Week 5: Data Analysis

Comparing ML algorithms

Topics

- Evaluating prediction algorithms
  - training and test errors
  - regression models
  - classification models
- Overfitting
  - bias and variance
  - regularisation
- Comparing algorithms
  - testing the null-hypothesis
  - p-values

- Evaluating prediction algorithms

Training and testing

- Goal:
  - to train algorithm on existing data
  - so that they will perform well on unseen data.

- In practice, we need both
  - a training and testing data set

- Can algorithms that perform well on the training data perform poorly on the test data?

Learning machines

- The learning process consists of
  1. Defining a learning machine $F: \mathcal{X} \times \Lambda \to \mathcal{Y}$
  2. Training the learning machine
     - using a given training data set,
       i.e. Finding $\hat{f}(x) = F(x, \hat{\lambda})$
  3. Specifying a prediction rule
     - which estimates the unseen label given a test object:
       - $\hat{y} = \hat{f}(x)$
       - $\hat{y} = g(\hat{f})$

     e.g. g can be the Bayes rule with the unknown conditional probability

Prob(y|x) approximated by $\hat{f}(x)$

Training

- Imagine we select a model
  i.e. $\hat{\lambda}$.
  - that minimizes the training risk,
    i.e. the risk computed on the training data.

- How can be sure that the model performs well on the testing data too?

- Assuming that training & testing data are similar,
  i.e. drawn from the same distribution, we may expect
  $g_{train}(\hat{f}) \approx g_{test}(\hat{f})$

- Empirical risk minimization principle
  - Given a learning machine $F: \mathcal{X} \times \Lambda \to \mathcal{Y}$
  - choose $\hat{\lambda} \in \Lambda$ which minimizes the empirical risk,
  i.e.
  $$\hat{\lambda} = \arg\min_{\lambda \in \Lambda} g_{train}(F(X, \lambda)).$$

Evaluation of a prediction rule

- The evaluation of a given prediction rule should not
  - depend on how the model is trained, i.e.
  1. The labelled test data set used for valuation
     - must be different from the training data set
  2. The cost function used for the evaluation
     - may be different from the one used for training.

- If you measure the performance of your algorithm
  - on the training data set,
  - you should specify that you performed an in-sample evaluation.

- Conformal predictions
  - The prediction rule can be also used to make conformal predictions.
  - The goal of conformal prediction (CP)
    - is to quantify how much you can trust the predictions of a given algorithm.

Evaluation of regression models

- The prediction rule of regression model is often straightforward, i.e. $\hat{y} = \hat{f}(x)$.
- You can measure the quality of the predictions using either the
  1. Mean Square Error (MSE)

$$MSE = |D|^{-1} \sum_{(x,y) \in D} (y - \hat{f}(x))^2$$

or 2. the Mean Absolute Error (MAE)

$$MAE = |D|^{-1} \sum_{(x,y) \in D} |y - \hat{f}(x)|$$

but other choices are also possible.

Note that: The MSE is a normalized version of RSS
- that does not depend on the size of the data set.

Evaluation of classification models

- The output of classification models
  - is a vector of probabilities,
    e.g. $\hat{f} = [\hat{f}_1, \ldots, \hat{f}_{|\mathcal{Y}|}]^T$ such that $\sum_{y \in \mathcal{Y}} \hat{f}_y(x) = 1$ for all x.
- Each element of $\hat{f}$ is the probability of a test object belongs to a given class
  - and the corresponding prediction is

$$\hat{y} = \arg \max_{y \in \mathcal{Y}} \hat{f}_y(x_0)$$

- Classification models can be trained by minimizing either
  i) the empirical error rate or
  ii) the negative log-likelihood of the training set

What is a good quantity for evaluating $\hat{f}$?

