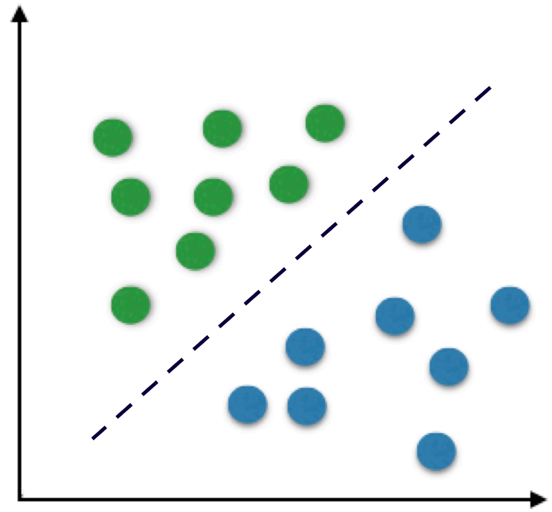


COMP9517: Computer Vision

Pattern Recognition Part 2

Pattern Recognition (First Lecture)

- Pattern recognition concepts
  - Definition and description of basic terminology
  - Recap of feature extraction and representation

- Supervised learning for classification
  - Nearest class mean classification
  - K-nearest neighbours classification
  - Bayesian decision theory and classification
  - Decision trees for classification
  - Ensemble learning and random forests

Pattern Recognition (Second Lecture)

- Supervised learning for classification
  - Linear classification
  - Support vector machines
  - Multiclass classification
  - Classification performance evaluation
- Supervised learning for regression
  - Linear regression
  - Least-squares regression
  - Regression performance evaluation

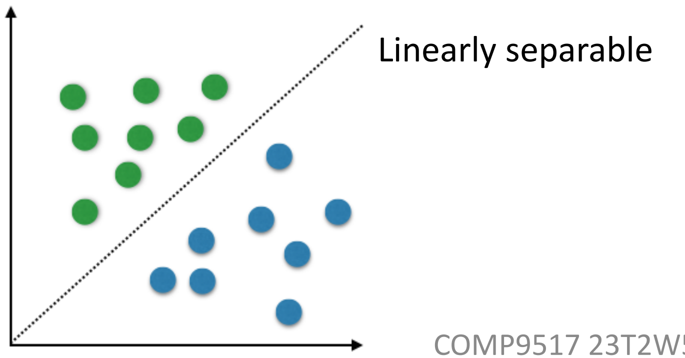
Separability

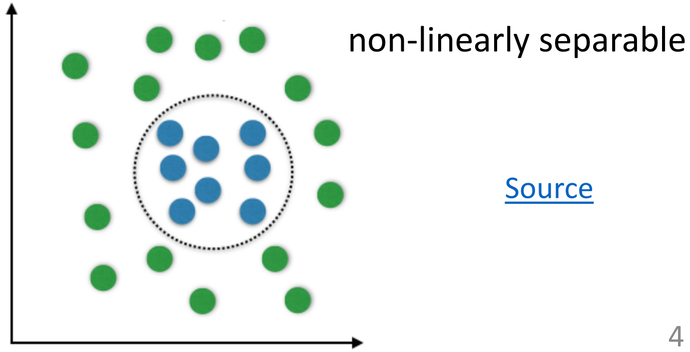
- **Separable classes**

  If a discrimination subspace exists that separates the feature space such that only objects from one class are in each region, then the recognition task is said to have separable classes

- **Linearly separable**

  If the object classes can be separated using a hyperplane as the discrimination subspace, the feature space is said to be linearly separable





Source

Linear Classifier

- Given a training set of N observations:

$$\{(x_i, y_i)\}, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1,1\}, \quad i = 1, \dots, N$$

- A binary classification problem can be modeled by a separation function $f(x)$ using the data such that:

$$f(x_i) = \begin{cases} > 0 \text{ if } y_i = +1 \\ < 0 \text{ if } y_i = -1 \end{cases}$$

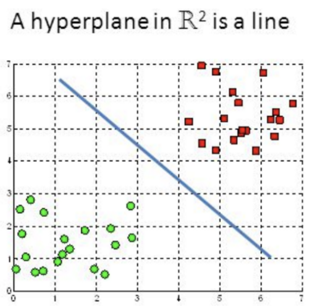
- So in this approach $y_i f(x_i) > 0$

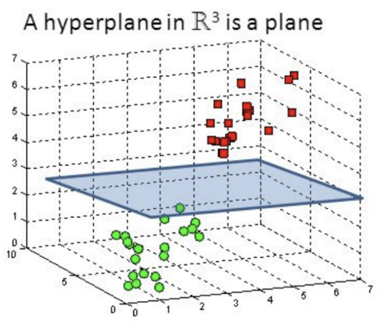
Linear Classifier

- A linear classifier has the form:

$$f(x) = W^T x + b = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

- Corresponding to a line in 2D, a plane in 3D, and a hyperplane in nD



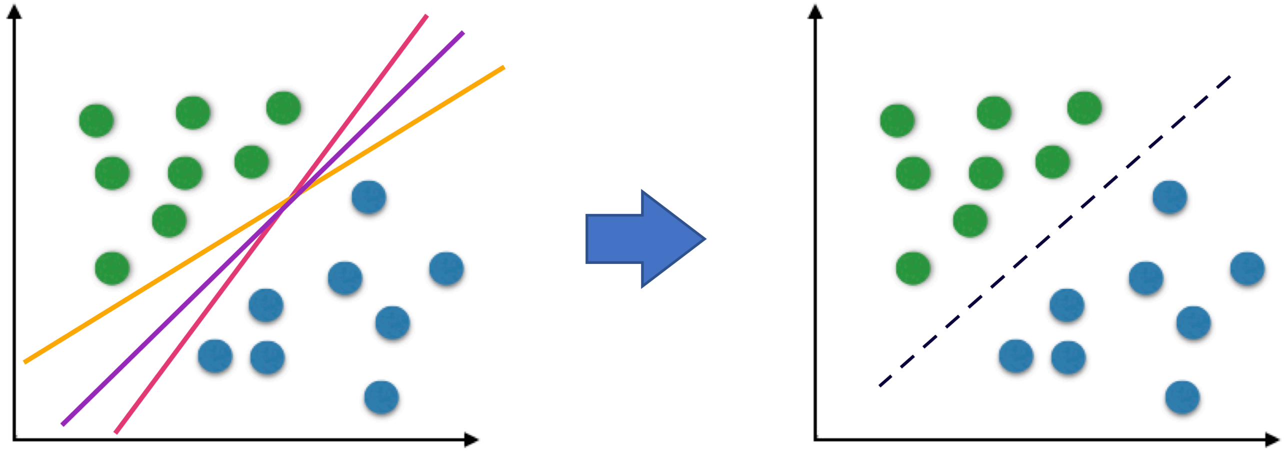


Source

- We use the training data to learn the weights W and offset b
- x_i are features

Linear Classifier

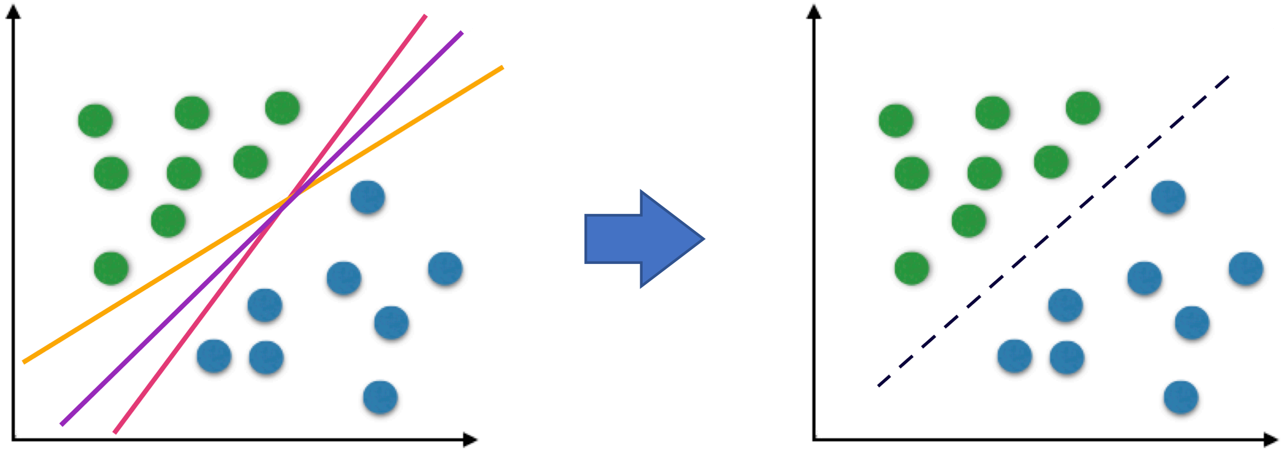
- Which hyperplane is the best…?

