



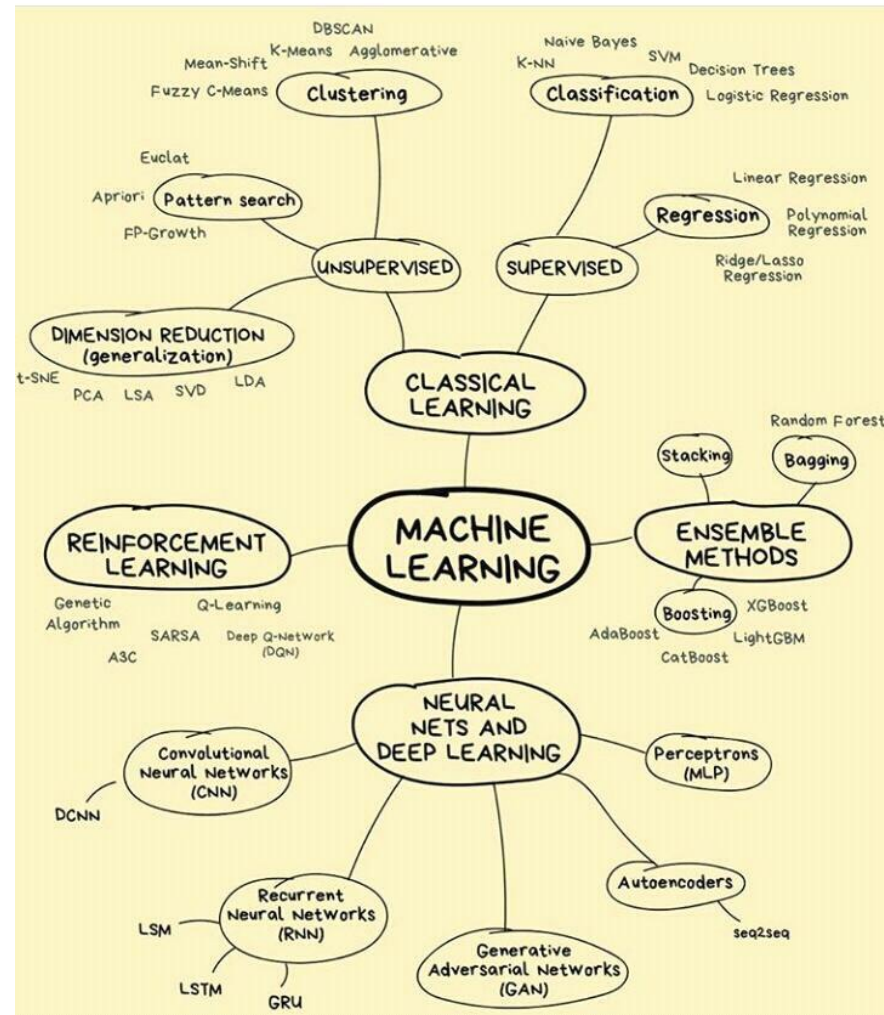
# Recap

Never Stand Still

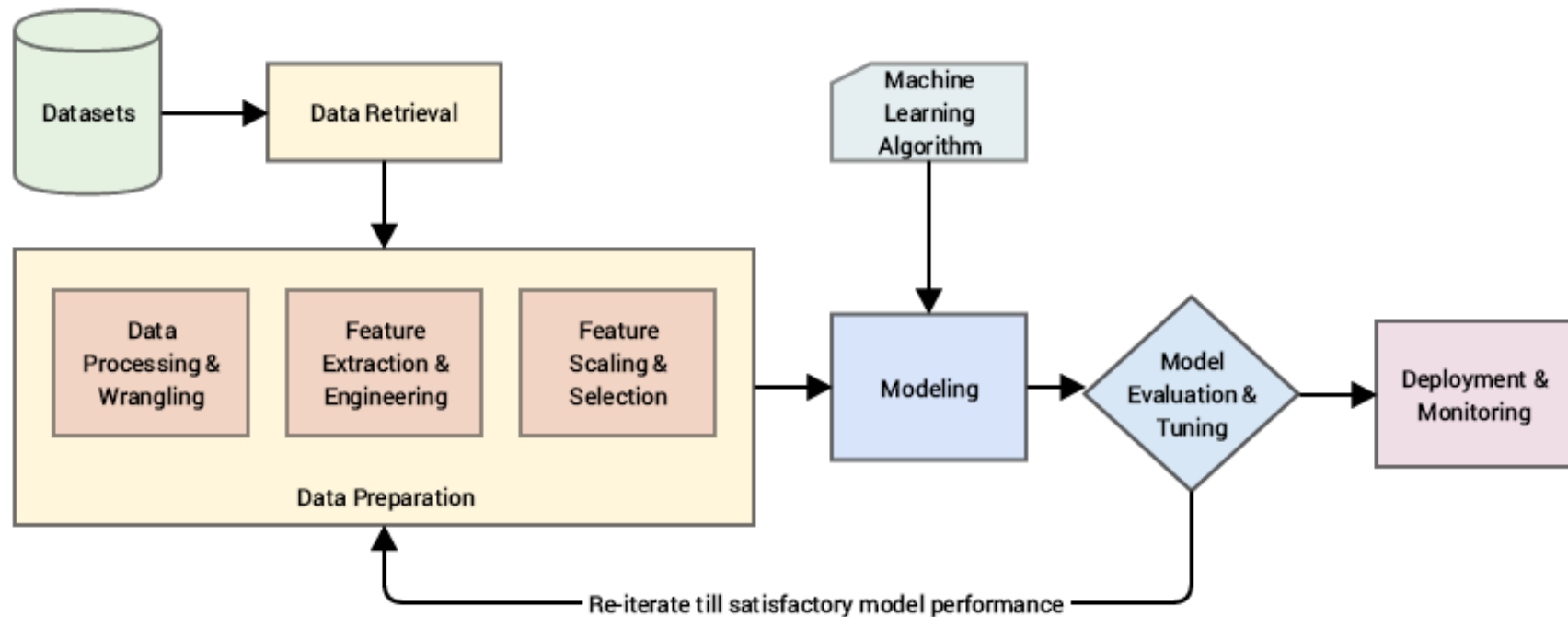
COMP9417 Machine Learning & Data Mining  
Term 1, 2020