- Given an unseen test object, (x_0, y_0) you can either evaluate
  1. The probability of the correct class,
     i.e. the likelihood of (x_0, y_0), $\hat{f}_{y_0}(x_0)$

  2. The prediction itself, $\hat{y}$

- The latter is more
  - practice,
    e.g. more similar to what the algorithm be used for
  - general,
    e.g. allows the comparison with non-probabilistic prediction approaches (for which $\hat{f}(x_0)$ is not available

Multi-class classification models

- Let $\mathcal{D}_{test} = \{(x_n, y_n)\}_{n=1}^{N}$ be a test data set

and $\hat{y}_n = \arg\max \hat{f}(x_n)$ the corresponding predictions.

- For multi-class classification problems we have $|\mathcal{Y}| > 2$
- Models can be evaluated by measuring the error rate on the test data set,
  i.e.

$$ER(\mathcal{D}_{test}) = |\mathcal{D}_{test}|^{-1} \sum_{n=1}^{N} 1[\hat{y}_n = y_n]$$

Binary classification models

- In the binary-classification setup,
  - we have $\mathcal{Y} = \{0, 1\}$
- The learning machine is a scalar function,
  - as $\hat{f}_0(x) = 1 - \hat{f}_1(x)$
  - is the predicted probability that x has label 0.

- In this case, you can use

1. A simplified version of the error rate, ER()

$$ER(\mathbb{D}_{test}) = |\mathbb{D}_{test}|^{-1} \sum_{n=1}^{N} (\hat{y}_n - y_n)^2$$

2. The area under the Receiver operating characteristic (ROC) curve

- The latter is usually referred to as AUC and takes into account both the model sensitivity and specificity

$\Rightarrow$ True and false positives and negatives

- Let $\{y_n \in \{0,1\}\}_{n=1}^{N}$ the set of the true binary labels of the test set and

$\{\hat{y}_n\}_{n=1}^{N}$ the corresponding predictions.

The ROC curve is built by counting the numbers of

1. True positive
   i.e. pairs $(y_n, \hat{y}_n)$ such that $y_n = \hat{y}_n \neq \hat{y}_n = 1$
2. True negative
   i.e. pairs $(y_n, \hat{y}_n)$ such that $y_n = \hat{y}_n \neq \hat{y}_n = 0$
3. False positive
   i.e. pairs $(y_n, \hat{y}_n)$ such that $y_n \neq \hat{y}_n \neq \hat{y}_n = 1$
4. False negative
   i.e. pairs $(y_n, \hat{y}_n)$ such that $y_n \neq \hat{y}_n \neq \hat{y}_n = 0$

Sensitivity, specificity, accuracy

- The ratios

$$TPR = \frac{TP}{P} = \frac{\#\text{ of true positives}}{\sum_{n=1}^{N} y_n} = \frac{TP}{TP+FN}$$

TP	FP
FN	TN

$$TNR = \frac{TN}{N} = \frac{\#\text{ of true negatives}}{\sum_{n=1}^{N} \mathbb{1}[y_n = 0]} = \frac{TN}{TN+FP}$$

- Define the model sensitivity, TPR, and specificity, TNR.
- The accuracy of the model is

$$ACC = \frac{TP+TN}{P+N} \qquad TNR = \frac{TN}{(TN+FP)}$$

The Area Under the Curve

- Given $\{(y_n, \hat{y}_n) \in \{0,1\}^2\}_{n=1}^{N}$
  - there is a finite number $(N+1)$ of thresholds. $\{\xi_{n'}\}_{n'=0}^{N}$
  - such that a threshold-based prediction rule

$$\hat{y}_\xi = \mathbb{1}[\hat{f}(x) \geq 1-\xi], \quad \text{Assume: } \hat{f}(x) \in (0,1)$$

  - produces $N+1$ monotonically increasing FPR values e.g.

$$FPR(\xi = 0) = 0 \text{ and } FPR(\xi = 1) = 1$$

- The AUC is a good summary of the model performance.
- Formally, the AUC is the probability that a classifier will
  - rank a randomly chosen positive instance
  - higher than a randomly chosen negative one.

