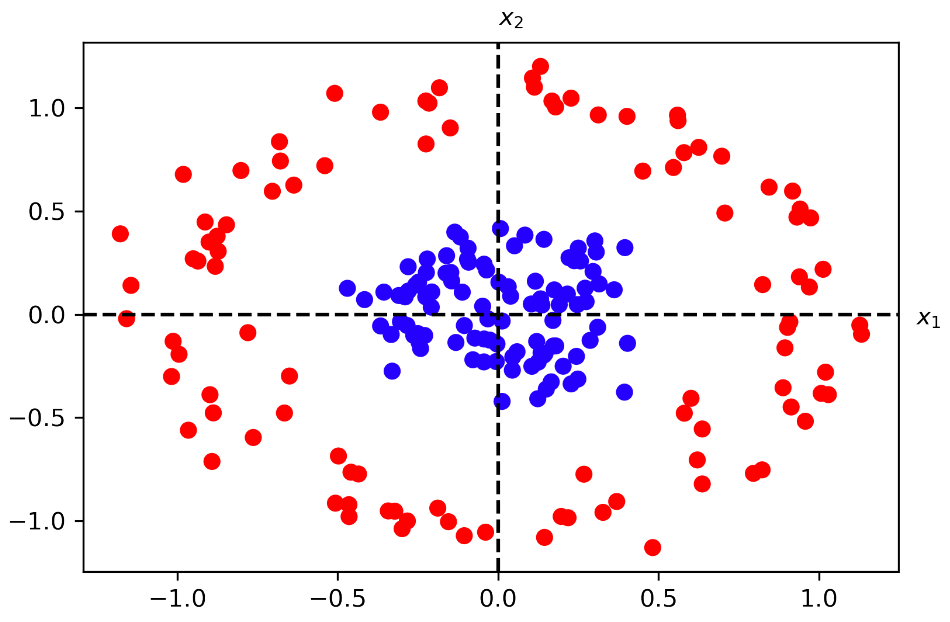
- $g(z) < 0.5$ when $z < 0$ and $g(z) \geq 0.5$ when $z \geq 0$
- Model predicts 1 if $g(z)$ is ≥ 0.5, and 0 if it is < 0.5

Shallow Neural Networks

Linear Decision Boundary



Non-Linear Decision Boundary



Decision Boundary

- There is a linear relationship between the features and the input of the sigmoid function.
- Decision boundary is a property of the hypothesis including parameters w_0, w_1, w_2 and not the property of the dataset.
- Adding polynomial features will result in more complex decision boundaries.

Loss Functions for Classification

- In general, the **0-1** loss function for classification take the form $l\,(\hat{y}, y) = 1(\hat{y} \neq y)$, where 1 is the indicator function.
- **Score Function**

  $$\text{Predict} = \begin{cases} +1, \text{ if } f(x) > 0 \\ -1, \text{ if } f(x) < 0 \end{cases}$$

  where $f(x)$ is the score for the input x and f may be called a score function.
- Magnitude of the score represents the confidence of our prediction.

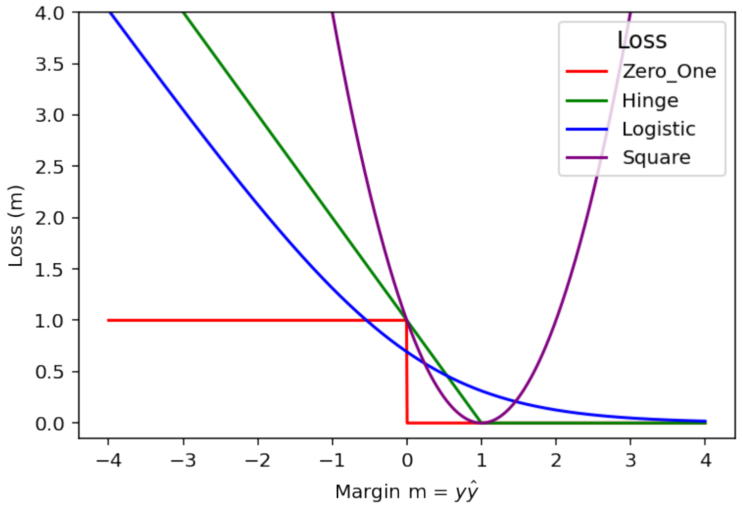
Loss Functions for Classification

- **Margin**
  - Functional margin is $\hat{y}y$
  - It is a measure of how correct our predictions are
    - If y and $\hat{y}$ are the same sign, prediction is correct and margin is positive.
    - If y and $\hat{y}$ have different sign, prediction is incorrect and margin is negative.
  - Objective is to maximise the margin.

- **Margin-based**
  - Most classification losses depend only on the margin.
  - Geometric margin - a related concept used in SVM.

Loss Functions for Classification

- **Zero-One Loss**
  - $l_{0-1} = 1(m \leq 0)$
  - Positive margin is correct classification
- **Hinge Loss**
  - $l_{Hinge} = max(1 - m, 0) = (1 - m)_+$
  - Margin error when $m < 1$
  - Not differentiable at $m = 1$
- **Log Loss**
  - $l_{Logistic} = log(1 + e^{-m})$
  - Differentiable, wants more margin (loss never zero).



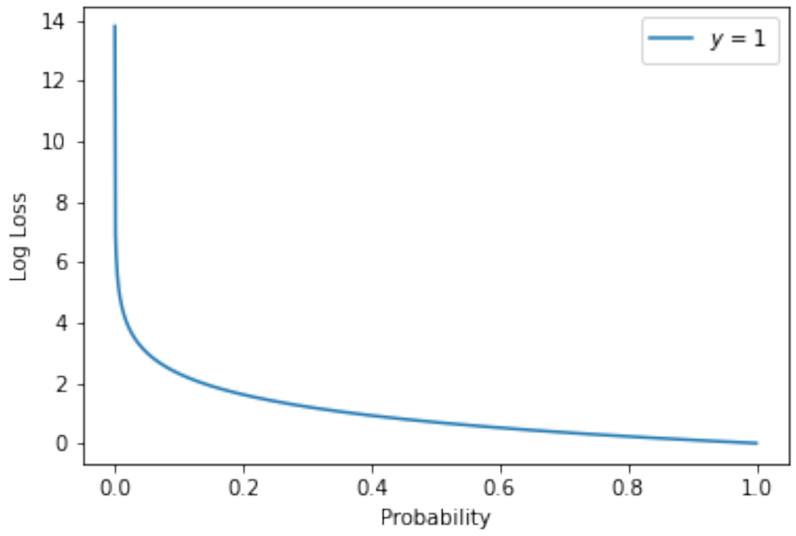
Why Not Square Loss?

Why Not Square Loss?