Adapted from slides by Dr Michael Bain

# Machine Learning

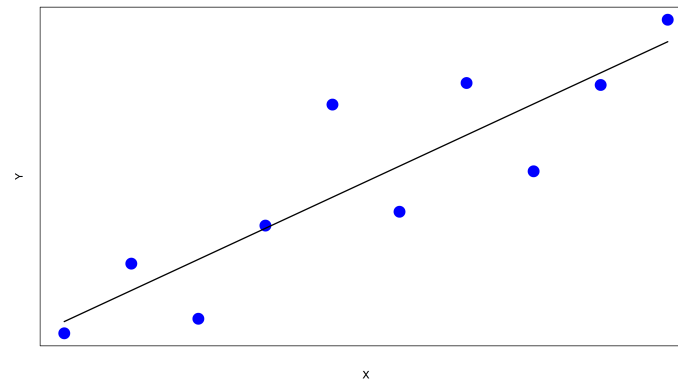
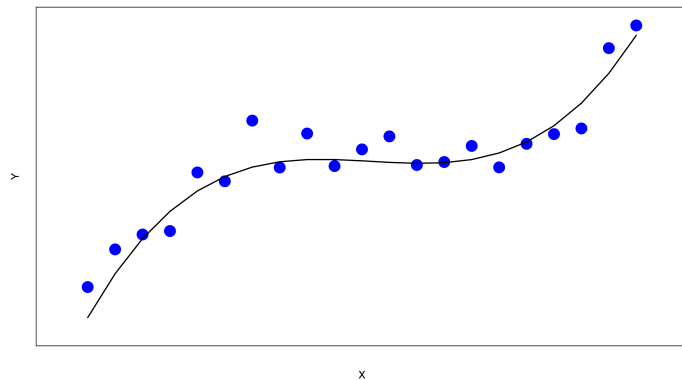


# Machine Learning Pipeline



# Regression

Regression models are used to predict a continuous value.



# Regression

## 1. Simple Linear Regression

- The most common cost function: Mean Squared Error (MSE)
- Cost function can be minimized using Gradient Descent (it has also closed form solution)
- Regression coefficients/weights ( $\theta_i$ ) describe the relationship between a predictor variable ( $x_i$ ) and the output variable ( $y$ )
- Regularization is applied to avoid overfitting
  - It applies additional constraints to the weights usually to keep weights small (shrinkage) and can be used as feature selection too
  - Most common regularization approaches:
    - Ridge (penalize  $\sum_i \theta_i^2$ )
    - Lasso (penalize  $\sum_i |\theta_i|$ )
    - Elastic Net (a combination of Ridge and Lasso)

# Regression

## 2. Polynomial Regression

- Create polynomial terms from your features
- Will be solved similar to simple Linear Regression
- Model is still linear in parameters

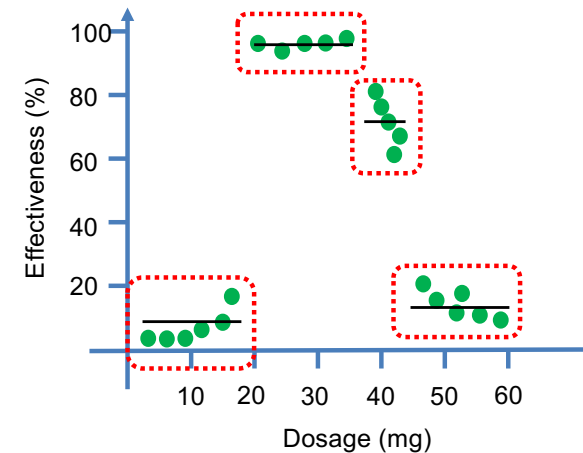
## 3. Local regression

- Use the  $k$  nearest neighbors to fit a regression line
- Produces a piecewise approximation