Overfitting

2 problems of ERM (Empirical risk Minimization)

- Given a learning machine & two models
  e.g. $\hat{\lambda}_1$ and $\hat{\lambda}_2$
  - there is no guarantee that the model with
  - the lowest training risk
  - will also have the lowest test risk
    i.e.

$$g_{train} = (F(X, \hat{\lambda}_1)) < g_{train}(F(X, \hat{\lambda}_2))$$

does not imply

$$g_{test}(F(X, \hat{\lambda}_1)) < g_{test}(F(X, \hat{\lambda}_2))$$

- $\hat{\lambda}$ may be chosen to capture specific details of
  - a small training set.

A U-shape behaviour

- As the level of flexibility increases, the curves fit the observed data more closely.
- The green curve is the most flexible & matches the data very well
  - However, it fits the true f poorly b/c it is too wiggly
- Flexibility is quantified by the degrees of freedom of a model.
- The MSE_{train} declines monotonically as flexibility increases.
- The MSE_{test} is computed over a large test set as a function of flexibility (not possible when f is unknown).
- As for MSE_{train}, MSE_{test} initially declines as the
  - level of flexibility increases
  - but at some point the MSE_{test} starts to increase

Overfitting

- The U-shape behaviour is general,
  I.e. - it is observed regardless of the particular data set at hand
  - and regardless of the prediction algorithm being used

- An algorithm with a small MSE_{train} but a large MSE_{test} is overfitting the data.
- An algorithm with a high MSE_{train} and a high MSE_{test} is underfitting the data.

- Overfitting
  - occurs when the algorithm works so hard to pick up patterns of the training set
  - that are just caused by noise ε
  - (rather than by true properties of f).

Goal (regression models)

- Let $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}$ be a random data set
  - generated by the noisy model

$$Y_i \sim f(X_i) + \varepsilon, \quad i = 1, \ldots, N$$

- Given $D \sim \mathcal{D}$, a realisation of $\mathcal{D}$,
  - the goal is to use D to find a regression model $\hat{f}(X, D)$
  - that approximates $f(X)$ for
    - all $(x, y) \in D$
    - for any other unseen pair (x_0, y_0)
      - generated by the same joint distribution

Note that: - $\hat{f}(X, D)$ is sample of $\hat{f}(X, \mathcal{D})$,
- as the estimate is computed using the sample $D \sim \mathcal{D}$

Expected error

- To understand why we observe the U-shape behaviour
  - we consider the variability of the expected error
  - at a given point over different choices of the
  - training data set,

i.e.

$$E_D\left((\hat{f}(x_0, D) - y_0)^2\right) = E_D\left((\hat{f}(x_0, D) - f(x_0) - \varepsilon)^2\right)$$

where;

- The expectation is over the random variable $\mathcal{D}$
- The Expected error at x_0 is the sum of 3 terms.

$$E_D\left((\hat{f}(x_0, D) - y_0)^2\right) = \text{bias}_{\hat{f}}(x_0)^2 + \text{variance}_{\hat{f}}(x_0) + \text{Var}(\varepsilon)$$

$$\text{bias}_{\hat{f}}(x_0) = E_D\left(\hat{f}(x_0, D) - f(x_0)\right).$$

$$\text{Variance}_{\hat{f}}(x_0) = E_D\left(\left(E_D\left[\hat{f}(x_0, D)\right] - \hat{f}(x_0, D)\right)^2\right)$$

Irreducible error

- The irreducible error
  - is the (unknown) lower bound on the accuracy
  - of our prediction for Y,
    i.e. the lowest possible expected error at x_0

  - is defined as

$$\mathrm{Var}(Y) = E_D((Y - f(x_0))^2) = E_D((\varepsilon - E_D(\varepsilon))^2) = \mathrm{Var}(\varepsilon)$$

Note that: - $\mathrm{Var}(\varepsilon)$ does not depend on the model estimate $\hat{f}(x_0, D)$,
i.e. there is nothing you can do to reduce it.