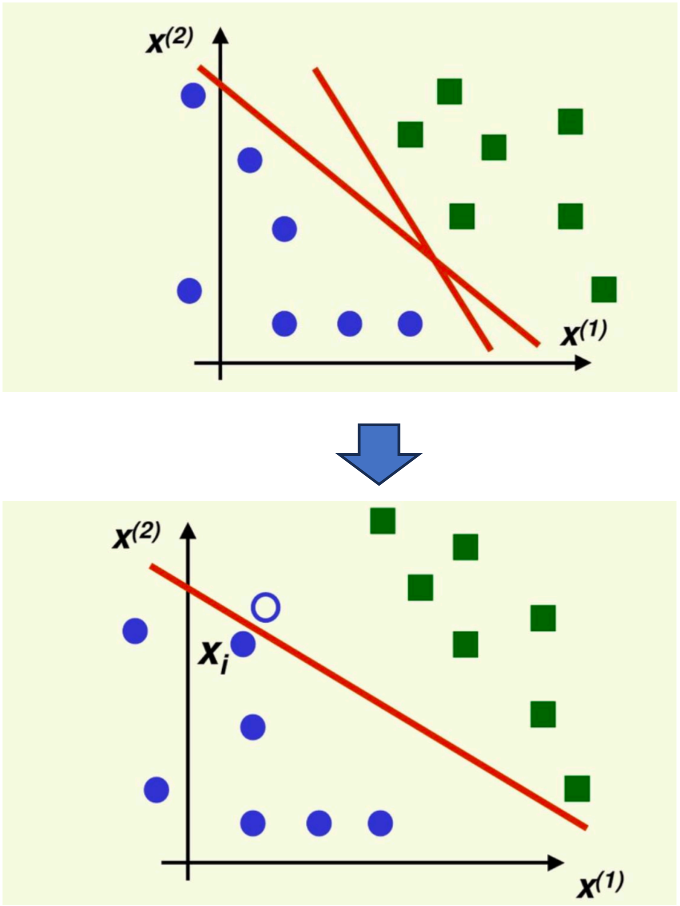


- For generalization purposes, a large margin is preferred
- Good generalization

Linear Classifier

- Which hyperplane is the best…?



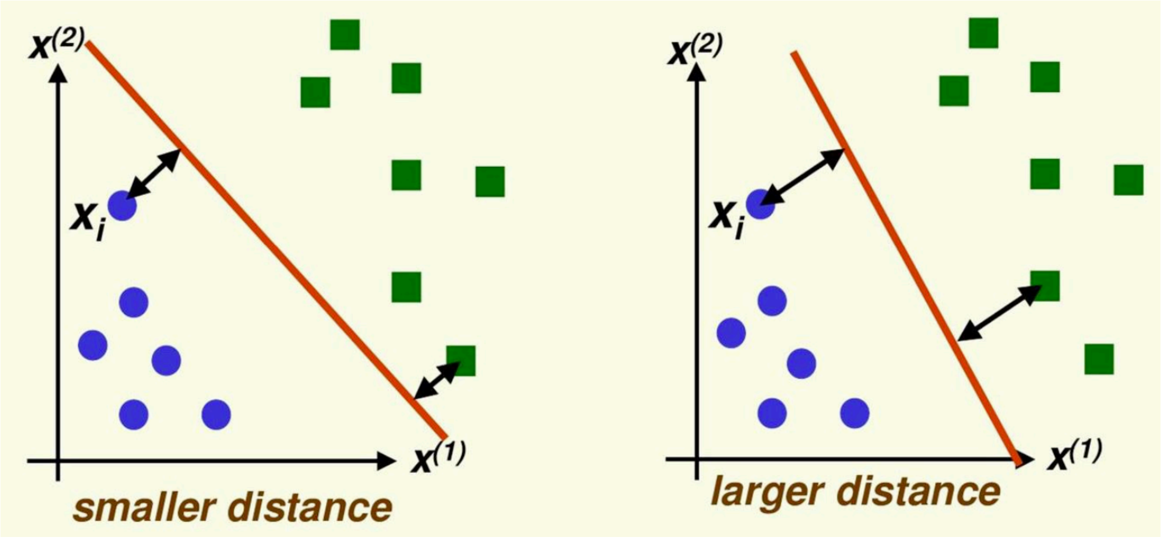


Bad generalization

- For generalization purposes, a large margin is preferred
- Good generalization

Support Vector Machines (SVMs)

- Maximize margin - the distance to the closest sample
  - Leads to an optimization problem
- Examples closest to the hyperplane are support vectors



Support Vector Machines

- The primal optimization problem for linear SVM (Hard-margin SVMs)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

- Decision rules in testing

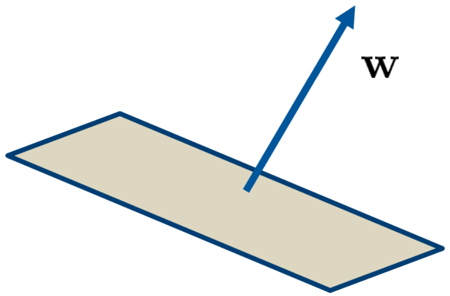
$$\hat{y} = 1 \quad if \quad \mathbf{w}^\top \mathbf{x} + b > 0$$
$$\hat{y} = -1 \quad if \quad \mathbf{w}^\top \mathbf{x} + b < 0$$

- Why?

Support Vector Machines – some preliminaries

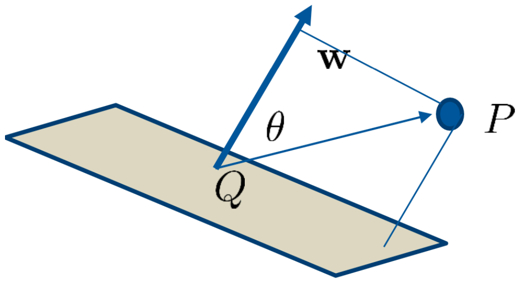
- Hyperplane (in the high-dimensional space) defined bv a linear model

$$\mathbf{w}^\top \mathbf{x} + b = 0$$



- Distance between a point to a hyperplane

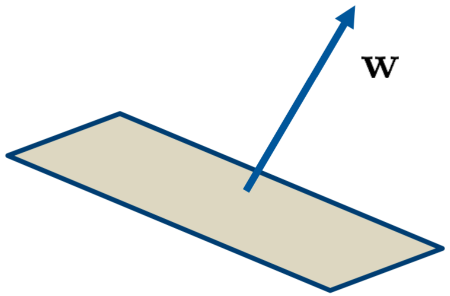
$$d = \frac{|\mathbf{w}^\top \mathbf{x}' + b|}{\|\mathbf{w}\|_2}$$