# Regression

## 4. Decision Tree Regression (regression tree)

- Partitioning data into homogeneous subsets
- Variance or standard deviation reduction is used to decide for splitting
- The predicted value for each leaf is the average value of the samples in that leaf



## 5. Model Tree

- Similar to regression trees but with linear regression at each leaf
- Splitting criterion is standard deviation reduction

# Model Evaluation

The most popular metrics are:

- Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{1}{m} \sum_{j=1}^m (y_j - \hat{y}_j)^2}$$

- Mean Absolute Error (MAE)

$$MAE = \frac{1}{m} \sum_{j=1}^m |(y_j - \hat{y}_j)|$$

- R-squared (  $[-\infty, 1]$  )

$$R^2 = 1 - \frac{\sum_{j=1}^m (y_j - \hat{y}_j)^2}{\sum_{j=1}^m (y_j - \bar{y}_j)^2}$$

- Adjuster R-squared

$$R_{adjusted}^2 = 1 - \left[ \frac{(1 - R^2)(m - 1)}{m - n - 1} \right]$$

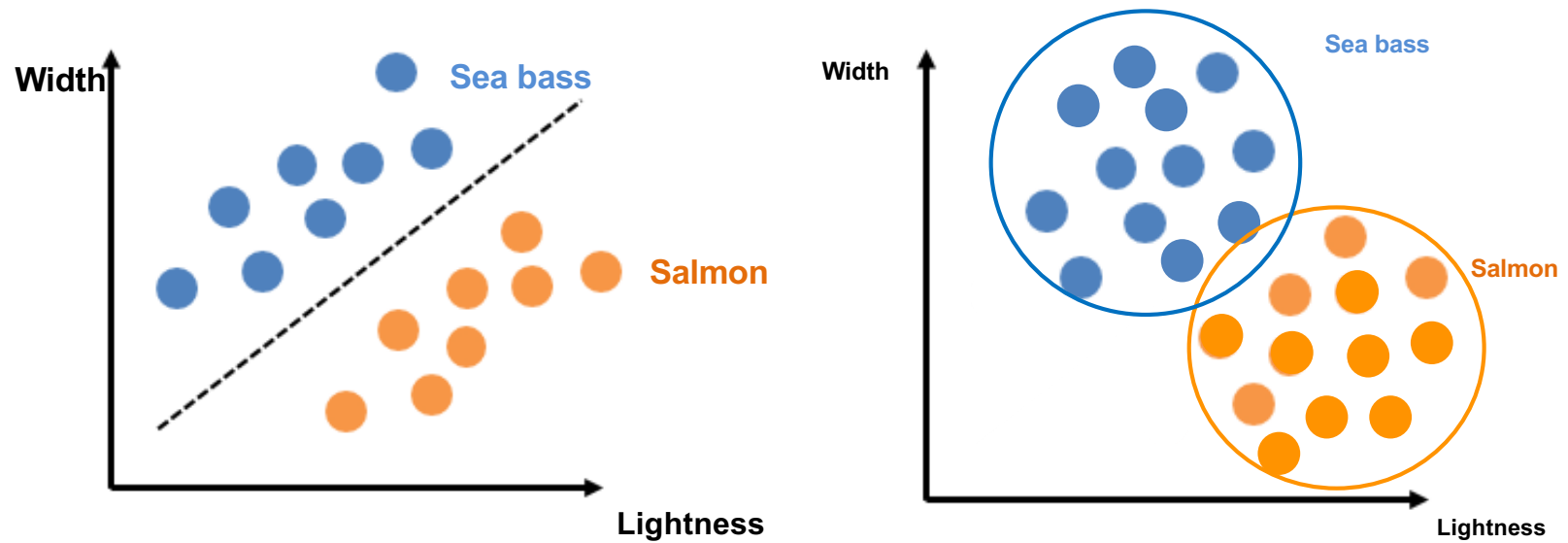
Where  $m$  is the total number of samples and  $n$  is the number of predictors/features.

- R-squared represents the portion of variance in the output that has been explained by the model



# Classification

Classification is prediction of categorical output from input attributes/features.



# Classification

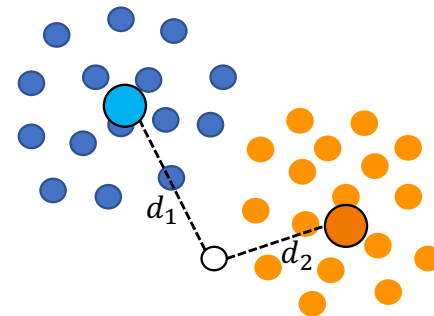
Two main types of classification:

- **Generative algorithm:** builds some models for each of the classes
  - Learns  $p(x|y)$
  - and then estimate  $p(y|x)$  using Bayes theorem
- **Discriminative algorithm:** Do not build models for different classes, but rather focuses on finding a decision boundary
  - Learns  $p(y|x)$  directly

