Machine Learning
V03: Model Assessment & Selection

Data handling for model evaluation
Measures for model performance
Selecting among competing models

Based on (slides of):
Witten, Frank, «Data Mining (2nd Ed.)», 2005, Ch. 5
Duda et al., «Pattern Classification (2nd Ed.)», 2000, Ch. 9
Mitchell, «Machine Learning», 1997, Ch. 5-6
Javier Béjar, BarcelonaTech
Murphy, «MLAPP», 2012, Ch. 5.3
Educational objectives

- **Understand** the need to **use** the available **data wisely** and **know how** to do it **correctly**

- **Explain** the influence of **bias** and **variance** on a model’s performance

- **Remember** prevalent **figures of merit** to document model performance

- **Use** sound **experimental setup** to **evaluate** and **choose** among models
1. DATA HANDLING FOR MODEL EVALUATION

How to learn and evaluate algorithms based on limited data?
How to deduce true error from training error?
Model assessment & selection

Model Assessment: **evaluating** a model’s **performance** (→ next 2 sections)
Model Selection: **selecting** among competing models one with a **proper level of flexibility**

Competition on two levels: different parameters ($\theta$) and different hypothesis spaces ($\mathcal{H}$)
# How to make the most of (small) data

## Training & evaluating hypotheses with limited data

1. **Data**

2. **Training set (ca. 70%)**  **Test set (ca. 30%)**

For big data (especially with deep learning) scenarios, see note on Ng’s talk @ NIPS 2016 in V06.

Don’t let *any* training procedure ever see any labels; better lock away all test data until final test!
How to make the most of (small) data
Training & evaluating hypotheses with limited data

1. Data

2. Training set (ca. 70%)  Test set (ca. 30%)
   
   Dilemma: all 3 sets should be large for good estimates of $h$ as well as for the true error.

3. Training set (ca. 60%)  Validation set (ca. 20%)  Test set (ca. 20%)

For big data (especially w/ deep learning) scenarios, see note on Ng’s talk @ NIPS 2016 in V06

Don’t let any training procedure ever see any labels; better lock away all test data until final test!

Use validation set for parameter optimization & estimating the true error (→ that’s a mere heuristic!)

Insidious form of “testing on training data”: do many repeated optimization trials on same validation set.
How to make the most of (small) data
Training & evaluating hypotheses with limited data

1. Data

2. Training set (ca. 70%)  Test set (ca. 30%)
   **Dilemma:** all 3 sets should be large for good estimates of \( h \) as well as for the true error.

3. Training set (ca. 60%)  Validation set (ca. 20%)  Test set (ca. 20%)
   Train \( k \) times on \((k - 1)\) folds, validate on the remaining one (until each fold was used for validation once); average the error.  Make sure CV is the loop around any optimization!

4. \( k \)-fold cross validation (CV) \((k = 5 \ldots 10)\)  Test set (ca. 20%)

For big data (especially w/ deep learning) scenarios, see note on Ng’s talk @ NIPS 2016 in V06

Don’t let any training procedure ever see any labels; better lock away all test data until final test!

Use validation set for parameter optimization & estimating the true error (→ that’s a mere heuristic!)

Insidious form of “testing on training data”: do many repeated optimization trials on same validation set.

This is best practice. See [ISL, 2014, ch. 5.1] for reasons.
How to make the most of (small) data
Training & evaluating hypotheses with limited data

1. Data

2. Training set (ca. 70%)  Test set (ca. 30%)
   
   Dilemma: all 3 sets should be large for good estimates of $h$ as well as for the true error.

3. Training set (ca. 60%)  Validation set (ca. 20%)  Test set (ca. 20%)
   
   Train $k$ times on $(k-1)$ folds, validate on the remaining one (until each fold was used for validation once); average the error.

4. $k$-fold cross validation (CV) ($k = 5..10$)  Test set (ca. 20%)
   
   Make sure CV is the loop around any optimization!

5. $k$-fold cross validation ($k = 5..10$) on all data

For big data (especially w/ deep learning) scenarios, see note on Ng’s talk @ NIPS 2016 in V06

Don’t let any training procedure ever see any labels; better lock away all test data until final test!

Use validation set for parameter optimization & estimating the true error (→ that’s a mere heuristic!)

Insidious form of “testing on training data”: do many repeated optimization trials on same validation set.

This is best practice. See [ISL, 2014, ch. 5.1] for reasons.