Bias

- The model bias
  - is the error that is introduced by approximating a real-life problem with a much simpler model.
- The bias of $\hat{f}$ on (fixed) input x_0 is defined as

$$E_D(f(x_0) - \hat{f}(x_0, D)) = f(x_0) - E_D(\hat{f}(x_0, D))$$

- The bias is caused by simplifying assumptions in $\hat{f}$,
  e.g. if
$f(x) = 1 + x + 3x^2$ and $\hat{f}(x) = F(\hat{\lambda}, x) = \lambda_1 + \lambda_2 x \,|\, \lambda = \hat{\lambda}$

- On real-world data, simplifying assumptions are unavoidable

Variance

- The model variance
  - is the amount by which our estimate $\hat{f}(x_0, \mathcal{D})$ would change
  - if we estimated it using a different training sets,
  - the difference between $\hat{f}(x_0, \mathcal{D})$ and $\hat{f}(x_0, \mathcal{D}')$

- The variance of $\hat{f}(x_0, \mathcal{D})$ at x_0 is defined as

$$E_{\mathcal{D}}\left(\left(\hat{f}(x_0, \mathcal{D}) - E_{\mathcal{D}}(\hat{f}(x_0, \mathcal{D}))\right)^2\right)$$

Ideally $\hat{f}(x_0, \mathcal{D})$ should not vary too much between training sets (low variance).

U-shape explained

- The U-shape observed in MSE_{test}
  - is the result of two competing term of $\hat{f}(x_0, \mathcal{D})$, $\text{bias}\hat{f}(x_0)$ and $\text{variance}\hat{f}(x_0)$

- The decomposition tells us that
  - in order to minimize the expected test error,
  - we need to select a prediction algorithm (a model class) $\hat{f}(X, \mathcal{D})$
  - that simultaneously achieves low variance & low bias.
- More flexible statistical methods
  - have higher variance;
    - e.g. the green model of slide 6,
      - as they try to capture all details of a specific $\mathcal{D}$
- Simpler models
  - have higher bias
  - because it is unlikely that real-life problem is truly simple

- The U-shape behaviour consists of two phases.

1. A low-flexibility phase
   - where the (squared) bias tends to initially decrease faster that the variance
   - (the expected test error decreases)

2. A high-flexibility phase
   - where increasing flexibility has little impact
     - on the bias
     - but starts to significantly increase the variance
   - (the expected test error increases)
   - when this happens the test MSE increases.

Good models

- bias-variance trade-off
  - is a relationship between bias, variance, and test set MSE.
- The model-class flexibility
  - is usually chosen a priori
  - and it is challenging
  - to find a method for which both the variance & the squared bias are low.

For example,
The complexity of a decision-tree model
- is proportional to the number of leaves
- and may depend on a specific
- stopping criterion fixed in advance

Regularization

- by using a regularization technique
  - it is possible to control the bias & variance
  - of the output model during training

Regularisation reduces the complexity of the output & hence
- regularised models normally have
  - higher bias
  - lower variance.

- A popular strategy in machine learning
  - is to use highly flexible model class
    e.g. - neural networks, and
    - apply strong regularisation schema during training.

Example: pruning decision trees

- Pruning decision tree
  - is an example of regularised learning
- In the case of pruning the regularisation
  - is applied after building a very flexible model,
- in a two step process:

1. learn a very flexible (and overfitting) decision-tree model, To
   e.g. - Stop the iterative splitting
   - when you have only one sample per leaf.