# Classification

## 1. Nearest centroid classifier

- Distance based classifier
- $\mu_k = \frac{1}{|C_k|} \sum_{j \in C_k} x_j$
- For complex classes (eg. Multimodal, non-spherical) may give very poor results
- Can not handle outliers and noisy data well
- Not very accurate



# Classification

## 2. k nearest neighbor classifier (kNN)

- Distance base classifier
- Find k nearest neighbor using an appropriate distance metric (e.g. Minkowski distance)
- Predict the output based on the majority vote
- Works better with lots of training data and small number of attributes
- Can be very accurate but slow at testing time
- Curse of dimensionality
- Assumes all attributes are equally important
  - Remedy: attribute selection or attribute weights
- Needs homogenous feature type and scale

# Classification

3. Bayesian decision theory (based on Bayesian theorem,  $P(h|D) = \frac{P(D|h)P(h)}{P(D)}$ )
- The prediction will be the most probable hypothesis if expected loss is equal for all classes :
    - Maximum a posteriori ( $h_{MAP} = \arg \max_{h \in H} P(h|D) = \arg \max_{h \in H} P(D|h)P(h)$ )
    - If  $P(h_i) = P(h_j)$  , we use maximum likelihood ( $h_{ML} = \arg \max_{h_i \in H} P(D|h_i)$ )
  - If the expected loss is not the same, then we have to predict the class which minimizes the expected loss
    - Expected loss:  $R(\alpha_i|x) = \sum_{h \in H} \lambda(\alpha_i|h) P(h|x)$

# Classification

4. Bayes optimal classification:  $(\arg \max_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D))$
- Here we are dealing with combining the decision from multiple hypothesis
  - No other classification method using the same hypothesis space and same prior knowledge can outperform this method on average
  - Bayes optimal classifier is very inefficient

# Classification

## 5. Naïve Bayes classifier

- Using Bayesian theory
- The main difference is the strong assumption that attributes are conditionally independent:  $P(x_1, x_2, \dots, x_n | v_j) = \prod_i P(x_i | v_j)$
- Prediction is based on maximum a posteriori:  
$$v_{NB} = \arg \max_{v_j \in V} P(x_1, x_2, \dots, x_n | v_j) P(v_j) = \arg \max_{v_j \in V} \hat{P}(v_j) \prod_i \hat{P}(x_i | v_j)$$
- Useful when:
  - moderate or large training set available
  - Attributes are conditionally independent (however this is usually violated and still NB can do a descent job!)
- Having too many redundant attributes will decrease the performance

# Classification

## 6. Decision tree:

- Works in divide and conquer fashion
  - Split into subsets
  - Check the subset purity
    - Use entropy to measure impurity at each node ( $E(s) = \sum_{i=1}^c -p_i \log_2 p_i$ )
    - Use information gain to decide which attribute works better for that node

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

- » However *IG* is more biased towards attributes with large number of possibilities, so we can use *Gain Ratio* instead
- Attribute with highest information gain will be selected for the node



# Classification

## 6. Decision tree:

- Decision trees can work with any type of data (discrete and numeric)
- Can handle missing values
- One of the main advantages is interpretability
- Can almost always classify training example perfectly if we let it grow enough which means it can overfit
- To avoid overfitting
  - Pre-pruning: stop growing when split is not statistically significant like chi-squared test(suffer from early stopping). Or limiting *min\_sample\_leaf*, *min\_impurity\_decrease*, *max\_leaf\_node* or *max\_depth*, ect.
  - Post-pruning: grow full tree, then remove sub-trees that cause overfitting based on cross validation
- Greedy algorithm (may not find the optimal tree)

# Classification

## 7. Linear Perceptron ( $\hat{y} = f(x) = \text{sgn}(w \cdot x)$ )

- Weights get updated iteratively until no mistake is made or max number of iteration is met
- Simple and fast at training
- Doesn't perform well if classes are not linearly separate

## 8. Non-linear perceptron

- Map attributes into new space consisting of polynomial terms and interaction terms
- Use kernel trick to make the computation much less

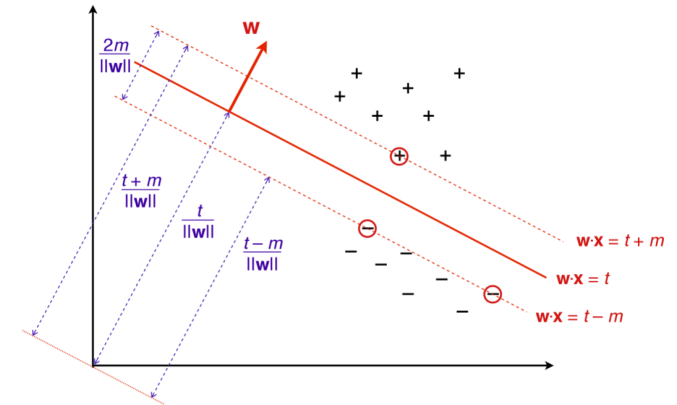
$$\hat{y} = \text{sign} \left( \sum_{i=1}^m \alpha_i y_i (\varphi(x_i) \cdot \varphi(x)) \right)$$