Statistically sound, but seldom seen and with a smell: using all data for CV.
Observable and unobservable errors
Or: why we need to *estimate* the true error

True error $E_D$
- Probability that $h$ will misclassify a random instance from complete domain $D$
- Unobservable

Empirical / test error $E_{emp}$
- Proportion of examples from sample $S \in D$ misclassified by $h$
- Estimate for true error, gets better with more data

Training error: proportion of training data misclassified $\rightarrow$ hopelessly optimistic estimate

How good is the estimate?
- Assumption: training and test data are representative of underlying distribution of $D$
- $S$ and $h$ are usually not chosen independently $\rightarrow$ Test error is *optimistically biased*
- Test error usually varies for different $S \in D$ $\rightarrow$ It has higher variance than the true error

$\rightarrow$ **Confidence intervals** give bounds depending on the test set size ($\rightarrow$ see appendix)
Sources of error
The bias-variance trade-off

Different sources for committed errors
• Chosen $\mathcal{H}$: the best hypothesis is at a distance of the true function
• Chosen $X$: different samples give different information
• Uncertainty in $(X,Y)$ and its representation
  • Partial view of the task: have all relevant features been observed?
  • Corrupted data: are all labels 100% correct?

Error decomposition
• $E_{\text{MSE}} = \text{systematic error} + \text{dependence on specific sample} + \text{random nature of process}$

  - Bias: average prediction’s deviation from the truth
  - Variance: sensitivity of prediction to specific training sample
  - Irreducible error or Bayes’ rate: due to “noise” (→ see appendix)

• Very helpful in comparing & evaluating learning algorithms (→ more in V06)
• Generally, for a more complex/capable model: ↓ bias, ↑ variance
• It’s a trade-off: Only way to reduce both is to increase the size of the sample
2. MEASURES FOR MODEL PERFORMANCE

A quick overview
Evaluating class predictions
Two types of error and their cost

Standard error measure
• Error \( E = \frac{1}{N} \sum_{i=1}^{N} (1 - id(\hat{y}_i, y_i)) \), where \( id(a, b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{else} \end{cases} \) is the identity function

Components of \( E \)
• Contingency table:
  - (Also extendable to the multi-class case)

<table>
<thead>
<tr>
<th>( \downarrow y, \hat{y} \rightarrow )</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>true positive (TP)</td>
<td>false negative (FN)</td>
</tr>
<tr>
<td>0</td>
<td>false positive (FP)</td>
<td>true negative (TN)</td>
</tr>
</tbody>
</table>

What if different (wrong) predictions have different costs attached?
• Example Terrorist profiling: “Not a terrorist” correct 99.99% of the time
• Classification with costs (→ see appendix):
  • Attach costs to each cell in the matrix above («cost matrix»)
  • Replace sum of errors with sum of costs per actual prediction
Measures based on contingency tables
Evaluating *fixed points* in the parameter continuum

- **Accuracy** $\frac{TP+TN}{TP+TN+FP+FN}$: Standard measure that doesn’t regard different «costs» of errors
- **Kappa statistic** for inter-rater agreement: Useful to show relative improvement over random predictor

- From information retrieval domain (used far beyond!)
  - **Recall** $\frac{TP}{TP+FN}$: How many of the *relevant* documents (i.e., $y = 1$) have been *returned* (i.e., $\hat{y} = 1$)?
  - **Precision** $\frac{TP}{TP+FP}$: How many of the *returned* documents are actually *relevant*
  - **F-measure** $\frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}}$: Combination of recall & precision via their harmonic mean
  - There’s a **trade-off** between recall and precision because they show the two different types of error

- From medical domain
  - **Sensitivity** (=true positive rate, recall) $\frac{TP}{TP+FN}$
  - **Specificity** (=true negative rate) $\frac{TN}{TN+FP}$

- Taking all possible operating points between the two errors into account (→ see next slide)
  - **AUC**: Area under ROC curve
  - For recall-precision curves, the farther away from a straight line they are, the better
Measures based on contingency tables (contd.)
Grasping the *trade-off* between type-I and type-II error