Support Vector Machines – some preliminaries

- Hyperplane (in the high-dimensional space) defined bv a linear model

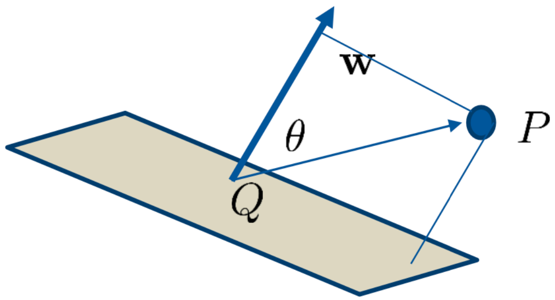
$$\mathbf{w}^\top \mathbf{x} + b = 0$$



- Distance between a point to a hyperplane

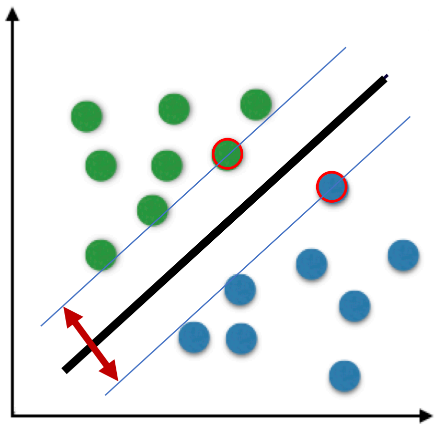
$$d = \frac{|\mathbf{w}^\top \mathbf{x}' + b|}{\|\mathbf{w}\|_2}$$

Proof:



$$\begin{aligned} d &= |PQ \cos(\theta)| \\ &= | \|(\mathbf{x}' - \mathbf{x})\|_2 \frac{\mathbf{w}^\top (\mathbf{x}' - \mathbf{x})}{\|\mathbf{w}\|_2 \|(\mathbf{x}' - \mathbf{x})\|_2} | \\ &= \frac{|\mathbf{w}^\top \mathbf{x}' - \mathbf{w}^\top \mathbf{x}|}{\|\mathbf{w}\|_2} = \frac{|\mathbf{w}^\top \mathbf{x}' - (\mathbf{w}^\top \mathbf{x} + b) + b|}{\|\mathbf{w}\|_2} = \frac{|\mathbf{w}^\top \mathbf{x}' + b|}{\|\mathbf{w}\|_2} \end{aligned}$$

Support Vector Machines



- SVM objective
  - maximize the distance from hyperplane to the closest examples
  - positive class and negative class samples are on each side of the hyperplane

$$\max_{\mathbf{w},b,\eta} \eta$$

$$s.t. \quad \frac{|\mathbf{w}^\top \mathbf{x}_i + b|}{\|\mathbf{w}\|_2} \geq \eta$$

The distance between the hyperplane and the closest examples

$$\mathbf{w}^\top \mathbf{x}_i + b \geq 0 \quad if \quad y_i > 0$$

$$\mathbf{w}^\top \mathbf{x}_i + b \leq 0 \quad if \quad y_i < 0$$

For correct classification

- This problem can be equivalently reformulated as:
  - The "standard" formulation of (hard-margin) linear SVM

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

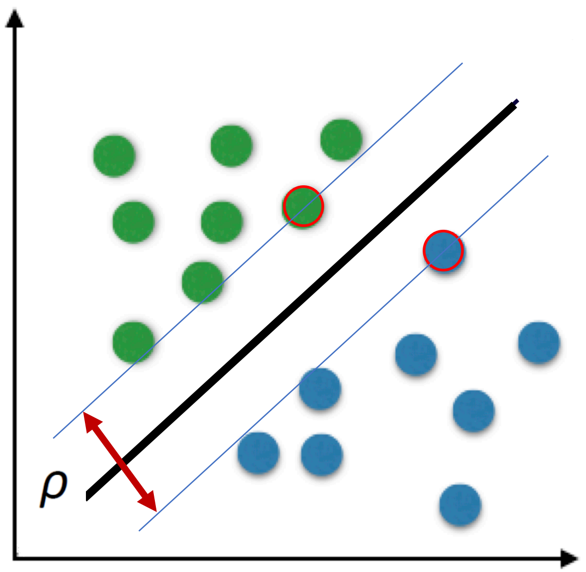
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

Support Vector Machines

- Hard-margin linear SVM

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2$$

$$s.t.\quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$



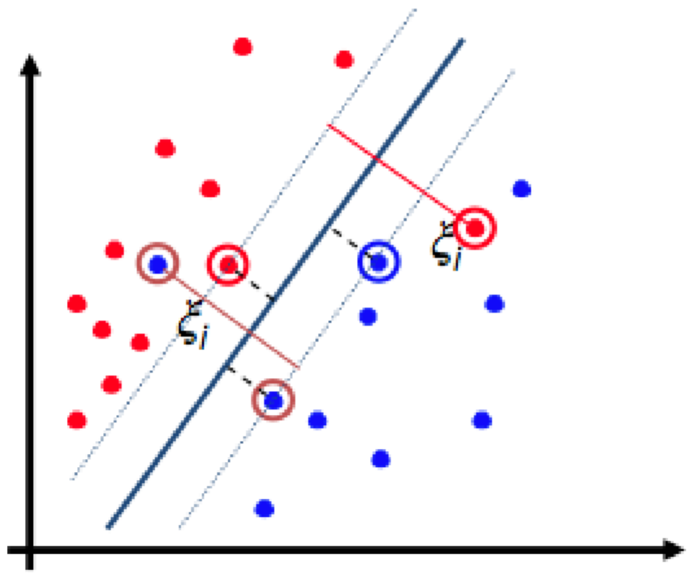
- Margin: $\rho = \frac{1}{\|\mathbf{w}\|_2}$
- All the support vectors are in

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) = 1$$

- Quadratic programming optimization problem subject to linear constraints
  - Convex optimization problem
  - With a dual form from Lagrangian method
- **hard margin SVM** which does not allow any misclassification of samples

Soft Margin Support Vector Machines

- In hard margin SVM, we assume classes are linearly separable, but what if separability assumptions doesn't hold?



ξ_i is the distance of x_i to the corresponding class margin if on the wrong side of the margin, or 0 otherwise

- Introduce "slack" variables ξ_i to allow misclassification of instances

Soft Margin Support Vector Machines

When classes were linearly separable, we had:

$$y_i(W^T x_i + b) \geq 1$$

But if we get some data that violate this slack value:

$$y_i(W^T x_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0$$

So, the total violation for all data is $\sum_i \xi_i$

This is a measure of violation of the margin and now we optimize for:

$$\min_{\mathbf{w}, b, \{\xi_i\}} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_i \xi_i$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i$$
$$\xi_i \geq 0$$

:a that violate this slack value:

$y_i(W^T x_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$

for all data is $\sum_i \xi_i$

olation of the margin and now we optin

$$\min_{\mathbf{w}, b} \frac{1}{2}\|\mathbf{w}\|_2^2$$
$$s.t. \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i$$