- A valid kernel function is equivalent to a dot product in some space ( $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$ )

# Classification

## 9. Linear Support Vector Machine (maximum margin)

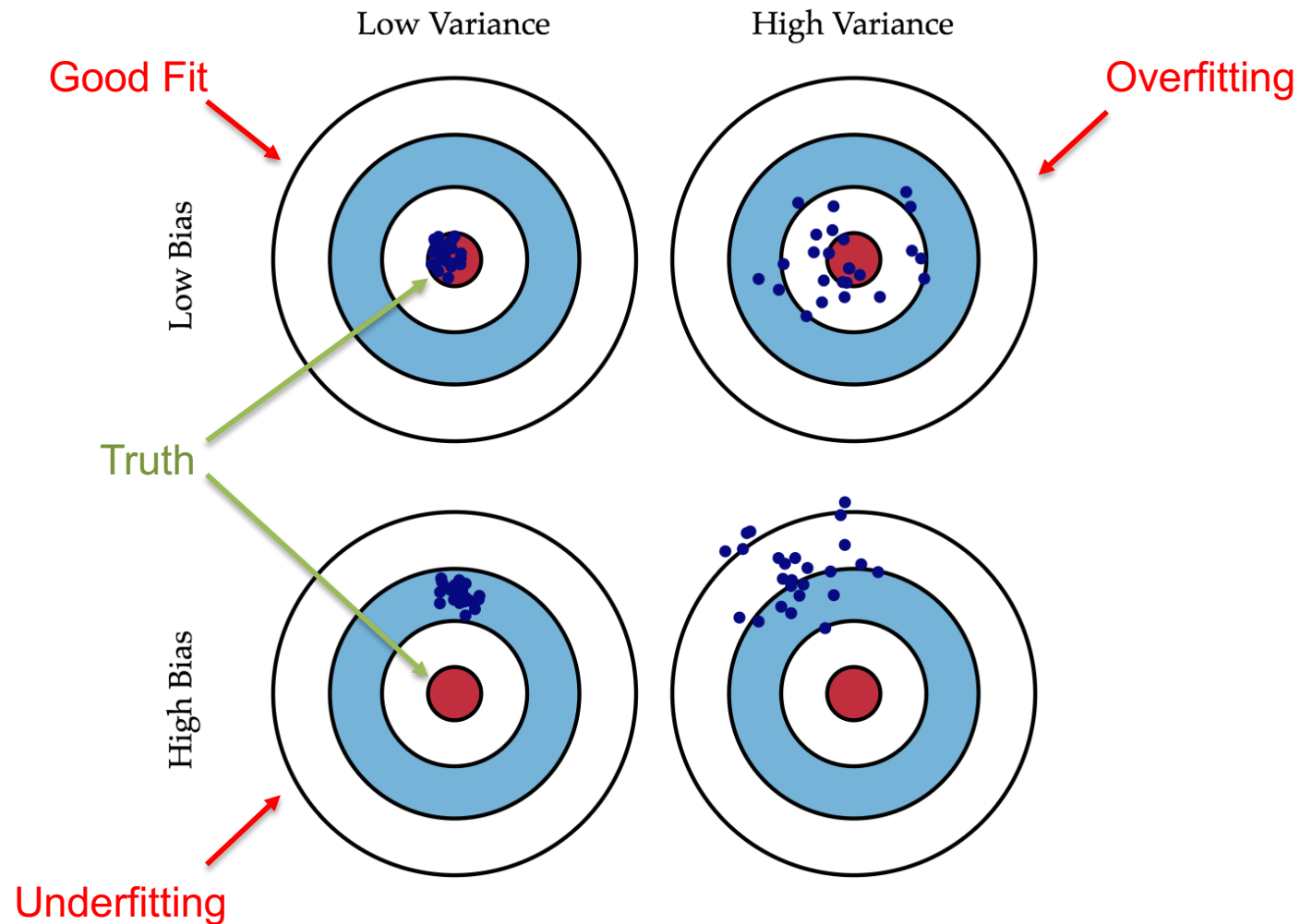
- $\hat{y} = \text{sign}(w \cdot x - t)$
- $w = \sum_{x_i \in \{\text{support vectors}\}} \alpha_i y_i x_i$
- $\alpha_i$  is non-zero for support vectors
- Is effective in high dimensional data
- Is effective when number of dimensions is bigger than the number of samples



## 10. Nonlinear SVM

- Similar to perceptron, Kernel trick can be applied using dual form
- $\hat{y} = \text{sign}(\sum_{\alpha_i > 0} \alpha_i y_i K(x_i, x) - t)$