<table>
<thead>
<tr>
<th>Domain &amp; content</th>
<th>Plot</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lift chart</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>see appendix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marketing</td>
<td><img src="image" alt="Lift Chart" /></td>
<td>$TP$</td>
</tr>
<tr>
<td>TP vs. subset size</td>
<td><img src="image" alt="Lift Chart" /></td>
<td>$N$ (ordered by predicted probability of being pos.)</td>
</tr>
<tr>
<td><strong>ROC curve</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>see next slide</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communications</td>
<td><img src="image" alt="ROC Curve" /></td>
<td>$\frac{TP}{TP + FN}$</td>
</tr>
<tr>
<td>TP rate vs. FP rate</td>
<td><img src="image" alt="ROC Curve" /></td>
<td>$\frac{FP}{FP + TN}$</td>
</tr>
<tr>
<td><strong>Recall-precision curve</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information retrieval</td>
<td><img src="image" alt="Recall-Precision Curve" /></td>
<td>$\frac{TP}{TP + FN}$</td>
</tr>
<tr>
<td>recall vs. precision</td>
<td><img src="image" alt="Recall-Precision Curve" /></td>
<td>$\frac{TP}{TP + FP}$</td>
</tr>
</tbody>
</table>
ROC curves
Receiver operating characteristic

History
• Used in signal detection to show trade-off between **hit rate** and **false alarm rate** over noisy channel

Construction
• y axis shows percentage of true positives in sample
• x axis shows percentage of false positives in sample
• Train different models, **varying** the parameter(s) that control the «**strictness**» of the method; for each parameter value, draw a point

Interpretation
• Straight line indicates a random process
• Jagged curve: created with one set of test data
• Smooth curve: created using averages from cross validation
Evaluating numeric predictions

Same strategies as for classification (i.e., independent test set, cross-validation, etc.)

Different error measures
- Given: Actual target values ("labels") $y_1, y_2, ..., y_N$, predicted values $\hat{y}_1, \hat{y}_2, ..., \hat{y}_N$
- Most popular measure: Mean-squared error $E_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$ (why? → see appendix of V02)
- Root mean-squared error: $E_{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2}$
- Less sensitive to outliers: Mean absolute error $E_{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|$
- Many more: relative errors (e.g., 10% for predicting 500 and being 50 off), correlation coefficient, ...

Example
- Algorithm D is best, C is 2\textsuperscript{nd} best
- A, B arguable

$\Rightarrow$ Choice of measure doesn’t matter too much in practice, but ideally use all
Evaluating clusterings
Just some pointers

Without labels
- **Silhouette coefficient**: Measure for cluster validity/consistency
- **Visual inspection of dendrograms**: Shows distances and balancing in hierarchical clusterings
- **Visual comparison** of dimension reduction on feature vectors (e.g., t-SNE, SOM)

With ground truth available
- **Purity**: a simple & transparent measure of correctness of clustering
- **Rand index**: compares two clusterings (e.g., own scheme with random partition)
- **Missclassification rate**: 
  \[ MR = \frac{1}{N} \sum_{j=1}^{C} e_j \]
  (\(N\) number of samples, \(C\) number of true clusters, \(e_j\) number of wrongly assigned samples of true cluster \(j\), i.e. spread to multiple clusters or mixed in not pure clusters)
- **Recall/precision etc.** also apply
3. SELECTING AMONG COMPETING MODELS

What basis is available to favor one algorithm over another?
How probable is it that the chosen method is truly significantly better?
Is there a best algorithm?
Theory & practice agree

Recap: No free lunch
• The no free lunch theorem (NFL) tells there’s no universally best learner (across problems)

