General Q&A

Organization

Are the programming exercises relevant for the exam?

- according to lecture `Introduction` (slide 9), "[e]verything in this lecture and the exercises" relevant for exam
- answer in session: please ask Edouard for further clarification
- after session: Edouard has answered question in discussion forum

Do we have to learn algorithms like ID3 (tree construction) by heart, or is it enough to understand the principles behind it?

- please ask Edouard (he is responsible for the exam)

Will you publish a solution for this session?

- Yes, you are currently reading it.

Will be third exercise session be online as well?

- if your preferences regarding online/offline haven't changed: yes, will be online again
- I will inform you about the format in the forum post / e-mail announcing the session

Content of the Lecture

Does k-fold cross-validation use the same test set each iteration?

- training set and test set both are formed from folds, i.e., partitions, of the full data
- each of the k folds is used for testing in one iteration
- each of the k folds is used for training in $k - 1$ iterations
- i.e., the test sets don't overlap, while the training sets do overlap (unless $k = 2$)

How are the folds created in k-fold cross-validation?

- random partitioning of original data, i.e., each data object assigned to exactly one fold
- instead of purely random assignment to partitions, might use stratification

Theory Tasks

Chapter 3: Indexes

How does the R tree search for the right rectangle to answer a query?

- inspect all rectangles that might contain the (point, range etc.) query
- even for point queries, these might be multiple rectangles, due to overlap in tree
- in contrast, k-d tree partitions data (only area of one leaf node can contain a point)

How does the R tree handle insertion of new points?

- preferably, insert into a rectangle whose area contains the point
- if there is no such rectangle, choose rectangle that needs least enlargement
- if node is full (pre-defined capacity reached), split it up

What kind of index is most useful for a highly imbalanced dataset?

- imbalance (uneven distribution) of target variable not relevant for indexes
  - index uses feature values, not target variable
- imbalance of feature values shouldn't be a problem as well
  - index does not need to partition range of features evenly, but can adapt to distribution
- feature imbalance is problem for equal-width histograms
  - e.g., a few bins might contain many values, many bins might be empty
  - but that has nothing to do with indexes we discussed in lecture . . .

Which query types can be handled by spatial indexes?

- in general: queries on multi-dimensional data
- match queries (points)
- range queries (rectangles); potentially also more complex shapes
- k-NN queries
- ranking queries

Chapter 4: Classification

How can we determine the optimal k for a k-NN classifier?

- split data into training and validation set (e.g., holdout split or cross-validation)
- try out different values of k
- train models on training set only, pick k based on validation-set performance
- typically, one tries small values of k only, e.g., 1 to 10

You are using a k-NN classifier on a dataset with two features (and an additional target variable). The first feature has the range $[0, 1]$ and the second feature has the range $[-10000, 10000]$. Why is this problematic? What can be done to reduce this effect?

- problematic because second feature will have more impact on distance computations
  - due to its larger range (which probably causes larger variance)
  - distance computations are essential for k-NN (define the neighborhood)
- solution: normalization, e.g., min-max scaling

How can we deal with the problem of correlated predictors in linear classifiers?

- apply PCA
  - features after transformation are always uncorrelated (a property of PCA)
  - doing regression after PCA also known as 'principal component regression'
- remove correlated predictors manually, e.g.,
  - loop over pairs of predictors
  - remove one predictor if correlation in pair over some threshold

Explain different ways to avoid overfitting in decision-tree learning!

- pre-pruning: grow smaller decision tree, e.g., by setting
  - minimum support: minimum number of data objects in a leaf
  - minimum confidence: minimum fraction of the majority class in a leaf
  - require maximum tree depth
  - further options might be available, e.g., check sklearn.tree.DecisionTreeClassifier
- post-pruning: cut down decision tree after growing it
- for both approaches: use validation set to decide how far to prune

How can we consider costs of classification errors in decision-tree learning?

- generic solution: re-sample / duplicate data objects according to their class
  - library (like `sklearn`) might allow passing instance weights or class weights instead
- modify objective function, e.g., consider conditional risk
  - for trees, might combine that with impurity criterion (like info gain or Gini index)

What is the prior in Bayesian classification?

- given feature values X and class C, Bayes' theorem is $P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$
- prior probability of feature values, $P(X)$, not interesting for us
  - normalized out when comparing $P(C|X)$ for different classes
- prior probability of classes, $P(C)$, is interesting for us, used in classifier
  - it's just the frequency of classes in data
  - one can also build simple baseline from it (always guess $P(C)$, ignore feature values)

How can we combine different classifiers?

- bagging
- boosting
- stacking
- all these three approaches are summarized under term 'ensembling'

How does boosting reduce bias and variance?

- reduce variance: train multiple models
- reduce bias: iterative training – data objects re-weighted after each training
  - misclassified objects get higher weight
  - thus, following classifiers guided towards improving the errors of previous classifiers

Chapter 5: Evaluation

What is stratification?

- approach for data splitting / property of splits
- each split should be representative of full data (have same distribution)
- typically, we only stratify for target variable (not for features) in cross-validation
- one Python implementation: sklearn.model_selection.StratifiedKFold

How can one describe the classification error in terms of bias and variance?

- mean-squared error (MSE) can be decomposed into bias and variance
- bias: difference between estimate of target value and actual target value
  - can be observed as error on training data (model not powerful enough to fit data)
- variance: variance of estimate of target value
  - can be observed as difference between training and validation/test error (overfitting)
- there usually is trade-off between bias and variance (decreasing one increases other)

Explain the F_β score and what it measures!

- classification performance measure
- range: $[0, 1]$ (higher is better)
- formula: $F_\beta = \frac{(1+\beta^2) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall}$
- harmonic mean of precision (weight: $\frac{1}{1+\beta^2}$) and recall (weight: $\frac{\beta^2}{1+\beta^2}$)
- precision: how accurate are we if we predict 'positive'?
- recall: how accurate are our predictions for the actually 'positive' objects?

Explain Cohen's kappa coefficient!

- in our context: compares one classifier to a reference classifier
- $\kappa = \frac{p_c - p_r}{1 - p_r}$, with p = performance (higher is better), c = our classifier, r = reference
- reference often is a simple baseline like random guessing
- $\kappa = 1 \rightarrow$ always right, $\kappa = 0 \rightarrow$ as good as reference, $\kappa = -1 \rightarrow$ always wrong

What advantage do you see in the informational loss compared to the quadratic loss?

- informational loss: $-\log_2 p_c$, with c being the actual class
- estimating a probability of nearly zero for actual class gets huge penalty
- in particular, loss can become arbitrarily high, while quadratic loss is bounded
- which loss measure is better depends on your use case