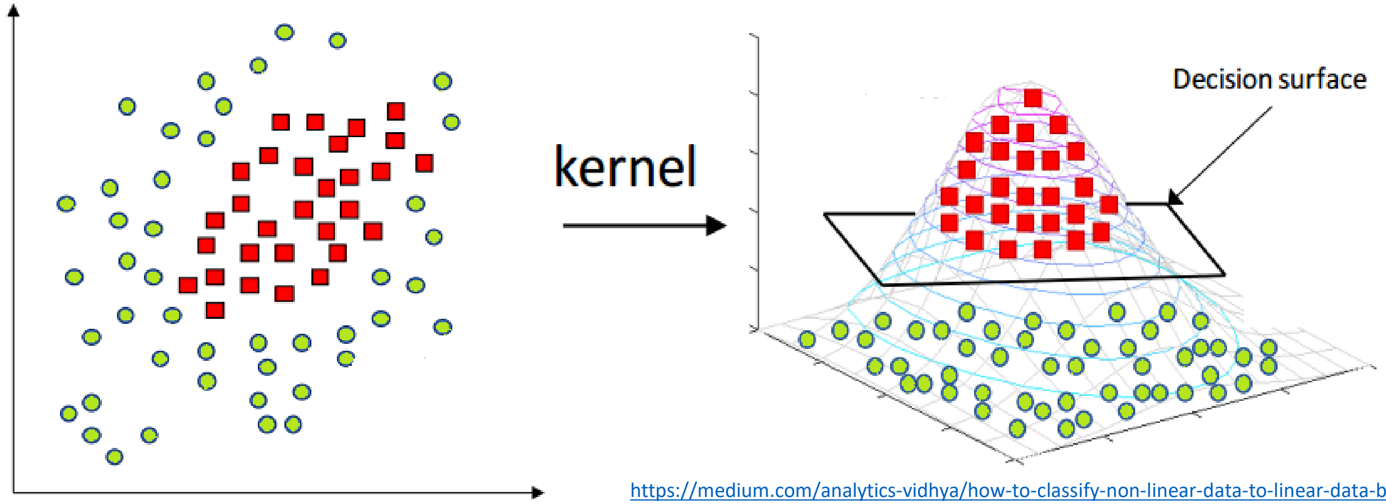
Hard-margin SVM

Soft Margin Support Vector Machines

- Soft margin SVMs are better able to handle noisy data
- Small C: more tolerance on miss-classified samples for larger margin
- Large C: focus on avoiding mistakes at the expense of smaller margin
- C to infinity means going back to the hard margin SVM
- Still a quadratic programming optimization problem

Nonlinear Support Vector Machines

- To generate nonlinear decision boundaries, we can map the features into a new feature space where classes are linearly separable and then apply the SVM there



https://medium.com/analytics-vidhya/how-to-classify-non-linear-data-to-linear-data-bb2df1a6b781

- Feature mapping into a higher dimensional space can be done using a kernel function which reduces the complexity of the optimization problem

Support Vector Machines

- **Pros**
  - ✓ Very effective in high dimensional feature spaces
  - ✓ Effective when the number of features is larger than the training data size
  - ✓ Among the best algorithms when the classes are (well) separable
  - ✓ Work very well when the data is sparse
  - ✓ Can be extended to nonlinear classification via kernel trick

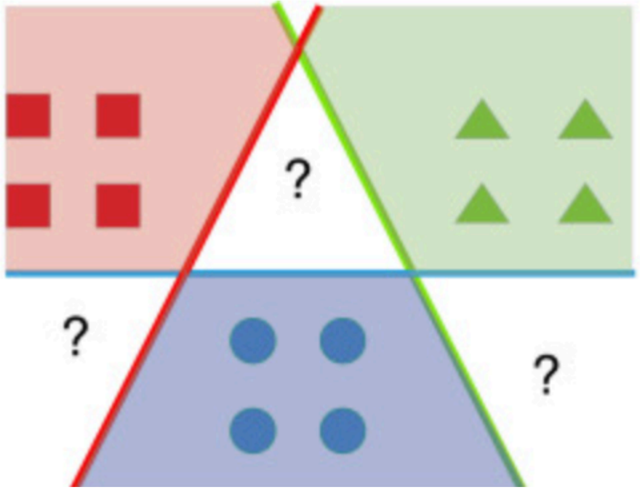
- **Cons**
  - × For larger datasets it takes more time to process
  - × Does not perform well for overlapping classes
  - × Hyperparameter tuning needed for sufficient generalization

Multiclass Classification

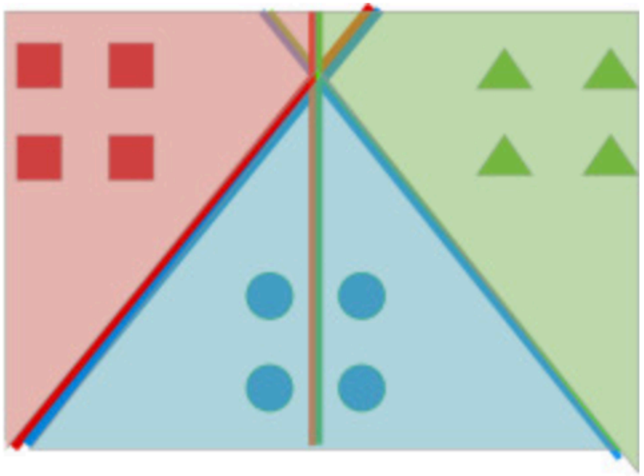
- If there are more than two classes, we must build a multiclass classifier
- Some methods may be directly used for multiclass classification:
  - K-nearest neighbours
  - Decision trees
  - Bayesian techniques
- For those that cannot be directly applied to multiclass problems, we can transform them to binary classification by building multiple binary classifiers
- Two possible techniques for multiclass classification with binary classifiers:
  - **One versus rest**: builds one classifier for one class versus the rest and assigns a test sample to the class that has the highest confidence score
  - **One versus one**: builds one classifier for every pair of classes and assigns a test sample to the class that has the highest number of predictions

Multiclass Classification

- Two possible techniques for multiclass classification with binary classifiers:
  - **One versus rest**: builds one classifier for one class versus the rest and assigns a test sample to the class that has the highest confidence score
  - **One versus one**: builds one classifier for every pair of classes and assigns a test sample to the class that has the highest number of predictions



one vs rest



one vs one

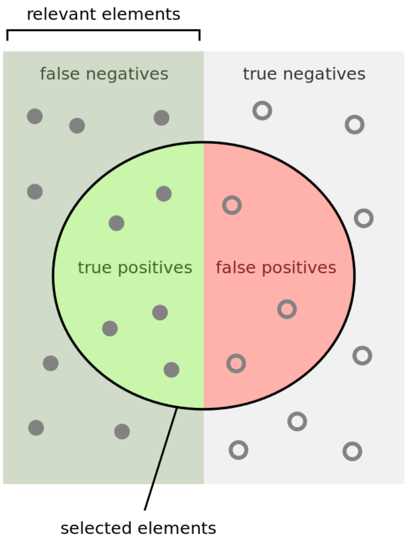
Evaluation of Classification Error

- **Error rate**
  - Measures how well/poor the system solves the problem it was designed for
- **Reject class**
  - Generic class for objects that cannot be placed in any of the known classes
- **Classification error**
  - The classifier makes a classification error whenever it classifies an input object as class C_i when the true class is C_j, $i \neq j$, and $C_i \neq C_r$ (the reject class)
- **Performance**
  - Performance determined by both errors and rejections made
  - Classifying all inputs into reject class means system makes no errors but is useless!