Empirical study [Caruana et al., 2006]
• Confirmation of NFL: «Even the best models sometimes perform poorly, and models with poor average performance occasionally perform exceptionally well»
• Mild take home message: Ensembles and SVMs are good out of the box methods
• But: Naïve Bayes is great in SPAM filtering; boosted decision stumps are ultimate in face detection etc.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>1ST</th>
<th>2ND</th>
<th>3RD</th>
<th>4TH</th>
<th>5TH</th>
<th>6TH</th>
<th>7TH</th>
<th>8TH</th>
<th>9TH</th>
<th>10TH</th>
</tr>
</thead>
<tbody>
<tr>
<td>BST-DT</td>
<td>0.580</td>
<td>0.228</td>
<td>0.160</td>
<td>0.023</td>
<td>0.009</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>RF</td>
<td>0.390</td>
<td>0.525</td>
<td>0.084</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>BAG-DT</td>
<td>0.030</td>
<td>0.232</td>
<td>0.571</td>
<td>0.150</td>
<td>0.017</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>SVM</td>
<td>0.000</td>
<td>0.008</td>
<td>0.148</td>
<td>0.574</td>
<td>0.240</td>
<td>0.029</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>ANN</td>
<td>0.000</td>
<td>0.007</td>
<td>0.035</td>
<td>0.230</td>
<td>0.606</td>
<td>0.122</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>KNN</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.009</td>
<td>0.114</td>
<td>0.502</td>
<td>0.245</td>
<td>0.038</td>
<td>0.002</td>
<td>0.000</td>
</tr>
<tr>
<td>BST-STMP</td>
<td>0.000</td>
<td>0.000</td>
<td>0.002</td>
<td>0.013</td>
<td>0.014</td>
<td>0.257</td>
<td>0.710</td>
<td>0.004</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>DT</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.004</td>
<td>0.616</td>
<td>0.291</td>
<td>0.089</td>
</tr>
<tr>
<td>LOGREG</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.040</td>
<td>0.312</td>
<td>0.423</td>
<td>0.225</td>
</tr>
<tr>
<td>NB</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.030</td>
<td>0.284</td>
<td>0.686</td>
</tr>
</tbody>
</table>

Overall rank by mean performance across 11 learning tasks and 8 metrics; from [Caruana & Niculescu-Mizil, ICML 2006]. Classifiers have been calibrated to emit class probabilities; data sets span a wide range from nominal attributes to pattern recognition.
**Maximum likelihood (ML) model comparison**

Simplistic model selection

Often, parameters of some $h \in \mathcal{H}$ are estimated via ML (using CV)
- Find parameters $\hat{\theta}$ such that the likelihood $p \left( X | h(X, \hat{\theta}) \right)$ of the training data $X$ is maximized

**Maximum a posteriori (MAP) hypothesis via Bayes’ theorem**

- $p(h|X) = \frac{p(X|h) \cdot p(h)}{p(X)}$, where
  - $p(X|h)$ is the likelihood of the data, given the model \(\rightarrow\) called the evidence for $h$
  - $p(X)$ is the a priori likelihood of the training data $X$ \(\rightarrow\) this normalization factor is rarely needed/used
  - $p(h)$ is the a priori likelihood of the hypothesis $h$ \(\rightarrow\) often neglected in practice due to dominance of evidence
  \(\Rightarrow\) $p(h|X) \approx p(X|h)$

\(\Rightarrow\) Given competing $h_i \in \mathcal{H}_j$, one can
- **Find ML parameter** estimates
- **Calculate** the likelihood $p(X|h_i)$
- **Select best** $\hat{h} = \max_{h_i} p(X|h_i)$
Model selection criteria (→ see more in appendix)
Guided by Ockham’s Razor

Goal: **Compromise** between model **complexity** and **accuracy** on **validation** data

Idea: A **good** model is a **simple** model that achieves high accuracy on the given data

Philosophical backup

- **Ockham’s Razor** (axiom in ML!): «Given 2 models with the same **empirical** (test) error, the simpler [i.e., more comprehensible] one should be preferred because **simplicity is desirable in itself»
- Albert Einstein: «Make things as simple as possible – but not simpler»
- Vladimir Vapnik: «Don’t solve a more complex problem than necessary»
- Reasoning: For a simple hypothesis the **probability** of it **having unnecessary conditions** is **reduced**

History

- William of Ockham, born in the village of Ockham in Surrey (England) about 1285, was the most influential philosopher of the 14th century and a controversial theologian
- The original sentence «**Entities should not be multiplied beyond necessity**» was a critique of scholastic philosophy
- For a **comprehensive treatment** of Ockham’s razor for Machine Learning, see Pedro Domingos’ 1998 paper on «**Occam’s Two Razors - The Sharp and the Blunt**»

≠ often used version «Given 2 models with the same **training** error, the simpler should be preferred because it is **likely to have lower generalization error**» is **not generally true** → see [Domingos, 1998]
Is the best classifier really better than others? Using hypothesis tests to show significant predominance

In practice
- **10-fold CV often enough** (we don’t care if the best method isn’t really better)

Comparing two learners: is $\mathcal{L}_A$ on average better than $\mathcal{L}_B$?
- **Student’s t-test** tells whether the means of two samples differ significantly
- Our samples are…
  - CV: error estimates on $m$ different independently drawn data sets per learner
    (➔ **paired t-test**: tight bounds, but assumes vast data supply for the $m$ sets)
  - Bootstrapping: $m$ different error estimates run on (re-samplings of) the same data
    (➔ **corrected resampled t-test**: corrects for spurious differences when reusing the data $m$-fold)

