COMP9417 Revision

Regression predicts continuous value *(3)*

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 (θ_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 weigh 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 (combination of both)

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

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 *(7)*

- Root Mean Squared Error (RMSE)

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

- Mean Absolute Error (MAE)

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

- R-Squared

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

- Adjusted R-squared

$$R^2_{adjusted} = 1 - \frac{(1-R^2)(m-1)}{m-n-1}$$

m is the total number of samples
n is the number of predictors/features
R^2 represents proportion of variance explained by the model

Classification predicts categories *(8)*

- Generative algorithm
  - builds models for each of the classes
  - Learns $p(x|y)$
  - estimates $p(y|x)$ using Bayes theorm
- Discriminative algorithm
- Learns $p(y|x)$ directly

1. Nearest centroid classifier

- Distance based classifier

$$\mu_k = \frac{1}{|C_k|}\Sigma_{j \in C_k} x_j$$

- for complex classes e.g. multimodal, non-spherical may give poor results
- cannot handle outliers and noisy data
- not very accurate

2. k Nearest Neighbor Classifier (kNN)

- Distance based classifier
- Find k nearest neighbor using distance metric e.g. Minkowski
- Predict output based on majority vote
- Works better with lots of training data and small number of attributes
- can be very accurate but very slow testing
- curse of dimensionality
- Assumes attributes are equally important
  - Remedy: attribute selection &// weights
- Needs homogenous feature type & scale

3. Bayesian decision boundary

- Based on Bayersian theorem $P(h|D) = P(D|h)P(h)/P(D)$
- Prediction will me most probably if expected loss is equal for all classes:
  - Maximum a posteriori

$$h_{MAP} = argmax_{h \in H} P(h|D)$$
$$= argmax_{h \in H} P(D|h)P(h)$$

- If $P(h_i) = P(h_j)$
  we can use maximum likelihood
  $h_{ML} = argmax_{h_i \in H} P(D|h_i)$
- if the estimated loss is nt the same, we have to predict the class which minimizes the expected loss
  EL = $R(a_i|x) = \Sigma_{h \in H} \lambda(a_i|h) P(h|x)$

4. Bayes optimal classification
   $argmax_{v_j \in V} \Sigma_{h_i \in H} P(v_j|h_i) P(h_i|D)$
- Here we are dealing with combing the decision from multiple hypothesis
- No other classification method using the same hypothesis space and same prior knowledge can outperform on average
- Extremely inefficient

5. Naive Bayes Classifier
- Using Bayesian theory
- Strong assumption that attributes are conditionally independent
- Prediction is based on maximum a posteriori
  $P(x_1, x_2, \dots, x_n|v_j) = \Pi_i \hat{P}(x_i|v_j)$
- Useful when moderate / large training set

$v_{NB} = argmax_{v_j \in V} P(x_1, \dots, x_n|v_j) P(v_j)$

$= argmax_{v_j \in V} \hat{P}(v_j) \Pi_i \hat{P}(x_i|v_j)$

- Attributs are conditionally independent
- Usually violated but still decent
- too many attributes decreases performance

6. Decision Tree - Divide and conquer
- Split into subsets (s)
- check subset purity with attributes (A)
  - use entropy to measure impurity
    $E(s) = \Sigma_{i=1}^{s} - p_i \log_2 p_i$
  - Use information gain to decide attribute
    $E(s) - \Sigma_{v \in Values(A)} (|S_v|/|S|) E(S_v)$
  - However IG is more biased towards attributes w/ large number of possibilities, Gain Ratio can be used instead
  - Attributes with highest information gain will be selected for the node
- Can work w/ any data discrete / numeric
- Can handle missing values
- Advantage: interpretability
- Almost always classify training example w/ enough growth (overfitting)
  - Pre-pruning: stop growing when split is not significant w/ chi-squared test
  - set lowest sample leaf, impurity decrease
  - set highest leaf node, max depth
  - Post-pruning, remove sub-trees causing overfitting based on cross validation
- Greedy algorithm (no optimality)

7. Linear Perceptron

$\hat{y} = f(x) = sgn(w.x)$

- Weights get updated iteratively until no mistake is made or max iterations
- Simple and fast at training
- Doesn't perform well if classes are not linearly separable.

8. Non-Linear Perceptron
- Map attributes into new space consisting of polynomial terms and interaction terms
- Use kernel trick to simplify computation
  $\hat{y} = sgn(\Sigma_{i=1}^{m} a_i y_i (\varphi(x_i).\varphi(x)))$
- A valid kernel function is equivalent to a dot product in some space
  $K(x_i, x_j) = \varphi(x_i).\varphi(x_j)$

9. Linear Support Vector Machine
- Maximum Margin
- $\hat{y} = sgn(w.x - t)$
- $w = \Sigma_{X_i \in \{support\ vectors\}} a_i y_i x_i$
- a_i is non-zero for support vectors
- Is effective in high dimensional data
- is effective when number of dimensions bigger than number of samples

10. Nonlinear SVM
- Similar to perceptron, kernel trick can be applied using dual form
- $\hat{y} = sgn(\Sigma_{a_i > 0} a_i y_i K(x_i, x) - t$

Bias-Variance *(20)*
- Bias: The inability of a learning algo to capture the true relationship between output and features / attributes
  - due to model choice (e.g. not complex)
- Variance: The learning algorithm difference between datasets
  - due to small sample size
  - high complexity of model
- aim: good bias variance tradeoff
  - Methods include regularization, ensemble learning, bagging, boosting