Evaluation of Classification Error

- **Empirical error rate**
  - Number of errors on independent test data divided by number of classifications attempted
- **Empirical reject rate**
  - Number of rejects on independent test data divided by number of classifications attempted
- **Independent test data**
  - Sample objects with true class (labels) known, including objects from the reject class, and that were not used in designing the feature extraction and classification algorithms
- **Samples used for training and testing should be representative**
  - Available data is split for example in 80% training and 20% test data

False Alarms and False Dismissals

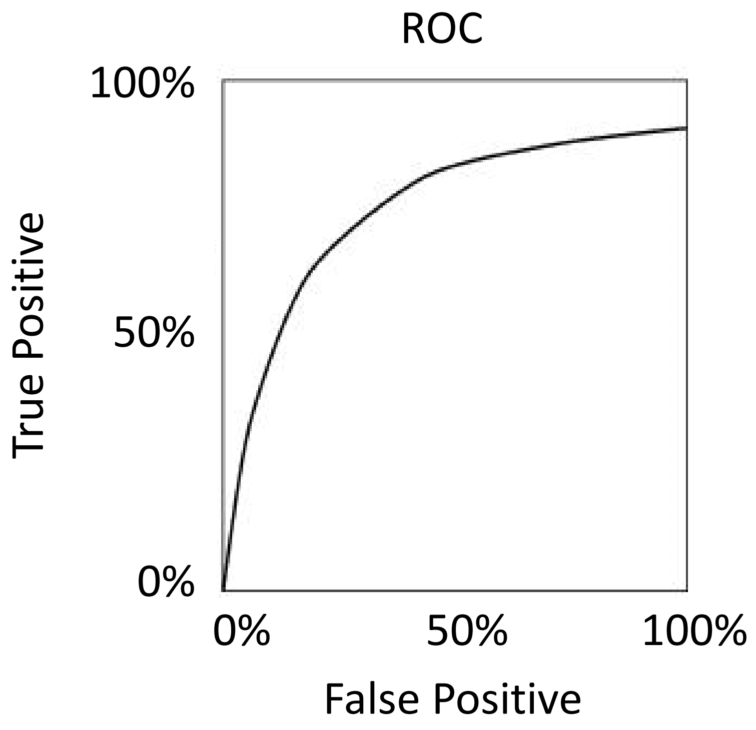


- For two-class classification problems, the two possible types of errors have a special meaning and are not symmetric
- For example, in medical diagnosis:
  - If the person does NOT have the disease, but the system incorrectly says they do, then the error is a **false alarm** or **false positive** (also called **type I error**)
  - If the person DOES have the disease, but the system incorrectly says they do NOT, then the error is a **false dismissal** or **false negative** (also called **type II error**)
- Consequences and costs of the two errors can be very different
  - There are bad consequences to both, but false negatives are generally more catastrophic
  - So, the aim is to minimize false negatives, possibly at the cost of increasing false positives
  - The optimal/acceptable balance of the two errors depends on the application

Receiver Operating Curve (ROC)

- Binary classification;
- For each sample, probability of classifying as positive class, p1;
- Conducting classification with threshold on p1;
- Given different threshold, we can get different results.
  - different false positive and true negative rate on the whole dataset
- By applying different threshold, we can get ROC.

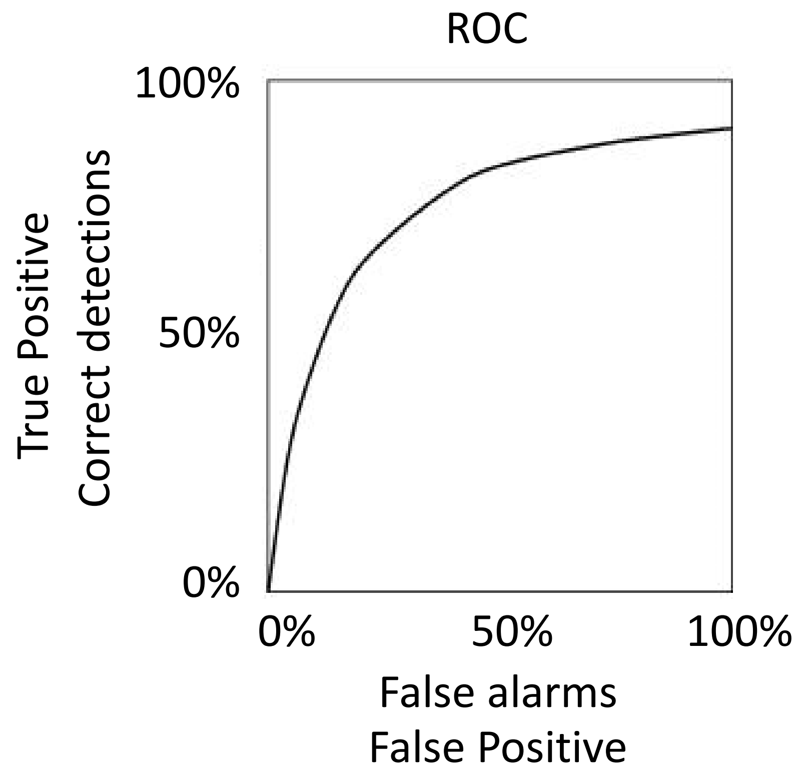
- The Receiver Operator Curve (ROC) relates the false positive to the true positive.
- Plots the correct true positive versus the false positive (false alarm) rate



Truth	Classification	Error?
Cancer	Cancer	Correct detection (no error)
No cancer	Cancer	False alarm (error)
Cancer	No cancer	False dismissal (error)
No cancer	No cancer	Correct dismissal (no error)

Receiver Operating Curve (ROC)

- Generally, false alarms go up with attempts to correctly detect higher percentages of known objects
- Area Under the ROC (AUC or AUROC) summarizes overall performance
- How to evaluate the quality of a classifier based on ROC?

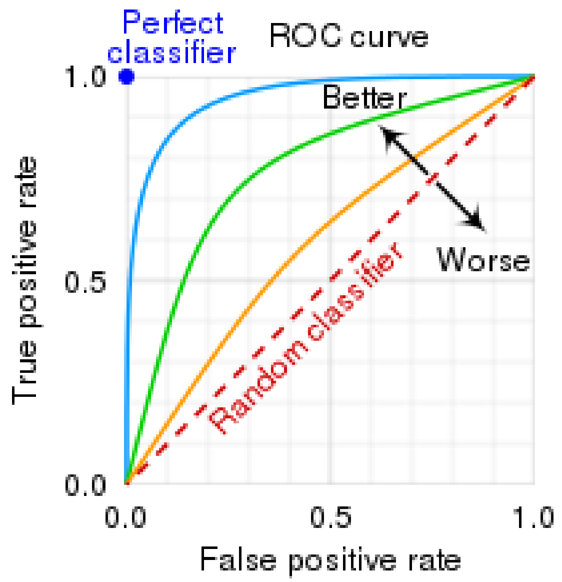


Truth	Classification	Error?
Cancer	Cancer	Correct detection (no error)
No cancer	Cancer	False alarm (error)
Cancer	No cancer	False dismissal (error)
No cancer	No cancer	Correct dismissal (no error)

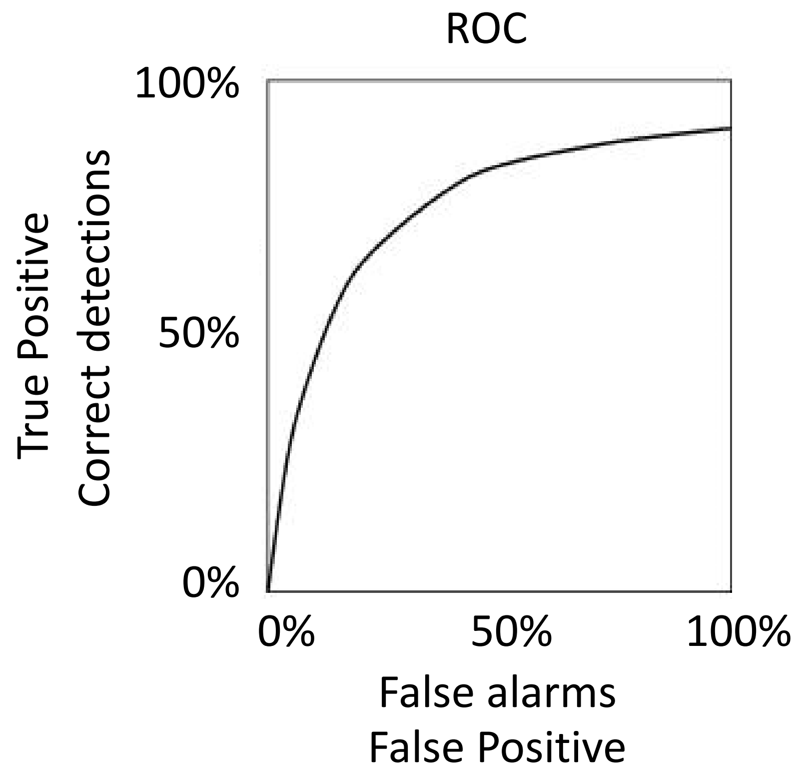
Receiver Operating Curve (ROC)