# Bias-Variance Tradeoff



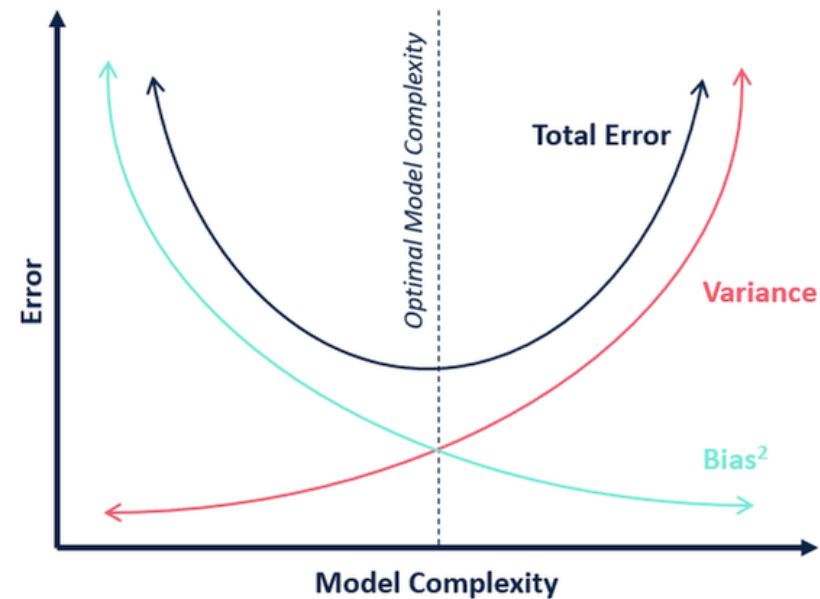
Source: Scott-Fortmann, Understanding Bias-variance tradeoff

# Bias-Variance

- Bias-variance:
  - Bias: The inability of the learning algorithm to capture the true relationship between the output and the features/attributes is called **bias**.
    - due to model choice which (e.g. is not complex enough)
  - Variance: The learning algorithm difference in fits between datasets is called **variance**.
    - due to small sample size
    - high complexity of the model

# Bias-Variance

- The aim is to have a good bias variance tradeoff
  - methods to find a good bias-variance trade-off:
    - Regularization
    - Ensemble learning in general
    - Bagging
    - Boosting



# Ensemble Learning

Ensemble methods: meta-algorithms that combine different models into one model

1. Simple ensembles: combining several learning algorithm
  - Majority vote or unweighted average will be used for prediction
  - Using weighted average or weighted votes to predict the output
  - Treat the output of each algorithm as a feature and train another learning algorithm on them
2. Mixture of experts
  - Each learning algorithm defines  $\alpha_i(x)$  which indicated the expertise of that algorithm for that particular location of  $x$  in the input space
  - It may use a weighted average or just pick the model with the largest expertise

# Ensemble Learning

## 3. “Bagging” method: (“**B**ootstrap **A**ggregation”)

- Training many classifiers, but each on a Bootstrapped dataset
  - Bootstrap: Create a random subset of data by sampling with replacement
  - Bagging: Repeat  $k$  times to generate  $k$  subsets
- Then aggregate through model averaging / majority voting
- Bagging is applied on a collection of low-bias high-variance models
  - by averaging them the bias would not get affected
  - by averaging them the variance will be reduced



# Ensemble Learning

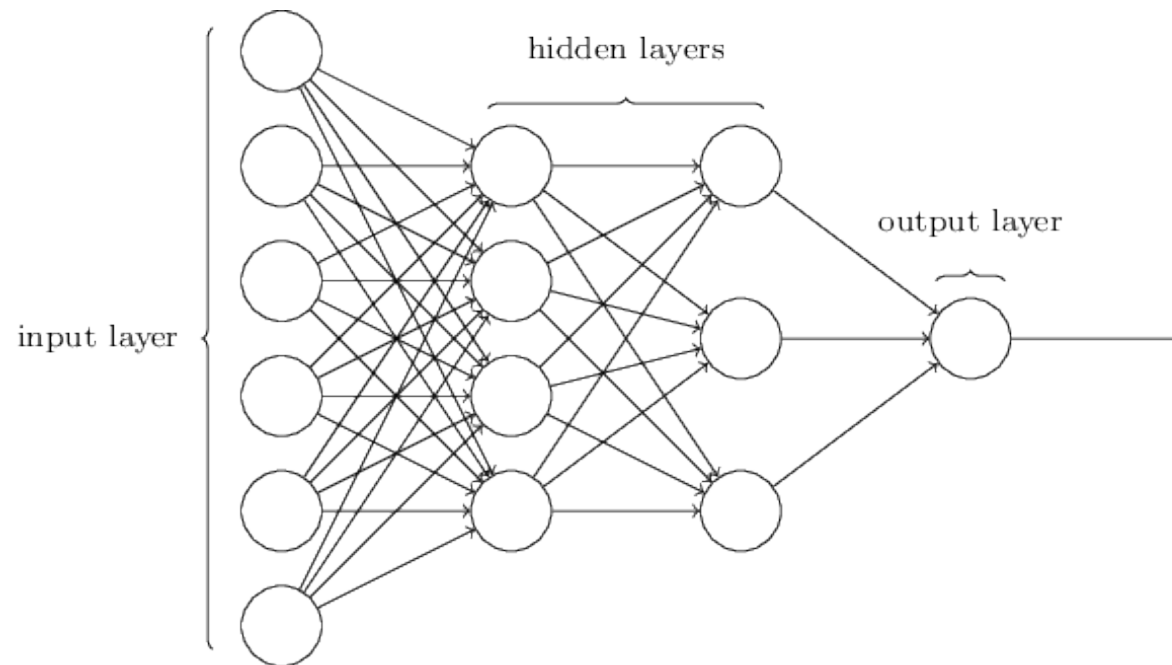
4. Add randomization to the models to introduce more diversity in the models for example
  - For every model use a subset of features, selected randomly, e.g. in Random Forest (it can also help with training time)
  - For algorithms that are dependent on initial weights, use different random initial weights
5. Boosting: A sequence of weak learners, each trying to correct its predecessor
  - Learners are trained sequentially
  - New learners focus on errors of earlier learners
  - New learners try to get misclassified samples right by operating on a weighted training set in favor of misclassified instances
  - Combine all learners in the end using weighted majority/weighted average of  $k$  learners