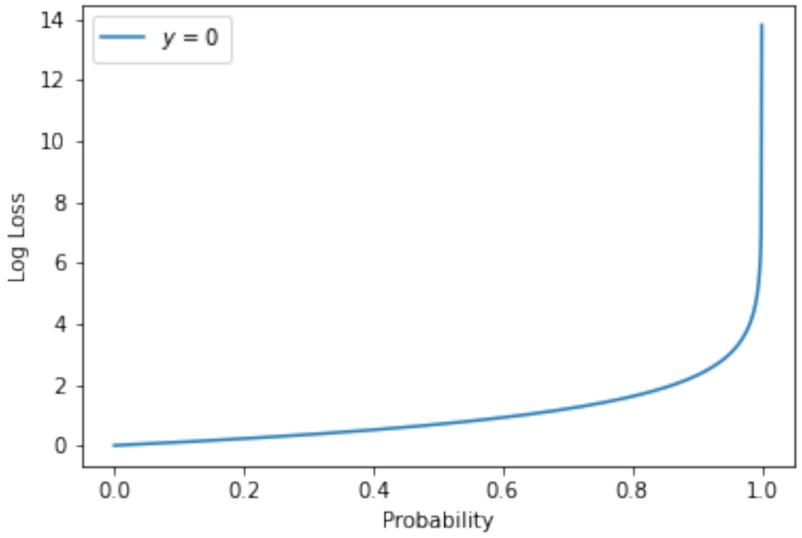
Why Not Square Loss?

Evaluation Metrics : Classification

$J(W) = -\,log(\hat{p})$



$J(W) = -\,log(1-\hat{p})$



- If $y = 1$, and $\hat{p} = 1$, then cost $= 0$
- $\hat{p} \to 0$, cost $\to \infty$

- If $y = 0$, and $\hat{p} = 0$, then cost $= 0$
- $\hat{p} \to 1$, cost $\to \infty$

Evaluation Metrics : Cost Function

Evaluation Metrics : Cost Function

- **For a single training pair (x, y)**

$$\hat{y} = P(Y = 1 \,|\, X = x) = \sigma(\mathbf{w} \cdot \mathbf{x}), \qquad where \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$P(y|x) = \begin{cases} \hat{y} & if \;\; y = 1 \\ 1 - \hat{y} & if \;\; y = 0 \end{cases}$$

If the labels are 0 and 1, then Y is a Bernoulli random variable $Y \sim Ber(p)$,

where, $p = \sigma(\mathbf{w} \cdot \mathbf{x})$,

$$\therefore \quad P(Y = y \,|\, X = x) = \sigma(\mathbf{w} \cdot \mathbf{x})^{\;y} \, [1 - \sigma(\mathbf{w} \cdot \mathbf{x})]^{\;(1-y)}$$

$$P(y|x) = \hat{y}^{\;y} \, (1 - \hat{y})^{\;(1-y)}$$

$$log \; P(y|x) = y \; log \hat{y} + (1 - y) \; log(1 - \hat{y}) = -\, L\,(\hat{y}, y)$$

- Maximising the log of probabilities is minimising the loss

Evaluation Metrics : Cost Function

- **Binary Cross Entropy -** used with models that output a probability between 0 and 1, the cost function is given as below,

$$
\begin{aligned}
J(W) = log\prod_{n=1}^{n} P(y^{[i]}\,|\,x^{[i]}) &= -\frac{1}{n}\sum_{n=1}^{n} y^{[i]}log\left(\hat{y}^{[i]}\right) + (1-y^{[i]})log\left(1-\hat{y}^{[i]}\right)\\
&= -\frac{1}{n}\sum_{n=1}^{n} y^{[i]}log\left(f(x^{[i]};W)\right) + (1-y^{[i]})log\left(1-f(x^{[i]};W)\right)\\
&= -\frac{1}{n}\sum_{n=1}^{n} y^{[i]}log\left(\hat{p}^{[i]}\right) + (1-y^{[i]})log\left(1-\hat{p}^{[i]}\right)
\end{aligned}
$$

- Cost function is convex
- Use Gradient Descent to find the global minimum

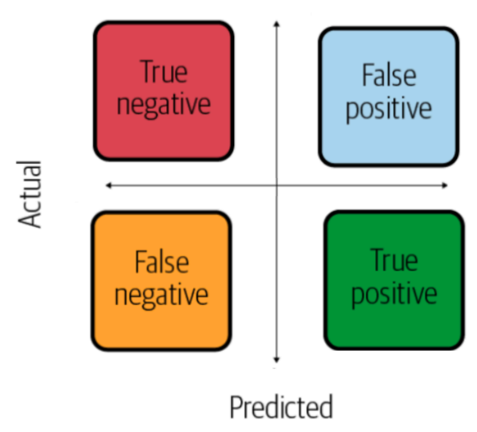
Confusion Matrix

- **Confusion Matrix**

$$\textbf{Precision} = \frac{\text{True positive}}{\text{Actual results}} \quad \text{or} \quad \frac{\text{True positive}}{\text{True positive} + \text{False positive}}$$

$$\textbf{Recall} = \frac{\text{True positive}}{\text{Predictive results}} \quad \text{or} \quad \frac{\text{True positive}}{\text{True positive} + \text{False negative}}$$

$$\textbf{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{Total}}$$



$$\textbf{F1 Score} = 2 * \frac{\textbf{Precision * Recall}}{\textbf{Precision + Recall}} = \frac{\textbf{TP}}{\textbf{TP + 0.5 * (FP+FN)}}$$

Adapted from Hariom Tatsat, Sahil Puri, and Brad Lookabaugh (2020), Machine Learning & Data Science Blueprints for Finance.

Accuracy Paradox

Predicted

Actual

Model - 1		
	0	1
0	5000	70
1	30	20

Predicted

Actual

Model - 2		
	0	1
0	20	30
1	70	5000

$$\text{Accuracy} = \frac{5000+20}{5000+20+70+30} = 98.05\%$$

Misclassification = 1 - Accuracy = 1.95%

Adapted from Motunrayo Olugbenga (2022), Balanced Accuracy: When Should You Use It?

Balanced Accuracy

- Machine Learning algorithms work best when the number of samples in each class are about equal.

  $$\text{Balanced Accuracy} = \frac{\text{sensitivity} + \text{specificity}}{2}$$

- Sensitivity is also known as true positive rate or recall that measures the proportion of real positives that are correctly predicted out of total positive prediction made by the model.
- $\text{Sensitivity} = \dfrac{TP}{(TP + FN)}$
- Specificity is also known as true negative rate, it measures the proportion of correctly identified negatives over the total negative prediction made by the model.
- $\text{Specificity} = \dfrac{TN}{(TN + FP)}$

Balanced Accuracy

Predicted

Actual

Model - 1		
	0	1
0	5000	70
1	30	20

$$\text{Sensitivity} = \frac{20}{20+30} = 40\%$$

$$\text{Specificity} = \frac{5000}{5000+70} = 98.62\%$$

$$\text{Balanced Accuracy} = \frac{40 + 98.62}{2} = 69.46\%$$

- A model can have high accuracy with bad performance, or low accuracy with better performance, which can be related to the accuracy paradox.
- Accuracy can be a useful measure only if we have a balanced dataset.

Adapted from Motunrayo Olugbenga (2022), Balanced Accuracy: When Should You Use It?