- Generally, false alarms go up with attempts to correctly detect higher percentages of known objects

- Area Under the ROC (AUC or AUROC) summarizes overall performance



https://en.wikipedia.org/wiki/Receiver_operating_characteristic



Truth	Classification	Error?
Cancer	Cancer	Correct detection (no error)
No cancer	Cancer	False alarm (error)
Cancer	No cancer	False dismissal (error)
No cancer	No cancer	Correct dismissal (no error)

Confusion Matrix

- Matrix whose entry (*i*, *j*) records the number of times an object of class *i* was classified as class *j*
- Often used to report the results of classification experiments
- Diagonal entries indicate successes
- High off-diagonal numbers indicate confusion between classes

Handwritten digits recognition

class j output by the pattern recognition system

true object class i	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'R'
'0'	97	0	0	0	0	0	1	0	0	1	1
'1'	0	98	0	0	1	0	0	1	0	0	0
'2'	0	0	96	1	0	1	0	1	0	0	1
'3'	0	0	2	95	0	1	0	0	1	0	1
'4'	0	0	0	0	98	0	0	0	0	2	0
'5'	0	0	0	1	0	97	0	0	0	0	2
'6'	1	0	0	0	0	1	98	0	0	0	0
'7'	0	0	1	0	0	0	0	98	0	0	1
'8'	0	0	0	1	0	0	1	0	96	1	1
'9'	1	0	0	0	3	0	0	0	1	95	0

Binary Confusion Matrix

- Confusion matrix for binary classification

		Prediction	
		P	N
Actual	P	**# True Positives (TP)**	# False Negatives (FN)
	N	# False Positives (FP)	**# True Negatives (TN)**

- Accuracy

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \qquad \left(\frac{\text{Correct}}{\text{Total}}\right)$$

Precision versus Recall

- **Precision / correctness**

  Fraction of relevant elements among the selected elements

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{(P)}$$

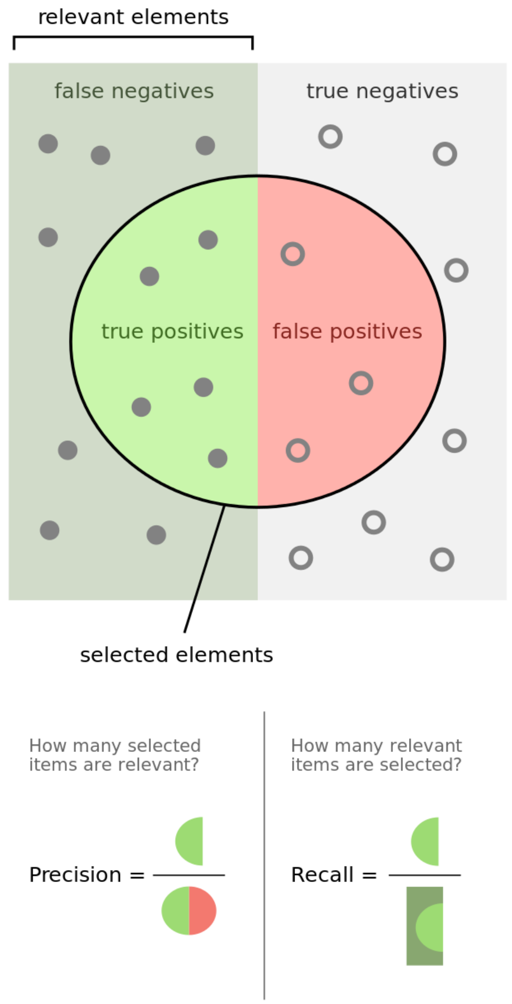
- **Recall / sensitivity / completeness**

  Fraction of selected elements among the relevant elements

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{(R)}$$

- **F1 score**

  Harmonic mean of precision and recall: $\text{F1} = \dfrac{2\text{PR}}{\text{P} + \text{R}}$



https://en.wikipedia.org/wiki/Precision_and_recall

More Terminology and Metrics

condition positive (P)
the number of real positive cases in the data

condition negative (N)
the number of real negative cases in the data

true positive (TP)
eqv. with hit

true negative (TN)
eqv. with correct rejection

false positive (FP)
eqv. with false alarm, type I error or underestimation

false negative (FN)
eqv. with miss, type II error or overestimation

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} = 1 - \mathrm{FDR}$$

negative predictive value (NPV)

$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FN}} = 1 - \mathrm{FOR}$$

miss rate or false negative rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TP}} = 1 - \mathrm{TPR}$$

fall-out or false positive rate (FPR)

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{N}} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} = 1 - \mathrm{TNR}$$

false discovery rate (FDR)

$$\mathrm{FDR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TP}} = 1 - \mathrm{PPV}$$

false omission rate (FOR)

$$\mathrm{FOR} = \frac{\mathrm{FN}}{\mathrm{FN}+\mathrm{TN}} = 1 - \mathrm{NPV}$$

prevalence threshold (PT)

$$\mathrm{PT} = \frac{\sqrt{\mathrm{TPR}(-\mathrm{TNR}+1)}+\mathrm{TNR}-1}{(\mathrm{TPR}+\mathrm{TNR}-1)} = \frac{\sqrt{\mathrm{FPR}}}{\sqrt{\mathrm{TPR}}+\sqrt{\mathrm{FPR}}}$$

threat score (TS) or critical success index (CSI)

$$\mathrm{TS} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}+\mathrm{FP}}$$

accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}}$$

balanced accuracy (BA)

$$\mathrm{BA} = \frac{TPR+TNR}{2}$$

F1 score
is the harmonic mean of precision and sensitivity:

$$\mathrm{F}_1 = 2 \times \frac{\mathrm{PPV}\times\mathrm{TPR}}{\mathrm{PPV}+\mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$$

Matthews correlation coefficient (MCC)

$$\mathrm{MCC} = \frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}$$

Fowlkes–Mallows index (FM)

$$\mathrm{FM} = \sqrt{\frac{TP}{TP+FP}\times\frac{TP}{TP+FN}} = \sqrt{PPV\times TPR}$$

informedness or bookmaker informedness (BM)

$$\mathrm{BM} = \mathrm{TPR}+\mathrm{TNR}-1$$

markedness (MK) or deltaP (Δp)

$$\mathrm{MK} = \mathrm{PPV}+\mathrm{NPV}-1$$

Table of metrics computed from the confusion matrix and often used in classification

https://en.wikipedia.org/wiki/Confusion_matrix

Regression

- Suppose we have a training set of N observations:

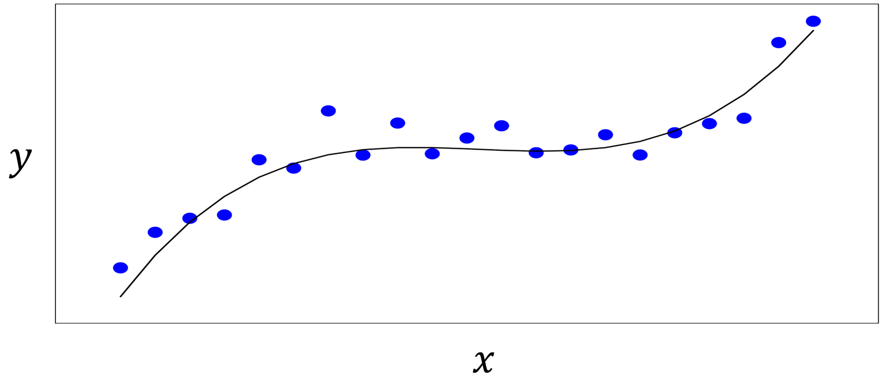
$$\{(x_i, y_i)\}, \quad x_i \in \mathbb{R}^d, \quad y_i \in \mathbb{R}, \quad i = 1, \dots, N$$