# Ensemble Learning

- AdaBoost is a boosting algorithm using stump trees
  - Misclassified instances gain higher weights
  - Correctly classified instances lose weight
- Main advantages:
  - Use very simple (weak) learners
  - It boost the performance
  - Decrease bias
  - Decrease variance
- Slow during training and lack of interpretability
- Gradient Boosting is a boosting algorithm using stump tree for regression
  - At every step models the residuals

# Neural Networks

- Neural Nets: composed of a large number of interconnected processing elements known as neurons
  - They use supervised error correcting rules with back-propagation to learn a specific task



# Neural Networks

- Perceptron: Output is thresholded sum of products of inputs and their weights
  - Perceptron learning is simply an iterative weight-update ( $\mathbf{w}' = \mathbf{w} + \eta y_i \mathbf{x}_i$ )
- Multilayer Perceptrons
  - can represent arbitrary functions
  - consists of an input, hidden and output layer each fully connected to the next, with activation feeding forward
- Neural nets are more useful when:
  - Input is high dimensional
  - form of target function is unknown
  - Interpretation is not important

# Neural Networks

- Deep Learning: similar to regular neural nets just with more layers
  - Relies on large amount of data
  - Deeper learning architecture
- Convolutional Neural Net: among the most well-known deep learning models
  - Neurons are arranged in 3 dimensions (width, height and depth)
  - Proposes a parameter sharing scheme that minimize the number of parameters
  - Neurons in each layer are only connected to a small region of the layer before it (not fully connected)
  - Parameters of each layer play the role of a filter which is applied locally
  - The pooling layer: to progressively reduce the spatial size of the representation to reduce the number of parameter. Therefore they help with overfitting

# Neural Networks

- To avoid overfitting:
  - dropout layer is used
    - In each forward pass, randomly set some neurons to zero
  - Early stopping
  - Reduce the network's capacity by removing some layers
  - Regularisation: adding a cost to the loss function for large weights
  - Data Augmentation
    - Increase the data size
    - Rotation, cropping, scaling, flipping, Gaussian filtering

# Evaluation of classification

- For two-class prediction case:

| Actual Class | Predicted Class     |                     |
|--------------|---------------------|---------------------|
|              | Positive            | Negative            |
| Positive     | True Positive (TP)  | False Negative (FN) |
| Negative     | False Positive (FP) | True Negative (TN)  |

- $acc = \frac{1}{|Test|} \sum_{x \in Test} I[\hat{c}(X) = c(X)]$
- $Precision = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$
- $AUC - ROC \text{ curve}$

# Missing Values

How to handle missing values (common approaches):

- Deleting samples with missing values
- Replacing the missing value with some statistics from the data (mean, median, ...)
- Assigning a unique category
- Predicting the missing values
- Using algorithms that support missing values



# Model (Feature) Selection

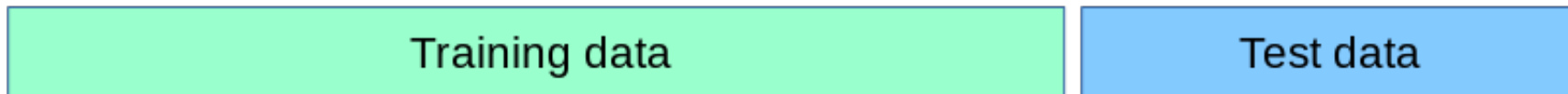
- Taking all the features will lead to an overly complex model. There are 3 ways to reduce complexity:
  - Subset-selection: feature forward selection or feature backward selection
  - Shrinkage, or regularization of coefficients to zero, by optimization. There is a single model, and unimportant variables have near-zero coefficients.
  - Dimensionality-reduction, by projecting points into a lower dimensional space