Class Imbalances

- Machine Learning algorithms work best when the number of samples in each class are about equal.
- Observation in one class is higher than the observation in other classes leads to class imbalances - which is a common problem in Machine Learning.
- Most algorithms are designed to maximize accuracy and reduce errors.
- Imbalance data can hamper model accuracy big time.
  - high accuracy by predicting the **majority class** (not that model is any good)
  - fail to capture the **minority class**
- Algorithm selection, under/over sampling and penalising algorithms are some of the most common approaches to address imbalance in data.

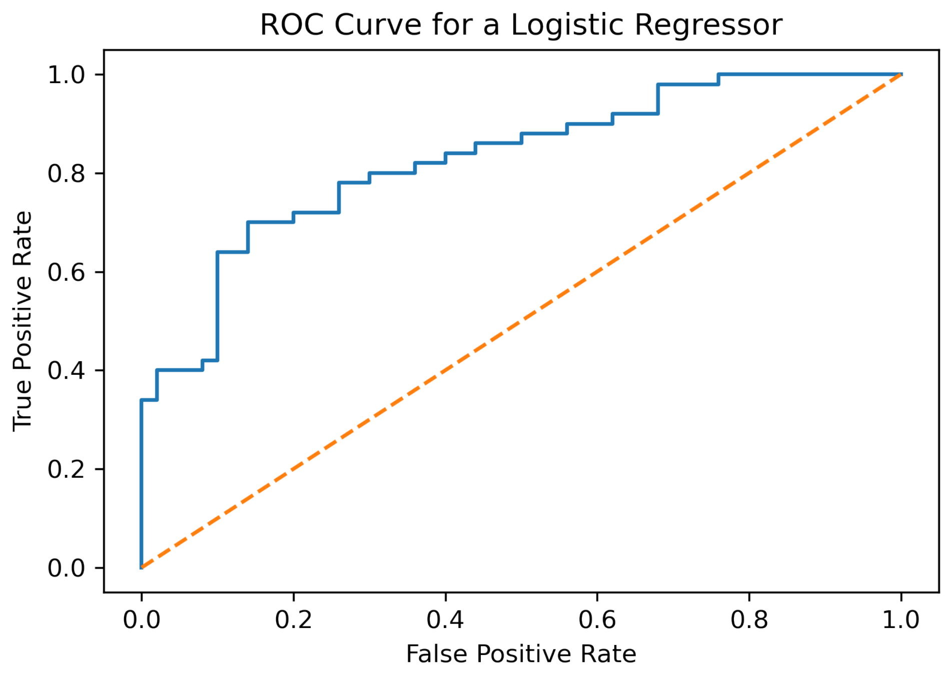
Refer Advanced Machine Learning Workshop for detailed discussion on class imbalances.

ROC Curves

- Receive Operating Characteristic curve is a plot of the false positive rate versus the true positive rate.
- True positive rate is the recall and describes the goodness of the model in predicting the positive class when the actual outcome is positive. It is also referred to as sensitivity.
- False positive rate is defined as the inverted specificity (ie., 1-Specificity).
- ROC curves are used where there is equal number of observations of each class.

ROC Curves

- The area under the ROC curve (AUC) is a measure of how well a model can distinguish between two classes.



Multiclass Classification

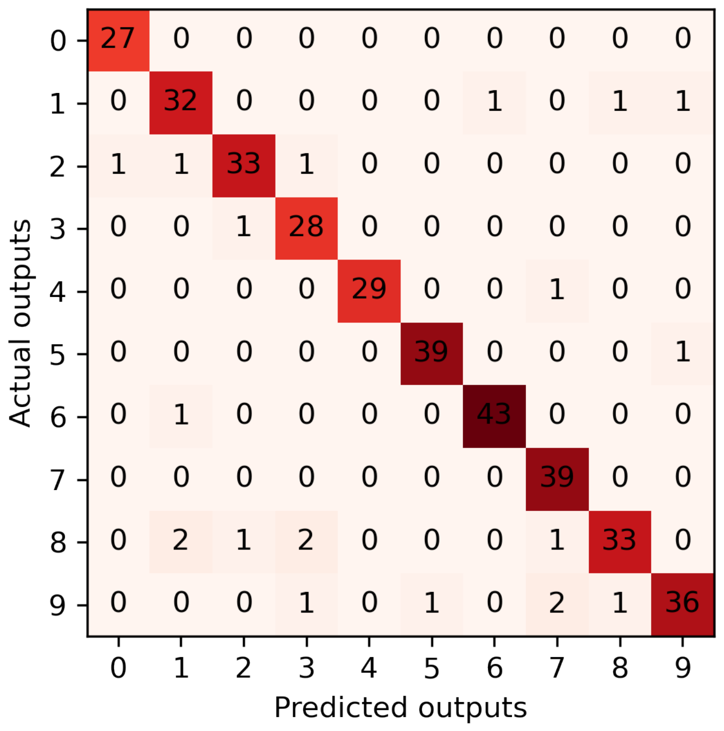
- Logistic Regression model can be extended to multiple classes.
- Given an instance x, the model computes a score $s_k(x)$ for each k,
- Probability of each class is then estimated by applying softmax function (normalised exponential) to the scores
- Mathematically, this is represented as

$\hat{p}_k \frac{f_k}{\sum_k f_k}$, where f_k is the output of classifier k and $\hat{p}_k$ is the probability of the observation being in class k and sum of $\hat{p}_k$ for all the values of k is equal to 1.

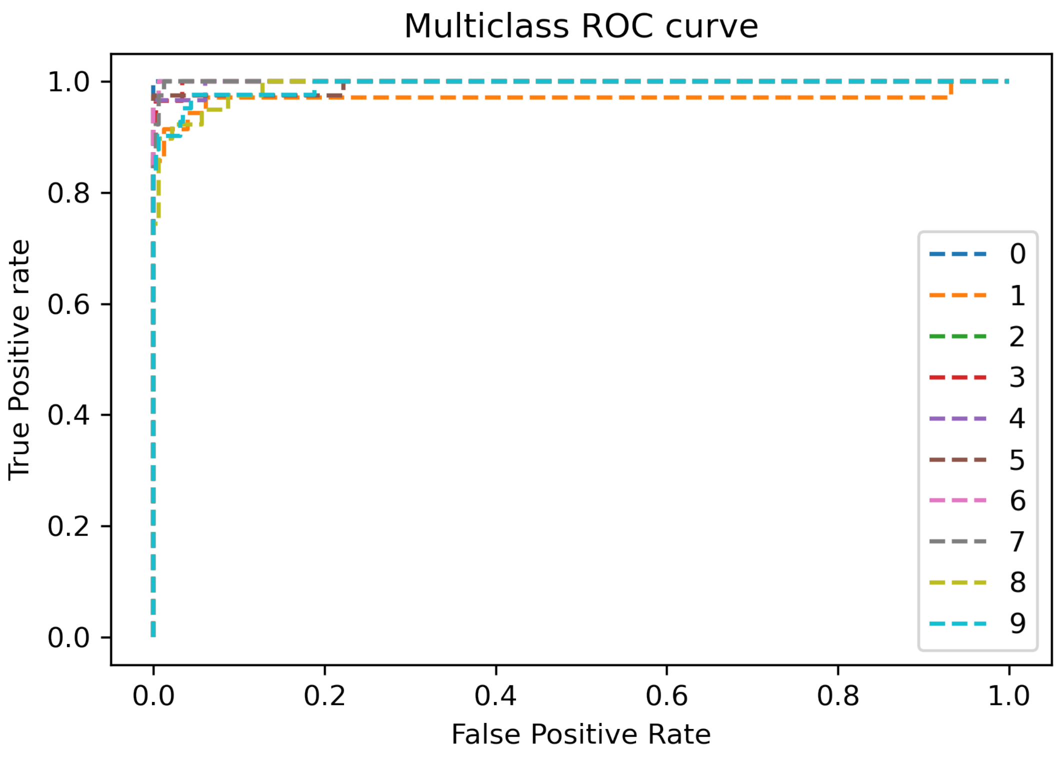
$\hat{p}_k = \sigma(s(x))_k = \frac{exp^{s_k(x)}}{\sum_{j=1}^{K} exp^{s_j(x)}}$ and for k classes, we train $\frac{k(k-1)}{2}$ classifiers.