- Training process is to learn $f(x)$ from the training data such that:

$$y_i = f(x_i)$$

- But here the output variable has a continuous value

Example for $d = 1$



Linear Regression

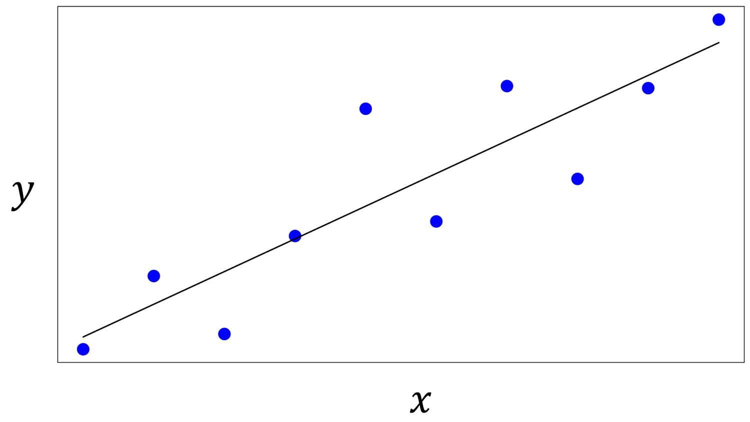
- Linear regression assumes there is a linear relationship between the output and the features:

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

$$x = [1, x_1, x_2, \ldots, x_d] \qquad \text{(features)}$$

$$W = [w_0, w_1, \ldots, w_d]^T \qquad \text{(weights)}$$

$$f(x) = xW$$



- How to find the best line?

  The most basic estimation approach is **least squares fitting**

Least Squares Regression

- The idea is to minimize the residual sum of squares (sum of the squared error)

  $\mathrm{RSS}(W) = \sum_{i=1}^{N}[y_i - f(x_i)]^2 = (Y - XW)^T(Y - XW)$

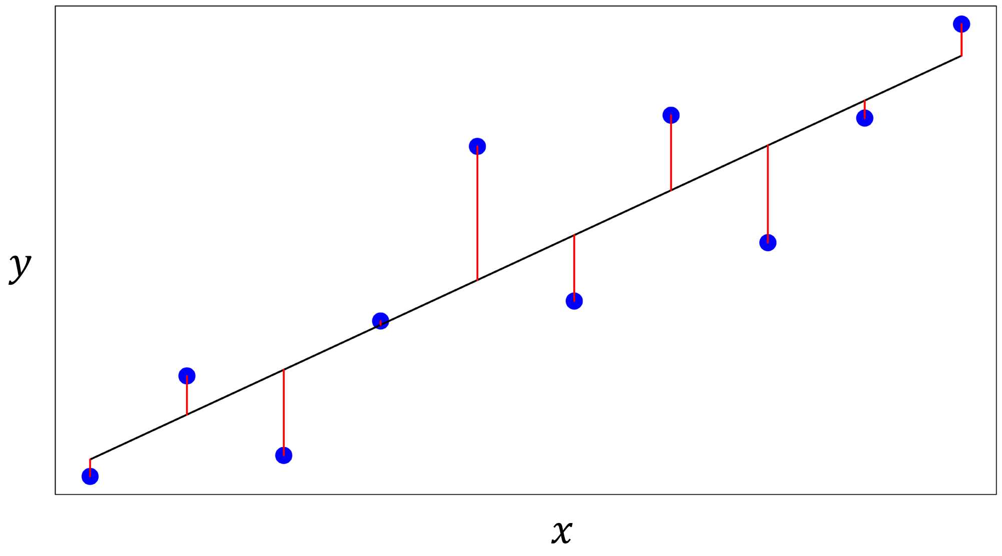
  $Y = [y_1, y_2, \dots, y_N]^T$ (all sample values)

  $X = [x_1, x_2, \dots, x_N]^T$ (all sample features)

- How to find the best fit?

  $\widehat{W} = \arg\min_W \mathrm{RSS}(W)$

- RSS is a quadratic function that can be differentiated with respect to W



Least Squares Regression

- Differentiation of RSS with respect to W yields:

$$\frac{\partial \mathrm{RSS}}{\partial W} = -2X^T(Y - XW)$$

$$\frac{\partial^2 \mathrm{RSS}}{\partial W \partial W^T} = 2X^T X$$

- If we assume that X has full rank, then $X^T X$ is positive and that means we have a convex function which has a minimum, so:

$$\frac{\partial \mathrm{RSS}}{\partial W} = 0 \quad \Rightarrow \quad X^T(Y - XW) = 0$$

$$\widehat{W} = (X^T X)^{-1} X^T Y$$

Linear Regression: Example

- Assume we have the length and width of some fish and we want to estimate their weights from this information (features)
- Start with one feature (say x_1) which is easier for visualization

$$y = w_0 + w_1 x_1$$

$$X = \begin{bmatrix} 1 & 100 \\ 1 & 102 \\ & \vdots \\ 1 & 97 \end{bmatrix}, \quad W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}, \quad Y = \begin{bmatrix} 5 \\ 4.5 \\ \vdots \\ 4.3 \end{bmatrix}$$

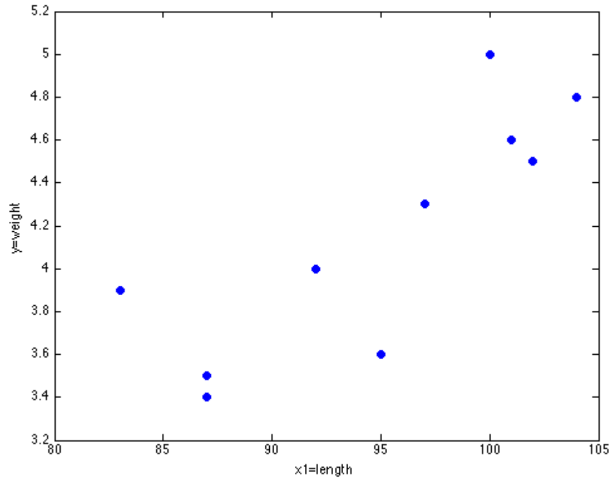
Length (x_1)	Width (x_2)	Weight (y)
100	40	5
102	35	4.5
92	33	4
83	29	3.9
87	36	3.5
95	30	3.6
87	37	3.4
104	38	4.8
101	34	4.6
97	39	4.3

Linear Regression: Example

- For one feature we obtain:

$$W = (X^T X)^{-1} X^T Y = \begin{bmatrix} -1.8 \\ 0.0635 \end{bmatrix}$$

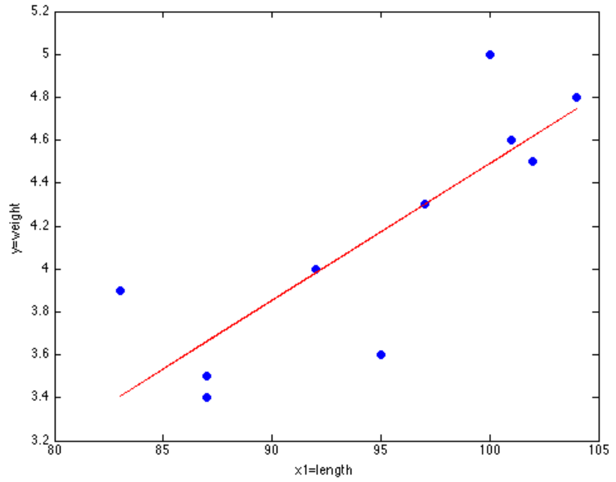
$$\mathrm{RSS}(W) = \sum_{i=1}^{N} [y_i - f(x_i)]^2 = (Y - XW)^T (Y - XW) = 0.9438$$