Ensamble Learning: meta-algos *(22)*

1. Simple ensambles: combining several
- Majority vote or weighted average
- Treat each output as a feature and train another learning algorithm on them
2. Mixrture of experts
- Each learning algorithm defines $a_i(x)$ which indicates the expertise of that algorithm for that particular x in the space
- Use as weighted average / pick best
3. "Bagging" method: *B*ootstrap *Agg*regation
- Train many classifiers, each with bootstrapped dataset
  - Bootstrap: random subset of data by sampling with replacements
  - Bagging: Repeat k times for k subsets
- Then averaging / majority voting
- Bagging is applied on a collection of low-bias high-variance models
  - bias would not be affected by averaging
  - Variance would be reduced
4. Add randomization to the models to introduce more diversity
- Use a sunset of features, selected randomly, e.g. Random Forest
  - Helps with training time
- Use different random initial weights
5. Boosting: Sequence of weak learners each trying to correct its predecessor
- Learners are trained sequentially
- New learners focus on errors of earliers
- new learners try to get misclassified samples on a weighted training set in favor of misclassified instances
- combine all learners in the end using weighted (majority / average) of k learners
- AdaBoost: boosting algo w/ stump trees
  - Misclassified instances -> higher weights
  - Correctly classified instance lose weight
- Main advantages:
  - Use very simple (weak) learners
  - boosts performance
  - decrease bias and variance
- Slow during training & lack of interpretability
- Gradient boosting is a boosting algo using stump tree for regression
  - At every step models the residuals

Neural Networks *(26)*

- Neural Nets: composed of large number of interconnected processing elements known as "neurons"
- Supervised error correcting rules with back-propagation to learn specific task
- Perceptron: Output is threshold sum of products of inputs and their weights
  - learning is simply iterative weight update

$$w' = w + \eta y_i + x_i$$

- Multilayer Perceptrons
  - Can represent arbitrary functions
  - consists of input, hidden, and output layer each fully connected to the next, with activation feeding forward
- Neural nets more useful when:
  - Input is high dimensional
  - form of target function is unknown
  - Interpretation is not important
- Deep learning: similar with more layers
  - Relies on large amount of data
  - Deeper learning architecture
- Convolutional Neural Net: well-known
  - Neurons are arranged in 3 dimensions, width, height, and depth
  - Proposes a parameter sharing scheme that minimizes the number of parameters
  - Neurons in each layer are only connected to a small region of the layer before it
  - The pooling layer: to progressively reduce the spatial size of the representation to reduce the number of parameters, helps with overfitting
- To avoid overfitting:
  - Dropout layer is used
    - In each forward pass, randomly set some neurons to zero
  - Early stopping
  - Reduce capacity by removing layers
  - Regularization: add cost for large weights
  - Data augmentation: increase data size
    - rotation, cropping, scaling, flipping, gaussian filtering

Evaluation of classification *(30)*

-	Actual	Predicted P	Predicted N
-	Positive	True P	False N
-	Negative	False P	True N

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

Missing Values how to handle *(31)*

- Deleting samples w/ missing values
- Replacing missing values with statistics
  - (mean, median, mode, …)
- Assigning a unique category
- Predicting the missing values
- Using algorithms that support missings

Model (Feature) Selection *(32)*

- all features -> overly complex model.
  - subset-selection: feature forward selection, feature backward selection, or feature importance analysis
  - Shrinkage, or regularization of coefficient to zero. Unimportant variables have near-zero coefficients.
  - Dimensionality reduction - projecting points into a lower dimension space

Data Normalization *(33)*

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

Validation *(34)* Hold-out, Leave One Out Cross Validation, K-fold validation

Unsupervised Learning *(35)*

- Unknown initial classes and need to be discovered with their definitions from data
  - Useful for dimensionality reduction (simplify the problem, getting rid of redundant features)
  - exploratory data analysis
  - group data into subsets
  - discover structure
  - learn new “features” for later use
  - to track “concept drift” over time

Clustering: form homogeneous clusters *(36)*

- well separated clusters
- success often measured subjectively
- Hierarchical / Partitioning methods

1. K-means

- Initialise k random centers from the data
- Assign each to closest center and recompute the centers using mean or weighted average and reiterate
- simple and can be efficient method
- not so easy to predict k
- different initialisation -?> different clusters
- sensitive to outliers

2. Expectation Maximization

- similar to k-means
- computes probabilities of cluster memberships based on one or more distributions (e.g. mixture of gaussian)
- maximize the overall probability or
- likelihood of data, given the final clusters.
- Easy with independence assumption

3. Hierarchical clustering

- Agglomerative: Starts by each object as a singleton cluster and merge by similarity
- Divisive: Include all objects in a single large cluster. At each iteration, the most heterogeneous cluster is divided into two. Repeat until all are in their own cluster.
- Doesn’t require specifying number
- Different linkage methods can produce very different dendrograms

+ Finding the 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
- $Gap(k) = \Sigma_b log(W_{kb}) - log(W_k))$
- Choose the number of clusters as the smallest k value 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, homogeneity and separation
- Silhouette method

Dimensionality Reduction (41)

- reducing number of attributes / features
  - helps w/ removing redundant / correlated feature & helps w/ curse of dimensionality

1. Principal Component Analysis (PCA)

- Features not correlated (orthogonality)
- New dimensions computed using eigenvectors and eigenvectors of the data (rows: observations, columns: features)
- Features have to be normalized before

2. Autoencoders: NNM - encoder transforms the data into smaller dimension such that the decoder can interpret and reconstruct with minimum error