Digit Classification

- Image recognition of handwritten digits.



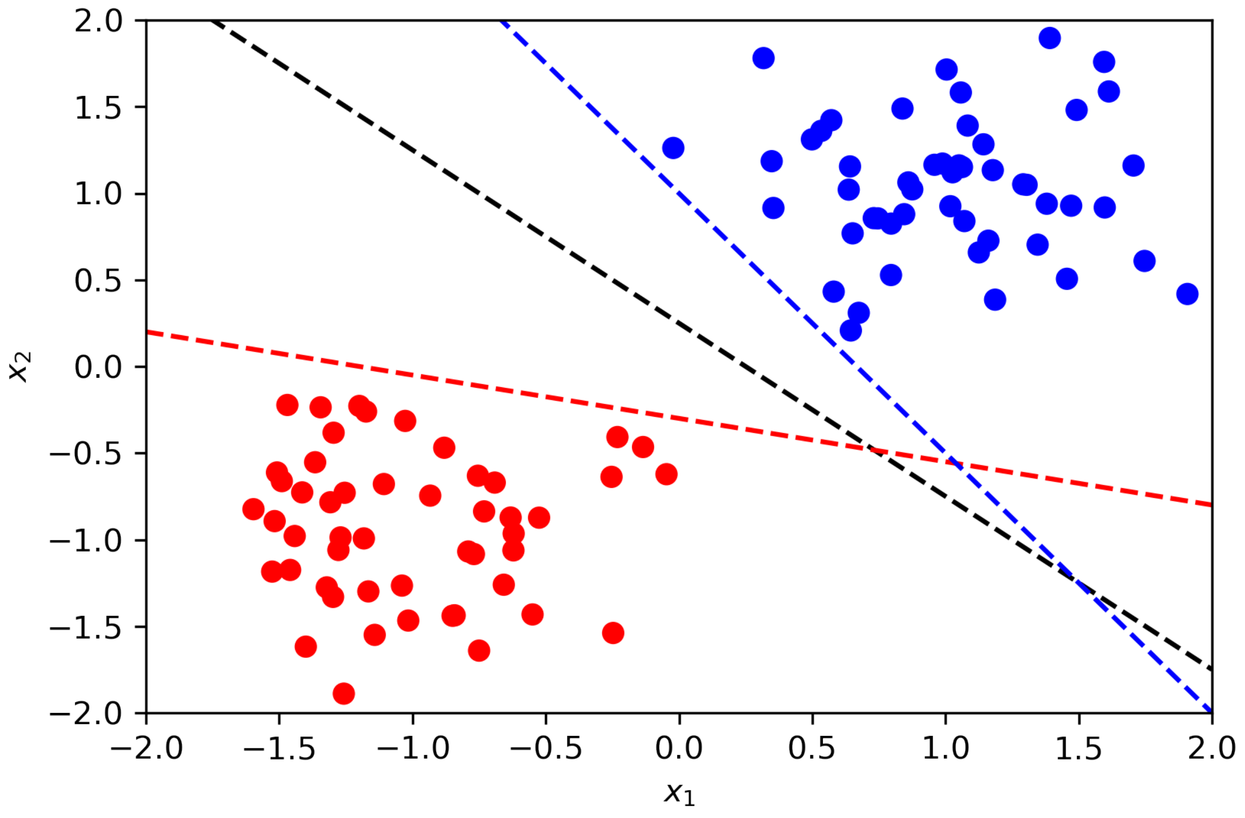
Actual outputs \ Predicted outputs	0	1	2	3	4	5	6	7	8	9
0	27	0	0	0	0	0	0	0	0	0
1	0	32	0	0	0	0	1	0	1	1
2	1	1	33	1	0	0	0	0	0	0
3	0	0	1	28	0	0	0	0	0	0
4	0	0	0	0	29	0	0	1	0	0
5	0	0	0	0	0	39	0	0	0	1
6	0	1	0	0	0	0	43	0	0	0
7	0	0	0	0	0	0	0	39	0	0
8	0	2	1	2	0	0	0	1	33	0
9	0	0	0	1	0	1	0	2	1	36