- For two features we repeat the same procedure with updated X:

$$X = \begin{bmatrix} 1 & 100 & 40 \\ 1 & 102 & 35 \\ & \vdots & \\ 1 & 97 & 39 \end{bmatrix}$$

$$W = \begin{bmatrix} -2.125 \\ 0.0591 \\ 0.0194 \end{bmatrix} \qquad \mathrm{RSS}(W) = 0.9077$$



Regression Evaluation Metrics

- **Root Mean Square Error (RMSE)**

  Represents the standard deviation of the predicted values from the observed values

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

- **Mean Absolute Error (MAE)**

  Represents the average of the absolute differences between the predicted and observed values

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i\,|$$

RMSE penalizes big differences between predicted values and observed values more heavily

Smaller values of RMSE and MAE are more desirable

Regression Evaluation Metrics

- **R-Squared (R^2)**

  Indicates how well the selected feature(s) explain the output variable

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

  R-squared tends to always increase by adding extra features

- **Adjusted R-Squared (Adjusted R^2)**

  Indicates how well the selected feature(s) explain the output, adjusted for the number of features:

$$R_{\text{adj}}^2 = 1 - \left[\frac{(1 - R^2)(N - 1)}{N - d - 1}\right]$$

  where N is the number of samples and d is the number of features

  Larger values of R-Squared and Adjusted R-Squared are more desirable

Normalization on features -- preprocessing

- Goal: to change the scale of numeric values to a common scale
- Commonly applied techniques:
  - **Z-score:** re-scales the data (features) such that it will have a standard normal distribution ($\mu = 0$, $\sigma = 1$), which works well for normally distributed data:

$$\frac{x - \mu}{\sigma}$$

  - **Min-max normalization:** re-scales the range of the data to [0,1] such that the minimum value is mapped to 0 and the maximum value to 1:

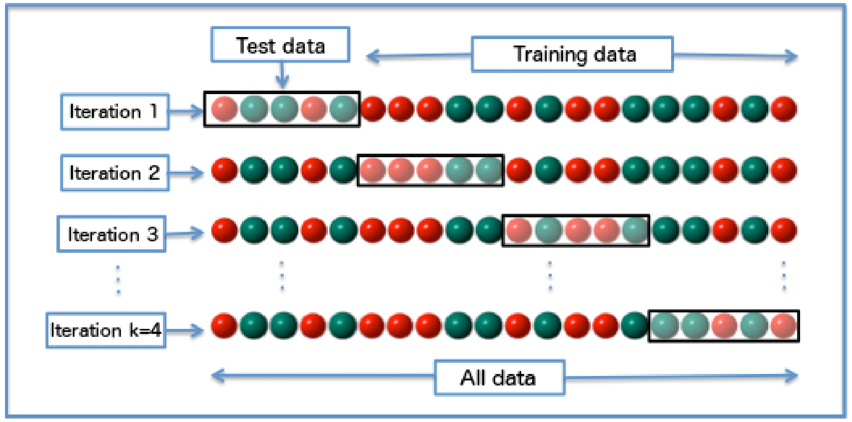
$$\frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

Cross Validation

- Ideally a trained model should work well also on new (unseen) data
- This means the model should neither underfit nor overfit the training data
- Can be used for hyperparameter tuning
- Cross validation (CV) is a technique to assess model performance across all data
  - **Train-test split:** The available data is randomly split into a training set and a test set (usually 80:20 ratio) for, respectively, training and testing the model
  - **K-fold cross validation:** The data is split into K subsets (folds) and at each iteration we keep one fold out for testing and use the rest for training

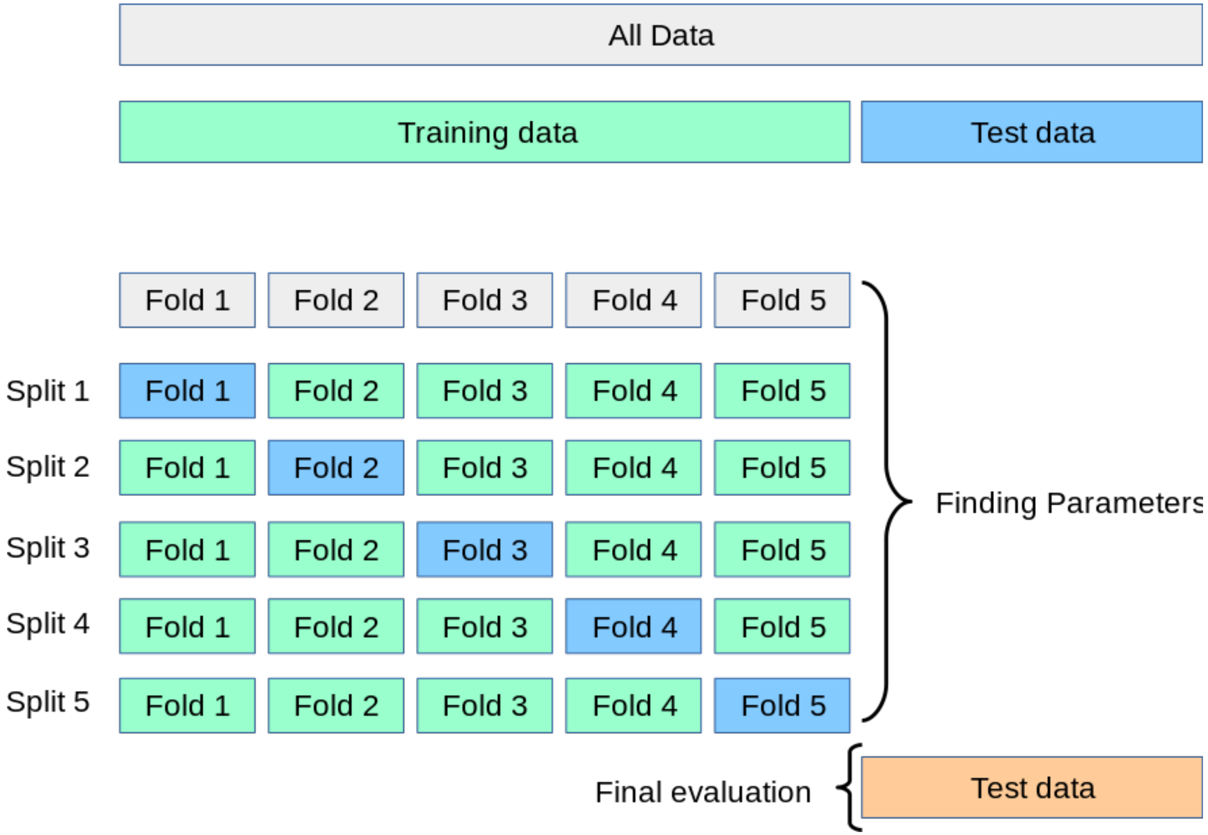
    This is repeated K times until all folds have been used once as the test set

    The performance of the model will be the average of the performance on the K test sets



Cross Validation

- Cross validation can be used for hyperparameter tuning (or model selection)
  - Leave a test set
  - Do cross validation on the rest of data with training set and validation set
  - Test set cannot be used for selecting the hyperparameter



Source: https://erdogant.github.io/hgboost/pages/html/Cross%20validation%20and%20hyperparameter%20tuning.html#:~:text=Cross%20validation%20and%20hyperparameter%20tuning%20are%20two%20tasks%20that%20we,crossvalidation%20to%20evalute%20our%20results.

References and Acknowledgements

- Shapiro & Stockman, Chapter 4
- Duda, Hart, Stork, Chapters 1, 2.1
- Hastie, Tibshirani, Friedman, *The Elements of Statistical Learning*, Chapters 2 and 12
- Theodoridis & Koutroumbas, *Pattern Recognition*, 2009
- Ian H. Witten & Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2005

- Some diagrams extracted from the above resources