2. prune certain nodes of the tree
   - to reduce the complexity of the final model, $T \subseteq T_o$

Cost complexity Pruning

- Cost complexity pruning
  - is a pruning technique that produces
    - a small set of subtrees,
    - ordered according to their complexity.
- The sequence of sub trees $T_\alpha \subseteq T_0$
  - is defined through the corresponding penalized RSS,
  - i.e.

$$T_\alpha = \arg\min_{T \subset T_0} \left(\alpha|T| + \sum_{j=1}^{|T|} \sum_{i: x_i \in R_j} (y_i - \hat{y}_{R_j})^2 \right)$$

$$\hat{y}_{R_j} = \frac{\sum_{i: x_i \in R_j} y_i}{\sum_{i: x_i \in R_j}}$$

where; • $|T|$ is the number of leaves / regions
• and $\hat{y}_{R_j}$ the predicted label of an object in region R_j, $j = 1, \ldots, |T|$

- The best subtree
  - is selected by measuring the unpenalized
  - performance of the models on a test set.

- The tuning parameter α controls the trade-off between
  - the complexity of T
    i.e. the number of terminal nodes $|T|$
  - the RSS fit of T to the training data

- As α increases,
  - the price to pay for having a tree with many terminal nodes is higher and
  - the minimum of penalised objective is a smaller subtree.
- The right value of α
  - can be obtained using
    - a validation set or
    - (which is more common) using cross-validation.

Overall algorithm for building a regression tree

1. Use recursive binary splitting
   - to grow a large tree on the training data,
   - stopping only when each terminal node
   - has fewer than some minimum number of observations.

2. Apply cost complexity pruning
   - to the large tree
   - in order to obtain a sequence of best subtrees,
   - as a function of α.

- Regularizing linear regression
  - The parameter space of a linear regression learning machine

$$F(x, \lambda) = [1, x^T]\lambda$$

  - is constrained by the size of x,
    e.g. $\Lambda = \mathbb{R}^{d+1}$ if $x \in \mathbb{R}^d$

  - This may cause overfitting problem
    - if, for example, $|D_{train}| \geq d$

- A usual technique to avoid overfitting
  - in these cases is to train the model
  - by adding an L_2-penalization term to
  - the least square objective,
    i.e. by letting

$$\hat{\lambda} = \arg\min_{\lambda \in \Lambda} J(D, \lambda) + \rho\|\lambda\|^2, \quad J(D, \lambda) = \sum_{(x,y) \in D_{train}} ([1, x^T]\lambda - y)^2$$

- The obtained model is usually called
  - a ridge regression model.

Comparing algorithms

Comparing two prediction algorithms

- A simple problem:
  - we train two classification algorithms and
  - test them on the same test set.
- The test results are summarized by a contingency table

e.g.

Algorithm 0 \ Algorithm 1	correct	wrong
correct	189	9
wrong	17	23

TP	FP
FN	TN

- Total number of observations: $189+9+17+23 = 238$
- Algorithm 0's error rate: $\frac{FN+TN}{\text{Total observations}} = \frac{17+23}{238} = 16.8\%$
- Algorithm 1's error rate:

$$\frac{FP+TN}{\text{Total observations}} = \frac{9+23}{238} = 13.4\%$$

Setting

- Let us first suppose that:
  - Algorithm 0 (A0) is the state-of-the-art algorithm
  - Algorithm 1 (A1) is a new algorithm

We want to prove that Algorithm 1 is better.

Question: Is the difference between 13.4% and 16.8%
statistically significant?
or could it be a statistical fluke?

Strategy

- The comparision strategy can be summarized in 3 main steps:
  1. Discard all tests
     - where both algorithms made a correct or wrong prediction
  2. let
     - N the number of remaining tests,
     - k the number of testes
     - where; A1 was correct and A0 wrong
       i.e. the number of successes for A1.
  3. Compare our results the expected results
     - for a binomially distributed random variable,
       i.e. a random variable modelling the number of successes in N tries.

Null hypothesis

- Is it possible that the probability of error is in fact the same for the two prediction rules?
  - This is our null hypothesis.
- (Or is even worse for Algorithm 1; let's ignore this possibility for now).