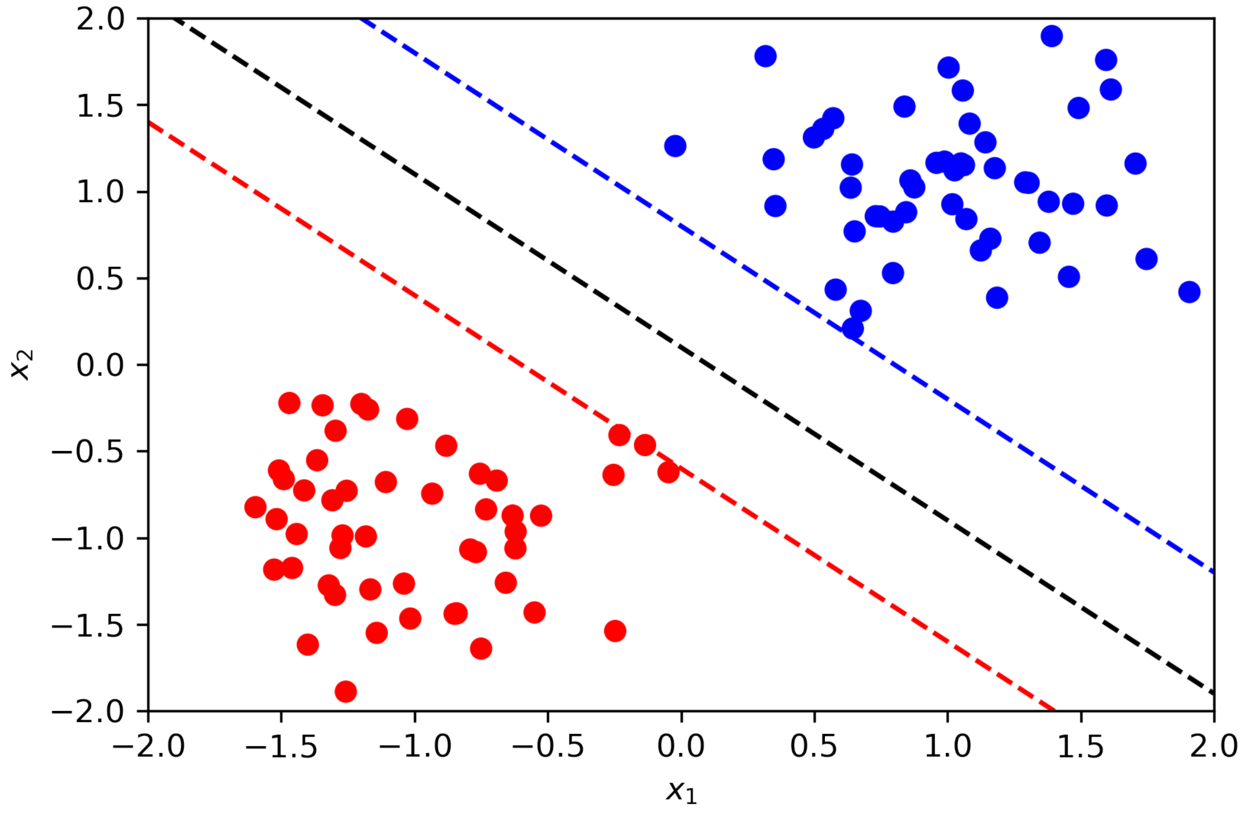
Support Vector Network

- One of the popular Machine Learning algorithms.
- Supervised Machine Learning for
  - Linear and non-linear classification
  - Regression and classification problem
- Work well with small or medium sized datasets.
- Maximal Margin Classifier
- Support Vector Classifier
- Support Vector Machine

The Hyperplane



Maximal Margin



Maximal Margin

Maximal Margin

Maximal Margin

Hard Margin Classifier

- Maximise the width of the margin is equal to

$$min \frac{1}{2} \|\mathbf{w}\|^2$$

s.t

$$y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) - 1 \geq 0$$

Support Vector Classifier

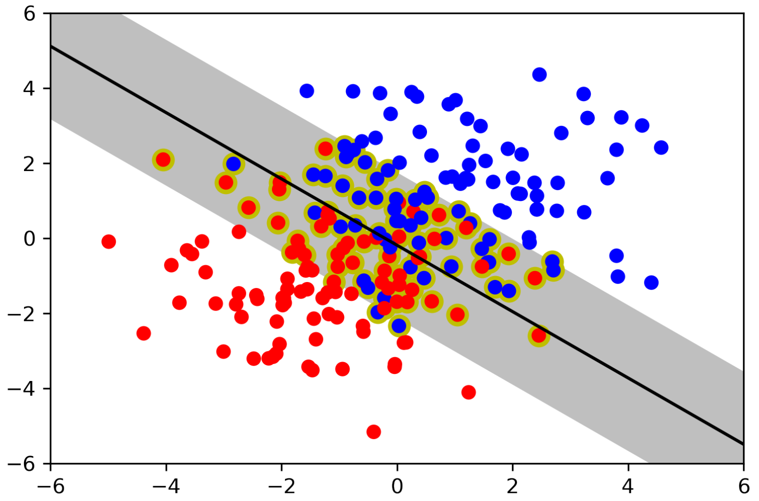
- Also called as soft margin classifier

$$min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=i}^{n} \zeta_i$$

subject to,

$$y_i \left(\mathbf{w} \cdot \mathbf{x_i} + b\right) \geq (1 - \zeta_i)$$

where $\zeta_i \geq 0$ and C is a non-negative tuning parameter



- $\zeta_i = 0$ when vector does not reside inside the margin and is on the correct side of the hyperplane.
- $0 > \zeta_i < 1$ when vector reside inside the margin.
- $\zeta_i = 1$ when vector resides on the hyperplane.
- $\zeta_i > 1$ when vector is on the wrong side of the hyperplane.

Support Vector Classifier

- Choose slack variables as

$$\zeta_i = max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x_i} + b))$$

- We can rewrite SVC formulation as a simplified optimization problem

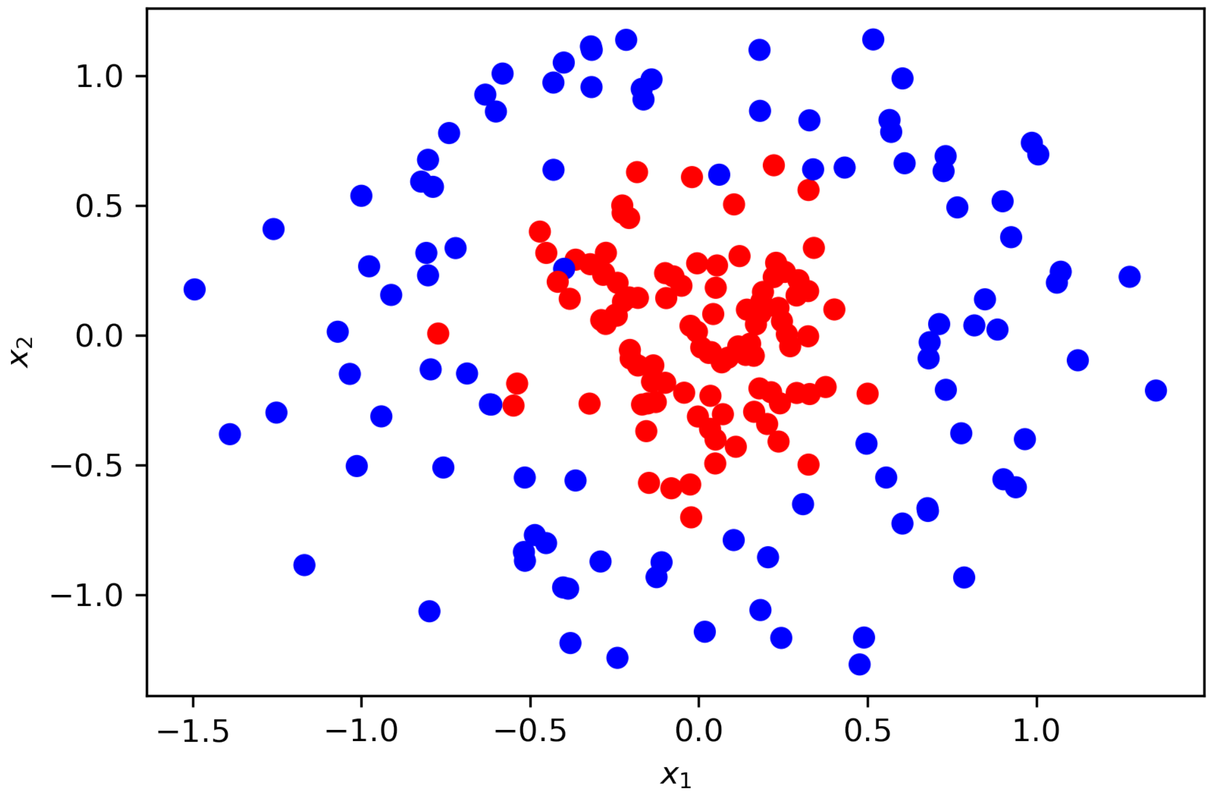
$$min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=i}^{n} max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x_i} + b))$$