Generally
- Difference of means $\mu_{\mathcal{L}_A}$ and $\mu_{\mathcal{L}_B}$ of the obtained cross validation error estimates follows a Student’s distribution with $m - 1$ degrees of freedom
  ➔ see appendix and [Wasserman, “All of statistics”, 2004, ch. 10]

History: William Gosset (wrote under the name "Student") invented the t-test to handle small samples for quality control in brewing.
Other things to consider for choosing models
…regarding data set composition

Is the number of features \( p \) large in comparison to the number of instances \( N \)?
• Also called \( p \gg N \)
  \( \Rightarrow \) Select appropriate methods, e.g. boosting or SVM (\( \rightarrow \) see V05/V04)

Is the data set severely imbalanced (i.e., probability of classes highly non-uniform)?
• E.g. for SPAM filtering in Email, anomaly-detection in sensor signals
  \( \Rightarrow \) Consider non-standard loss functions that take the class distribution into account
  \( \Rightarrow \) Consider Bayesian methods and appropriate prior probabilities (\( \rightarrow \) see appendix)
Review

How meaningful are the measured errors?
• Usually use 10-fold cross validation to estimate the test error
• Use the test error as an estimate of the true error
• Never let any optimization algorithm “see” any test / validation data (this sometimes comes in very subtly → see [ESL, ch. 7.10.2])

What to measure?
• Measure error / mean squared error for supervised learning
• Measure the Silhouette coefficient or Rand index for clusterings
• For a complete, cost-aware picture, make ROC curves for different settings

Which model to chose?
• The one with best CV score
• Use Student’s t-test to show that an improvement is significant (for publications)
• Ockham’s razor: prefer simpler models in absence of other evidence

⇒ Model selection is an empirical science.
P04.1: Analyzing bias and variance in (LOO)-CV

Work through P04.1:
• Follow the IPython notebook Analyzing_CV.ipynb
• Get used to IPython
  (For a quick intro to IPython → see appendix)
• Understand the bias-variance trade-off for evaluation methods
• Get to know the $k$-fold CV API of scikit-learn
APPENDIX

*Complementary material*

*Bayesian learning*

Quick introduction to *IPython*
COMPLEMENTARY MATERIAL
More on cross validation (CV)

Process
1. **Randomly split** (training) data into $k$ subsets of equal size
   (Probably do this with *stratification* to preserve class distribution in each fold:
   sample individually from each class, proportional to its share in the population)

2. For each fold $j$: use remaining $k - 1$ folds for training, compute error $E_{empj}$ on fold $j$

3. The final error is averaged: $E_{emp} = \frac{1}{k} \sum_{j=1}^{k} E_{empj}$

**Best practice**
- (stratified) **10-fold CV** (probably repeated 10 times and averaged)

**Alternatives**
- **Leave-one-out CV (LOOCV)**: uses $k = N$ (i.e., only one record for validation)
- **Bootstrap**: Draw $N$ random samples *with replacement* for training; use unused samples for validation
  - Training data will contain ca. 63.2% of the original training instances $\Rightarrow E_{validation}$ is very pessimistic
  - $E_{emp} = 0.632 \cdot E_{validation} + 0.368 \cdot E_{training}$, average several trials $\Rightarrow$ best for very small data sets

Yes, that’s the error on the training data
Confidence intervals for $E_{emp}$
What $E_{emp}$ tells about the true error

Foundation
- $E_{emp}$ is a random variable following a **Binomial distribution** (Bernoulli process)
- Can be **approximated with a Gaussian** distribution, given enough test data (i.e. $|S| \geq 100$)

Determining confidence intervals
- Approximately with probability $c$, $E_D(h) \approx E_{emp}(h) \pm z_c \cdot \sqrt{\frac{E_{emp}(h) \cdot (1 - E_{emp}(h))}{|S|}}$
- …where values for $z_c$ are given in the table below

<table>
<thead>
<tr>
<th>$c$</th>
<th>0.50</th>
<th>0.68</th>
<th>0.80</th>
<th>0.90</th>
<th>0.95</th>
<th>0.98</th>
<th>0.99</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z_c$</td>
<td>0.67</td>
<td>1.00</td>
<td>1.28</td>
<td>1.64</td>
<td>1.96</td>
<td>2.33</td>
<td>2.58</td>
</tr>
</tbody>
</table>