# Data Normalization

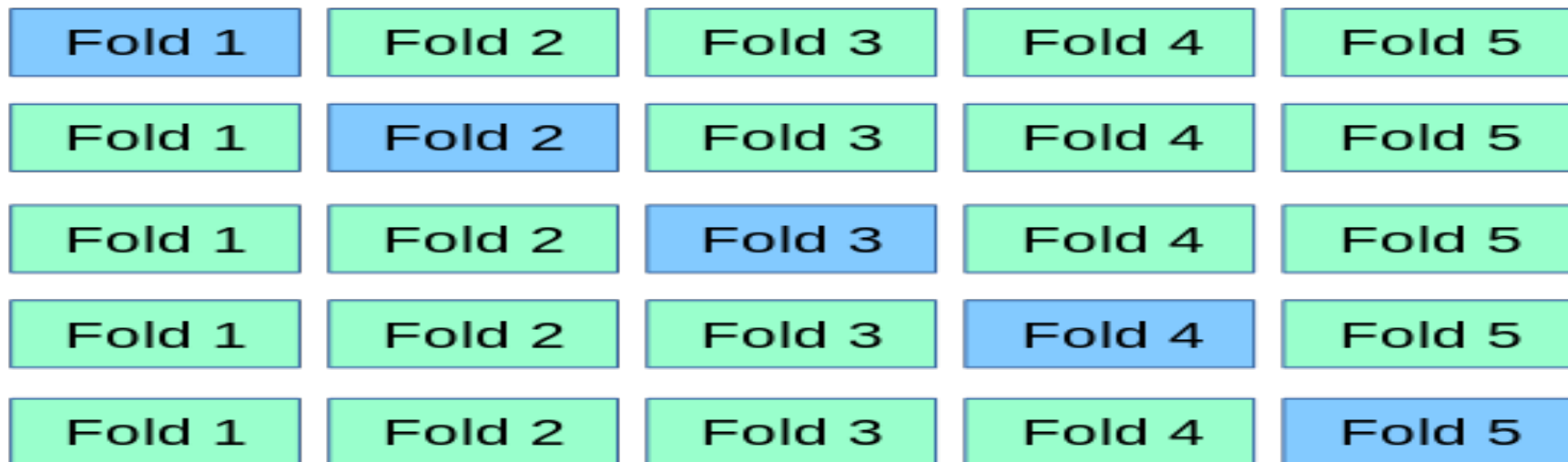
- Normalization is usually a data pre-processing step that change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.
  - Most of the distance based machine learning algorithms require normalization as a processing step if features do not have same scales
  - Most common normalization techniques:
    - Min-max normalization:  $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$
    - Z-score (standardization):  $x' = \frac{x - \bar{x}}{\sigma}$

# Validation

- Hold-out method:



- K-fold cross validation



# Unsupervised Learning

Unsupervised learning: classes are initially *unknown* and need to be “discovered” with their definitions from the data

- It is useful for:
  - Dimensionality reduction (simplify the problem, getting rid of redundant feature)
  - exploratory data analysis
  - to group data instances into subsets
  - to discover structure, like hierarchies of subconcepts
  - to learn new “features” for later use in classification
  - to track “concept drift” over time

# Clustering

- Goal: form homogeneous cluster and well separated clusters
- Success of clustering often measured subjectively
- There are two broad types of clustering:
  - Hierarchical methods
  - Partitioning methods

# Clustering

## 1. K-means

- Initialize  $k$  random centers from the data
- Assign each instance to the closest center and re-compute the centers using mean or weighted average and re-iterate
- Simple and can be efficient clustering method
- Not easy to predict  $k$
- Different initialization can result different clusters
- Sensitive to outliers

# Clustering

## 2. Expectation Maximization:

- Similar to k-means
- Computes probabilities of cluster memberships based on one or more probability distributions. (e.g. mixture of Gaussian)
- The goal is to maximize the overall probability or likelihood of the data, given the (final) clusters.
- Easy with independence assumption

# Clustering

## 3. Hierarchical clustering

- Agglomerative :starts by treating each object as a singleton cluster and gradually merge based on similarity
- Divisive: it starts by including all objects in a single large cluster. At each step of iteration, the most heterogeneous cluster is divided into two. The process is iterated until all objects are in their own cluster.
- Do not require to specify the number of clusters
- Different linkage methods can produce very different dendrograms



# Clustering

- Finding number of clusters:
  - Elbow method: using the within-cluster dispersion
  - Gap statistics: based on the within-cluster variance of original data and B sets of resampled data ( $Gap(k) = \sum_b \log(W_{kb}) - \log(W_k)$ )
  - Choose the number of clusters as the smallest value of k such that the gap statistic is within one standard deviation of the gap at k+1
- Quality of clusters
  - if clusters known, measure proportion of disagreements to agreements
  - if unknown, measure homogeneity and separation
  - silhouette method

# Dimensionality Reduction

- Dimensionality reduction: is the process of reducing the number of feature/attributes
  - Helps with removing redundant/correlated feature
  - Helps with curse of dimensionality
- 1. Principal Component Analysis (PCA): to capture the direction of the most variation in the original data space.
  - Features are not correlated in the new space (they are orthogonal)
  - New dimensions are computed using eigenvectors and eigenvalues of the data matrix (rows are observations and columns are features)
  - Note: Feature have to be normalized before applying PCA
- 2. Autoencoders: A neural network model – the encoder transforms the data into smaller dimension such that the decoder can then interpret and reconstruct with minimum error