- SVC (soft margin) can be formulated with hinge loss

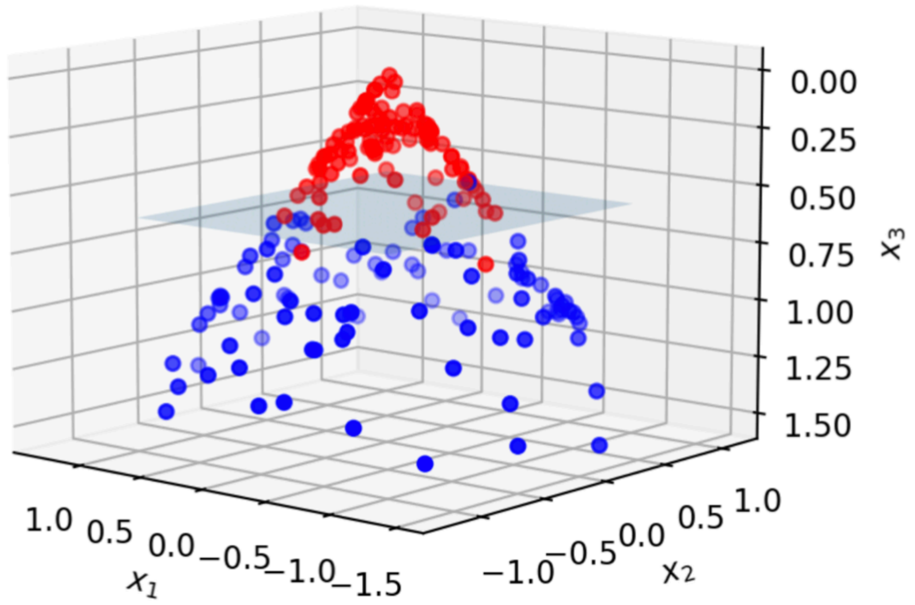
Support Vector Machines

- Extension of support vector classifier.
- Enlarge features space using kernels.
- Accommodate non-linear boundaries between the classes.
- Implicitly create features to project into higher dimensional space.
- Hyperplane divide the classes in the higher dimension space.
- Decision boundary on the original dimensional space is non-linear.

Support Vector Machines



Support Vector Machines



Lagrangian Dual Formulation

- In finding extremum of a function with constraints, we can use Lagrange multiplier.
- Lagrange multiplier will give us new expression which we can maximise or minimise without thinking about the constraints.
- Basic idea is to convert a constrained problem into a form such that derivative (can still be applied) of an unconstrained problems.
- Finding the extremum means, find ∇ and set them to 0.

Lagrangian Dual Formulation

Lagrangian Dual Formulation

Lagrangian Dual Formulation

Lagrangian Dual Formulation

- Step 1 : Apply method of Lagrange multipliers

$min \frac{1}{2}\|\mathbf{w}\|^2$ subject to constraints $y_i\,(\mathbf{w}\cdot\mathbf{x_i}+b)-1\geq 0$

- Step 2 : Differentiate vectors

$\frac{\partial\|\mathbf{w}\|^2}{\partial\mathbf{w}}=2\mathbf{w}$ and $\frac{\partial\mathbf{x}\cdot\mathbf{w}}{\partial\mathbf{w}}=\mathbf{x}$

- Step 3 : Find derivatives of the Lagrangian L

$$\frac{\partial L}{\partial\mathbf{w}}=\mathbf{w}-\sum_{i}^{l}\alpha_i y_i\mathbf{x_i}=0$$

$$\frac{\partial L}{\partial b}=\sum_{i}^{l}\alpha_i y_i=0$$

Lagrangian Dual Formulation

- Step 4 : Do the algebra

$$L = \sum_{i}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}$$

- Minimisation depends only on dot products of sample vectors

Kernel Trick

- Kernel is a function that creates the implicit mapping to a higher dimensional space.
- Transform non-linear on $\mathbf{x}$ into linear on $\phi(\mathbf{x})$.
- No explicit feature generation (hence no increase in the size of dataset).
- Solution involves only the inner products of the observations of the general form $K(x_i, x_j)$ which is equal to $\phi(\mathbf{x_i}) \cdot \phi(\mathbf{x_j})$
- Choice of kernels for best fit hyperplane.

Linear	$(\mathbf{x_i} \cdot \mathbf{x_j})$
Polynomial	$(r + \mathbf{x_i} \cdot \mathbf{x_j})^d$
Gaussian RBF	$e^{(-\gamma \lVert \mathbf{x_i} - \mathbf{x_j} \rVert^2)}$
Sigmoid	$tanh(\gamma \mathbf{x_i} \cdot \mathbf{x_j} + r)$

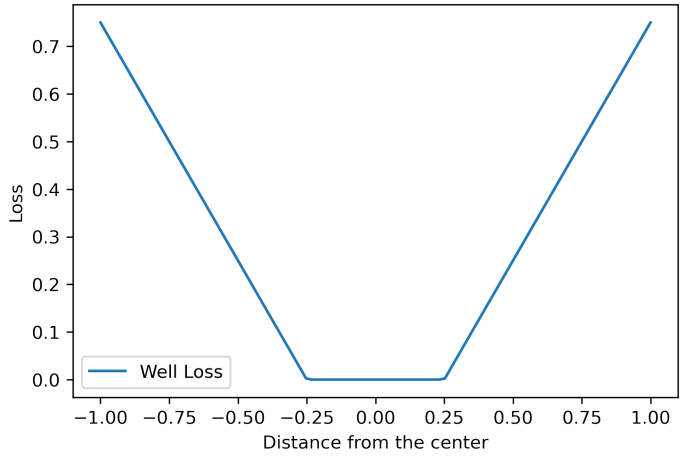
SVM for Regression

- Train model that includes as many vectors as possible inside the margin.
- No penalty for vectors that reside inside the margin.
- Penalise for vectors that lie outside the margin.
- Cost function would be the well-loss.
- Optimisation of width takes the form of

$$min\frac{1}{2}\|W\|^2 + C\sum_{i=i}^{n}(\zeta_i + \zeta_i^*)$$

subject to,

$$y_i - \mathbf{w}\cdot\mathbf{x_i} - b \le \epsilon + \zeta_i$$
$$\mathbf{w}\cdot\mathbf{x_i} + b - y_i \le \epsilon + \zeta_i^*$$
$$\zeta_i, \zeta^* \ge 0$$



KNN

- One of the simplest ML algorithms used for both regression and classification.
- Lazy algorithm as it doesn’t technically train a model to make a predictions.
- Observed is predicted to be in the class of that of the largest proportion of k- nearest observations.
- Direct attempt at approximating the conditional expectation using actual data.

KNN

- Non-parametric method and and it does not assume any function as there are no parameters to be estimated.
  - k close to 1 gives the most flexibility (low bias), but highest variability (high variance)
  - Large k gives the least flexibility (high bias), but lowest variability (low variance)
  - $k = N$ will assign all new test observations to a single class.
- Odd number k is preferred for binary classification.
- Best way to select k is through
  - Cross validation
  - Elbow method

KNN Regression

- For regression, the predicted value is the mean of K neighbours and the estimator is given as

$$\hat{f}(x) = Average\left[y_i \mid x_i \in \mathcal{N}_k(x)\right]$$

where, $\mathcal{N}_k(x)$ is a neighbourhood of x containing the k closest observations.