⇒ Larger test sets give **better** bounds
⇒ Anyway, bound has to be taken with a grain of salt

$(p \cdot 100\%)$ of area (probability) lies within $\mu \pm z_c \cdot \sigma$
(with $\mu$, $\sigma$ being the mean and standard deviation of the standard normal distribution)
Bias-variance trade-off: An example
Cross validation in the light of bias & variance

Advantages of $k$-fold CV over LOOCV
• Computational advantages: $k - N$ less training runs
• Usually also more accurate estimates of $E_{emp}$

Reason
• Bias
  • LOOCV introduces nearly no (selection) bias because the training set is as large as possible
  • $k$-fold CV introduces more bias, but still many times less then validation set approach
    ➞ Small advantage for LOOCV

• Variance
  • LOOCV averages $N$ highly correlated models (because training sets differ by only 1 sample)
  • $k$-fold CV averages models with less overlap in training data (training sets differ by $\frac{N}{k}$ samples)
  • Mean of highly correlated quantities has higher variance
    ➞ Comparatively bigger disadvantage for LOOCV

 ➞ There’s a bias-variance trade-off involved in the choice of $k$
 ➞ CV with $k = 5$ or $k = 10$ works best empirically
More on cost-sensitive training & prediction

Training
• Most learning schemes do not perform cost-sensitive learning
• Simple methods for cost-sensitive learning:
  • Resampling of instances according to costs
  • Weighting of instances according to costs, e.g. AdaBoost (→ see V05)
  • Some schemes can take costs into account by varying a parameter, e.g. naïve Bayes

Prediction
• Given: predicted class probabilities
• Basic idea: only predict high-cost class when very confident about prediction
• Make the prediction that minimizes the expected cost (instead predicting most likely class)
  • Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
  • Choose column (class) that minimizes expected cost

Practice
• Costs are rarely known ➔ decisions are usually made by comparing possible scenarios
• A lift chart allows a visual comparison (→ see next slide)
Lift charts
Comparing alternatives in marketing

Generating lift charts
• Sort instances according to predicted probability of being positive

Example
• Promotional mail to 1’000’000 households
• Mail to all; 0.1% respond (1’000)
• Improvement 1: subset of 100’000 most promising, 0.4% of these respond (400) (→ 40% of responses for 10% of cost)
• Improvement 2: subset of 400’000 most promising, 0.2% respond (800) (→ 80% of responses for 40% of cost)

→ Which is better? The lift chart allows a visual comparison

<table>
<thead>
<tr>
<th>Predicted probability</th>
<th>Actual class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.95</td>
</tr>
<tr>
<td>2</td>
<td>0.93</td>
</tr>
<tr>
<td>3</td>
<td>0.93</td>
</tr>
<tr>
<td>4</td>
<td>0.88</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• X axis is sample size, y axis is number of true positives
ROC curves and choice of classifier

Example
• For a small, focused sample, use method A
• For a larger one, use method B
• In between, choose between A and B with appropriate probabilities

The convex hull
• Given two learning schemes we can achieve any point on the convex hull!
• Example:
  • Let TP and FP rates for scheme A and B be $t_A$, $f_A$, $t_B$, $f_B$
  • If method A is used to predict $100 \cdot q\%$ of the cases and method B for the rest, then
    • $t_{A\cup B} = q \cdot t_A + (1 - q) \cdot t_B$
    • $f_{A\cup B} = q \cdot f_A + (1 - q) \cdot f_B$

→ See also: Scott et al., «Realisable Classifiers: Improving Operating Performance on Variable Cost Problems», 1998
Information criteria
Combining ML and complexity penalties

Classic information criteria
• Akaike information criterion (AIC)
  • Choose $\hat{h} = \max_{h_i} \ln p(X|h_i(X|\theta)) - \#\theta$, where $\#\theta$ is the number of tunable parameters in $h$
• Bayesian information criterion (BIC)
  • Choose $\hat{h} = \max_{h_i} \ln p(X|h_i(X|\theta)) - \frac{1}{2} \#\theta \ln N$

⇒ They do not take uncertainty in $\theta$ into account
⇒ Therefore usually prefer too simple models