- In this case, the null hypothesis is equivalent to say:
  - the number of success for A0
  - is what we expect from a binomially distributed random variable,
    i.e. - a random variable representing
    - the number of success in N tries with $p = 1/2$.
- Given N and k we know that

$$\text{Prob}(Y=k) = \binom{N}{k} p^k (1-p)^{N-k}$$

- Let the probabilities for a test observation to get into each of the cells (A–D) be:

	Algorithm 1 correct	Algorithm 1 wrong
Algorithm 0 correct	PA	PB
Algorithm 0 wrong	PC	PD

Where; $PA + PB + PC + PD = 1$

- The null hypothesis is $PC + PD = PB + PD$
- This is equivalent to $PB = PC$

- The null hypothesis can be restated as:
  - the conditional probability that a test observation belongs to cell B given that it belongs to B or C is $\frac{1}{2}$

Testing the null hypothesis

- So let us concentrate on the two cells,
  - B (9 observations) and
  - C (17 observations)

where, the two prediction rules produce different results.

- Our question can be restated as:
  - Can we get only 9 heads in 26 tosses of a fair coin?
- In principle we can, but perhaps this is a rare (unlikely) event.

Computing the P-value

- The p-value corresponding to our observation of 9 out of 26 is

```
> phinom(9, 26, 0.5)      = 0.084
```

- This is the probability that we will observe 9 or even fewer heads in 26 tosses.

- In other words:
  - the probability that we will observe a result as strange as,
  - or even stranger than,
  - the one we have actually observed
  - (under the null hypothesis).

Functions in R

- pbinom is the distribution function of the binomial distribution
- pbinom(k, n, p)
  - is the probability Prob(Y ≤ k),
  - where Y is the number of heads in n tosses of a coin
  - (perhaps) with probability p of a head.
- We could have
  > pbinom(9, 26, 0.5, lower.tail = TRUE) = 908

- p(k, n, p, lower.tail = FALSE)
  - is the probability prob(Y > k) (not "≥"!)
  - this is the probability of an "upper tail")

General definition of p-values

- In general,
  - we choose a test statistic T (a function of the outcome)
  - and decide whether large or small values of the statistic are significant
  - (by default, large values are significant).
- The interpretation of T
  - it measures the strangeness of the outcome

- Let t_0 be the observed value of the test statistic
  - The p-value is Prob($T \geq t_0$)
    - (if large values of T are significant)

  - If small value of T are significant,
    - the p-value is Prob($T \leq t_0$).

Interpretation of p-values

- Let us choose a significance level ε,
  - which is a small positive number.
  - (Customary values: $\varepsilon = 5\%$ and $\varepsilon = 1\%$.)
- Interpretation:
  - we consider fixed events of probability ε as rare (or unlikely).
- The probability that the p-value
  - is at most ε does not exceed ε:
    $\text{Prob}(p \leq \varepsilon) \leq \varepsilon$. Why?
- Therefore, $p \leq \varepsilon$ is a rare event (under the null hypothesis)
- So, - when the p-value is ε or less,
  - we have a disjunction:
    - either the null hypothesis is wrong or
    - a rare event has happened.

- If the p-value is ε or less; p-value $\leq \varepsilon$
  - we reject the null hypothesis.
  - Otherwise, we retain (but never accept) it.
- If we reject the null hypothesis
  - at the significance level 5%
  - our result is statistically significant.
- If we reject the null hypothesis
  - at the significance level 1%,
  - our result is highly statistically significant

Two-sided tests

- So far
  - we have assumed that one of the two prediction algorithms (Algorithm 0) is the base one,
  - and we are only looking for deviation from the null hypothesis in on direction

- Our alternative hypothesis is one-sided:
  - namely, it is that Algorithm 1 produces
  - a better prediction rule than Algorithm 0.

- If there is no such asymmetry
  - and we are just interested in which prediction rule is better,
  - we can simply multiply the p-value that we get as described above by 2.
- In this case, our alternative hypothesis is two-sided:
  - the first algorithm produces a better prediction rule
  - than the second or vice versa.