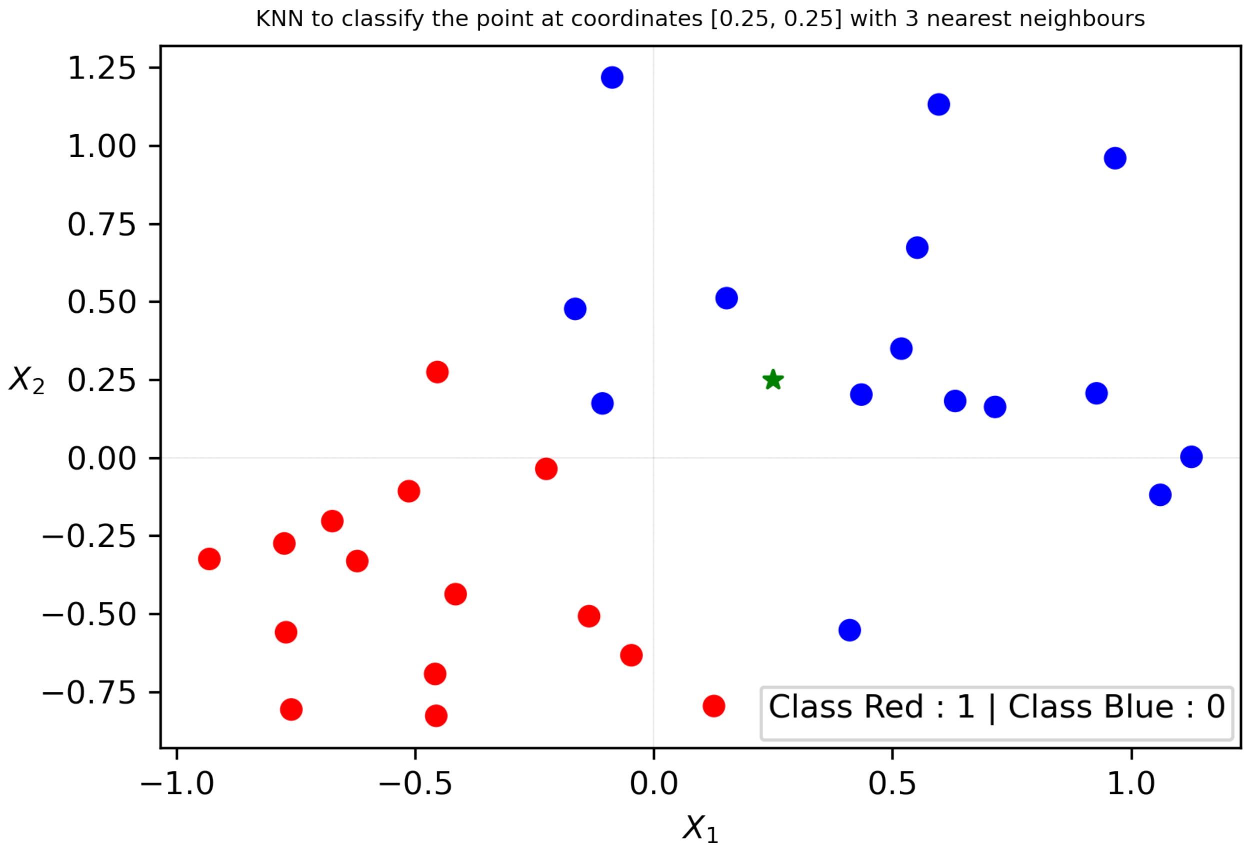
KNN Classification

- For classification, the predicted value is the class with the plurality, i.e., the class most represented among the neighbours.
- This is equivalent to taking a majority vote among the k nearest neighbours.
- For each class $j = 1, \cdots, K$, we then compute the empirical (conditional) probability

$$Pr(G = j \mid X = x_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} I(y_i = j)$$

  and assign the observation to the class with the highest empirical probability. Here, $I(y_i = j)$ is the indicator function returning 1 if $y_i = j$ and 0 otherwise.
- KNN output would be biased if the dataset is imbalanced or on presence of outliers.

KNN Example


KNN to classify the point at coordinates [0.25, 0.25] with 3 nearest neighbours
Class Red : 1 | Class Blue : 0
X_1
X_2
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.0
−0.5
0.0
0.5
1.0

The Three Steps

- Once, k is specified, finding nearest neighbours is a 3-step process.
  - Compute the distance, usually Euclidian
  - Sort by ascending distance to the find the k NN
  - Compute the average or probability of k NN observations

Nearest Neighbours

- KNN uses distance as the metric in determining its neighbours.
- Distance can be thought of as a measure of similarity.
- Generalised distance metric is called the Minkowski distance

$$d = \left(\sum_{n=i}^{n} \left| x_i - y_i \right|^p \right)^{1/p}$$

where x_i and y_i are the two observations for which distance d is being calculated with a hyper parameter, integer p.