Bayesian model selection
• Remember the MAP hypothesis $p(h|X) = \frac{p(X|h) \cdot p(h)}{p(X)}$ (⇒ see slide 16)
• The evidence for the model $h$ is correctly computed via the marginalized likelihood:
  $p(X|h) = \int p(X|\theta)p(\theta|h)d\theta$ (“summing” over all possible parameters)
⇒ The integrating out of the ML parameters compensates for model complexity (see figure)
⇒ Bayesian model selection performs the same as CV with less training runs

X-axis shows all possible data sets (ordered by complexity); y-axis shows likelihood of the data given a model. Broader applicable (i.e., more complex) models have lower likelihoods because of overall same probability mass. Here, model $M_2$ is optimal. 
From [Murphy, MLAPP, ch. 5.3.1]
The MDL principle
Minimum description length for model selection

Definition
• **Space** required to describe a **theory** + space required to describe the theory’s **mistakes**
• For classification: “Theory” is the classifier, “mistakes” are the errors on the validation data
  ➔ Goal: Classifier with minimal DL

Example: elegance vs. errors
• Theory 1: very simple & elegant ➔ explains the data almost perfectly
  E.g., Kepler’s three laws on planetary motion
• Theory 2: significantly more complex ➔ reproduces the data without mistakes
  E.g., Copernicus’s latest refinement of the Ptolemaic theory of epicycles
  ➔ Theory 1 is probably preferable (even though Copernicus’s theory is more accurate than Kepler’s on limited data)

MDL and data compression
• The **best theory** is the one that **compresses the data the most**
  (i.e., to compress a data set, generate & store (a) a model and (b) its mistakes)
  ➔ Computing size of error is easy (information loss)
  ➔ Computing size of model needs appropriate **encoding** method
MDL examples

MDL for clustering
• Computing the description length of the encoded clustering:
  • Model := bits needed to encode the cluster centers
  • Data := distance to cluster center (i.e., encode cluster membership and position relative to cluster)
→ Works if coding scheme uses less code space for small numbers than for large ones

MDL for binary classification
• Bits necessary for encoding the two «theories»:
  • A: 2 floats \((\theta_0, \theta_1)\) + relative errors
  • B: 11 floats + relative errors
MDL and MAP estimates
Maximum a posteriori (MAP) probabilities

Finding the MAP theory corresponds to finding the MDL theory
• **Difficulty** in applying **MAP** principle: determining the **prior probability** \( p(h) \) of the model
• **Difficulty** in applying **MDL** principle: finding a **coding scheme** for the model

\[ \text{if we know a priori that a particular model is more likely, we need fewer bits to encode it} \]

Disadvantages of MDL
• Appropriate coding schemes / prior probabilities for models are crucial
• No guarantee that the MDL model is the one which minimizes the expected error

\[ \text{Epicurus’s principle of multiple explanations: «keep all theories being consistent with the data»} \]
Performing the t-test usually on \( m \) repetitions of \( k \)-fold CV on same data

Test statistic \( t \)
- Corrected resampled (for practice): \( t = \frac{\mu_d}{\sqrt{(\frac{1}{mk} + \frac{k}{k^2 - k})\sigma^2}} \)
  where \( N \) is the number of instances in the training set, used \( m \) times with \( k \)-fold CV to produce \( mk \) error estimates per learner; the mean of their differences is \( \mu_d = \frac{1}{mk} \sum_{i=1}^{mk} E_{val}(h_{L_A}, X_i) - E_{val}(h_{L_B}, X_i) \); the variance of these differences is \( \sigma^2 \)
- Paired (for unlimited data): \( t = \frac{\mu_d}{\sigma \sqrt{mk}} \)

Process
- Compute the \( t \) statistic (w.r.t. the applicable version of the t-test)
- Fix a significance level \( \alpha \) (usually 0.01 or 0.05)
- Look up \( z \) corresponding to \( \frac{\alpha}{2} \) in the Student’s distribution table for \( km - 1 \) degrees of freedom
- Significant (with prob. \( 1 - \alpha \)) difference of the CV error estimates \( \iff t \leq -z \text{ or } t \geq z \)
BAYESIAN LEARNING
Bayesian reasoning & learning
based on [Mitchell, 1997], ch. 6

There’s a long-standing controversy pro/con Bayesianism in statistics
→ see e.g. http://lesswrong.com/lw/1to/what_is_bayesianism/

Bayesian reasoning
• Built upon Bayes’ theorem to convert prior probabilities into posteriors
• Quantities of interest are governed by probability distributions
• **Optimal decisions** are made by taking them plus observed data into account