- $p = 1$ is the Manhattan distance
- $p = 2$ is the Euclidean distance

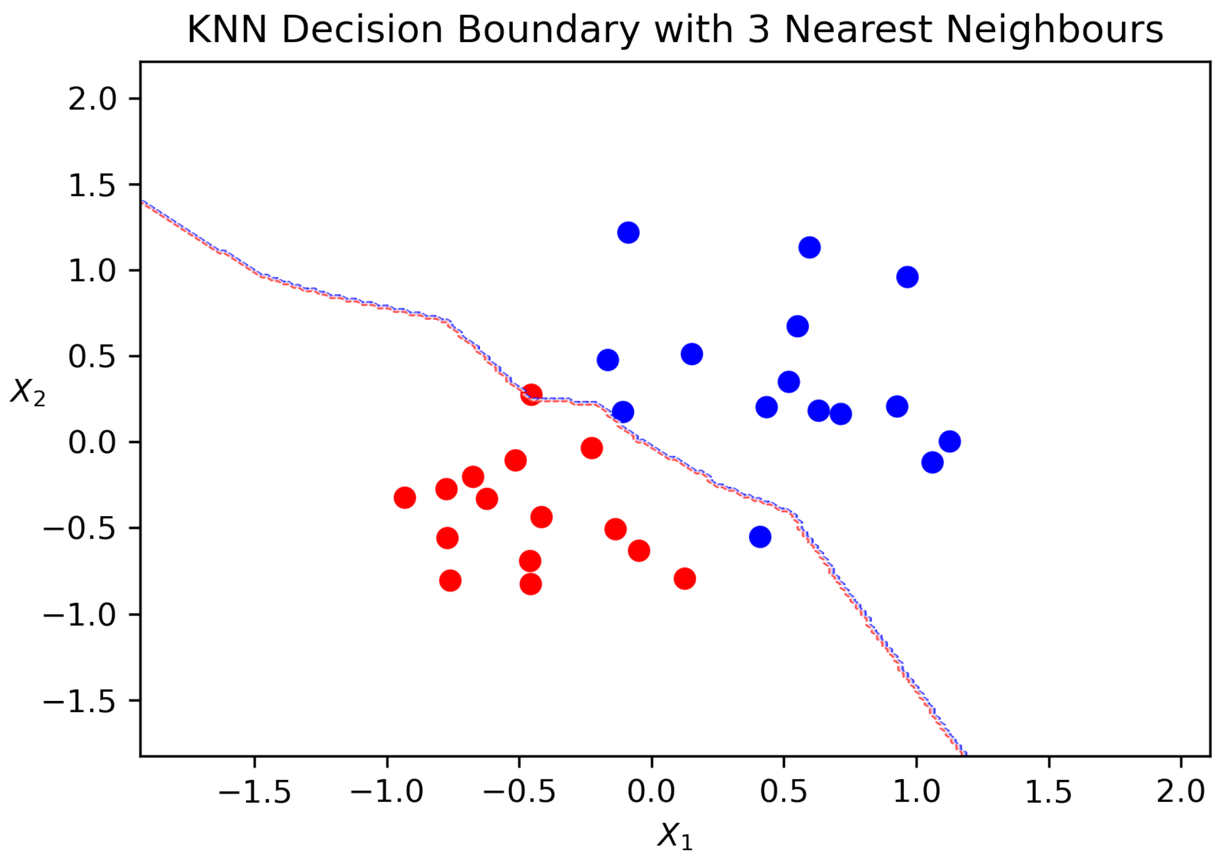
KNN Example

	X1	X2	Y	Class	Distance
0	-0.621282	-0.331191	1	Red	1.047337
1	0.410545	-0.552438	0	Blue	0.818341
2	-0.046838	-0.632821	1	Red	0.931389
3	-0.455832	-0.826784	1	Red	1.287502
4	-0.087054	1.217604	0	Blue	1.024628
5	-0.135597	-0.507716	1	Red	0.850188
6	0.518747	0.349231	0	Blue	0.286482
7	1.061250	-0.119117	0	Blue	0.891276
8	-0.225160	-0.035936	1	Red	0.554560
9	0.152672	0.510864	0	Blue	0.278429
10	-0.453268	0.274211	1	Red	0.703685
11	0.631439	0.181521	0	Blue	0.387537
12	-0.513032	-0.107380	1	Red	0.842579
13	0.597201	1.131221	0	Blue	0.947153
14	-0.164562	0.476409	0	Blue	0.472359
15	0.434864	0.202033	0	Blue	0.190986
16	-0.415179	-0.437181	1	Red	0.956390
17	0.966692	0.959089	0	Blue	1.008194
18	-0.770452	-0.559163	1	Red	1.302331
19	-0.931161	-0.324453	1	Red	1.313445
20	-0.107496	0.173775	0	Blue	0.365532
21	-0.458115	-0.692723	1	Red	1.179048
22	0.927720	0.206114	0	Blue	0.679139
23	-0.673497	-0.203217	1	Red	1.028714
24	-0.759423	-0.806535	1	Red	1.461233
25	0.126097	-0.795368	1	Red	1.052686
26	0.552044	0.672569	0	Blue	0.519418
27	-0.774020	-0.274782	1	Red	1.150658
28	1.126512	0.002594	0	Blue	0.910760
29	0.715034	0.162489	0	Blue	0.473197

	X1	X2	Y	Class	Distance
15	0.434864	0.202033	0	Blue	0.190986
9	0.152672	0.510864	0	Blue	0.278429
6	0.518747	0.349231	0	Blue	0.286482
20	-0.107496	0.173775	0	Blue	0.365532
11	0.631439	0.181521	0	Blue	0.387537
14	-0.164562	0.476409	0	Blue	0.472359
29	0.715034	0.162489	0	Blue	0.473197
26	0.552044	0.672569	0	Blue	0.519418
8	-0.225160	-0.035936	1	Red	0.554560
22	0.927720	0.206114	0	Blue	0.679139
10	-0.453268	0.274211	1	Red	0.703685
1	0.410545	-0.552438	0	Blue	0.818341
12	-0.513032	-0.107380	1	Red	0.842579
5	-0.135597	-0.507716	1	Red	0.850188
7	1.061250	-0.119117	0	Blue	0.891276
28	1.126512	0.002594	0	Blue	0.910760
2	-0.046838	-0.632821	1	Red	0.931389
13	0.597201	1.131221	0	Blue	0.947153
16	-0.415179	-0.437181	1	Red	0.956390
17	0.966692	0.959089	0	Blue	1.008194
4	-0.087054	1.217604	0	Blue	1.024628
23	-0.673497	-0.203217	1	Red	1.028714
0	-0.621282	-0.331191	1	Red	1.047337
25	0.126097	-0.795368	1	Red	1.052686
27	-0.774020	-0.274782	1	Red	1.150658
21	-0.458115	-0.692723	1	Red	1.179048
3	-0.455832	-0.826784	1	Red	1.287502
18	-0.770452	-0.559163	1	Red	1.302331
19	-0.931161	-0.324453	1	Red	1.313445
24	-0.759423	-0.806535	1	Red	1.461233

```
Classification
[0.0, 100.0]
```

KNN Example


KNN Decision Boundary with 3 Nearest Neighbours
X_1
X_2
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Hyper-parameter Tuning

- Parameters that are not directly learnt within estimators.
- Passed as arguments to the constructor of the estimator classes (in scikit-learn).
- Recommended to search the hyper-parameter space for the best cross validation score.
- Typical examples include tuning
  - Alpha for Lasso, Ridge
  - C, kernel and gamma for SVC
  - n_neighbors for KNN

Hyper-parameter Tuning

- Some of the popular approaches to hyper parameter tuning are
  - Manual
  - Grid Search
  - Random Search

GridSearch

- The conventional way of performing hyper-parameter optimisation.
- Exhaustive search through a manually specified subset of the hyper-parameter space of a learning algorithm.
- Algorithm must be guided by some performance metric
  - cross-validation on the training set
  - evaluation on a validation set
- Implements a “fit” and a “score” method among other methods.
- The parameters of the estimator used to apply these methods are optimised by cross-validated grid-search over a parameter grid.

RandomSearch

- Randomised search narrow down the results.
- Random combinations of the hyper-parameters are used to find the best parameters.
- Yields high variance during computing
- Works best under the assumption that not all hyperparameters are equally important.
- Given the random search pattern, changes of finding the optimal parameter are comparatively higher.

References

- Scott Fortmann-Roe (2012) , Understanding the Bias-Variance Tradeoff
- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani (2013), An Introduction to Statistical Learning
- Sebastian Raschka (2015), Python Machine Learning
- Francois Chollet (2017), Deep Learning with Python
- Aurelien Geron (2019), Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
- Hariom Tatsat, Sahil Puri, and Brad Lookabaugh (2020), Machine Learning & Data Science Blueprints for Finance
- Motunrayo Olugbenga (2022), Balanced Accuracy: When Should You Use It?
- Scikit-learn User Guide
- Google Developers, Introduction to Machine Learning