Pro
• Provides explicit probabilities for hypotheses
• **Helps to understand**/analyze algorithms that don’t emit probabilities
  (e.g., why to minimize sum of squares; what the inductive bias of decision trees is)
• **Everything done probabilistically**
  (e.g., every training instance contributes to the final hypothesis according to its prior probability; **prior knowledge** can be incorporated as prior probabilities for candidate hypotheses or distributions over training data; predictions can be easily combined)

Con
• Many needed **probabilities** are **unknown** in practice (approximations like sampling needed)
• Direct application of Bayes theorem often **computationally intractable**
The Bayes optimal classifier
Classification’s «gold standard»

Theoretically **optimal** (=most probable) classification
- **Combine predictions** of all hypotheses, **weighted by** their **posterior** probabilities:

  \[
  \arg\max_{y_j \in Y} \sum_{h_i \in \mathcal{H}} p(y_j|h_i)p(h_i|X)
  \]

  (where \(y_j\) is a label from the set \(Y\) of classes, \(h_i\) is a specific hypothesis out of the hypothesis space \(\mathcal{H}\), and \(p(h_i|X)\) is the posterior of \(h_i\) given the data \(X\))

- No other method using the same \(\mathcal{H}\) and \(X\) can do better on average

**Pro**
- In particular **outperforms** simply taking the classification of the **MAP hypothesis**

  Example: Let 3 classifiers predict tomorrows weather as \(h_1(x) = \text{sunny}\), \(h_2(x) = \text{rainy}\), \(h_3(x) = \text{rainy}\) with posterior probabilities of .5, .4 and .1, respectively; let the true weather tomorrow be \(\text{rainy}\). The MAP hypothesis \(h_1\) wrongly predicts \(\text{sunny}\) weather; the Bayes classifier truly predicts \(\text{rainy}\).

- Enforces the idea of **ensemble learning** (→ see V05)

**Con**
- Computationally **intractable** (linear in \(|\mathcal{H}|\) → see [http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf](http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf))
Other forms of Bayesian learning

The Naïve Bayes classifier

Basic idea
- The straightforward way of applying Bayes’ theorem to yield a MAP hypothesis is intractable (too many conditional probability terms need to be estimated)
- **Simplification**: Assume **conditional independence** among features given target value

\[
h(x_i) = \arg \max_{y_j \in Y} P(y_j| x_{i1}, x_{i2}, ..., x_{iD}) = \arg \max_{y_j \in Y} P(x_{i1}, x_{i2}, ..., x_{iD}|y_j) \cdot P(y_j) = \arg \max_{y_j \in Y} P(y_j) \cdot \prod_{d=1..D} P(x_{id}|y_j)
\]

\[\rightarrow\] Very successful in text classification (e.g., SPAM filtering, news classification)

Example (from [https://alexn.org/blog/2012/02/09/howto-build-naive-bayes-classifier.html](https://alexn.org/blog/2012/02/09/howto-build-naive-bayes-classifier.html))
- Imagine 74 emails: 30 are SPAM; 51 contain “penis” (of which 20 are SPAM); 25 contain “Viagra” (24 are SPAM)
- Bayes classifier:

\[
p(\text{SPAM}|\text{penis}, \text{viagra}) = \frac{p(\text{penis}\mid \text{SPAM}\cap \text{viagra}) \cdot p(\text{viagra}\mid \text{SPAM}) \cdot p(\text{SPAM})}{p(\text{penis}\mid \text{viagra}) \cdot p(\text{viagra})} = \cdots
\]

\[\rightarrow\] intractable with more words because of **cond. prob. terms** also get numerically small

- Naïve Bayes classifier:

\[
p(\text{SPAM}|\text{penis}, \text{viagra}) = \frac{p(\text{penis}\mid \text{SPAM}) \cdot p(\text{viagra}\mid \text{SPAM}) \cdot p(\text{SPAM})}{p(\text{penis}) \cdot p(\text{viagra})} = \frac{20 \cdot 24.30 \cdot 30}{30 \cdot 74 \cdot 51.25 \cdot 74} = 9.928
\]
Other forms of Bayesian learning

The Bayes net (or Bayesian belief network)

In a nutshell
- **Loosens naïve Bayes constraint**: Assumes only conditional independence among certain sets of features
- Model of joint probability distribution of features (also unobserved ones):
  - a **directed acyclic graph** for independence assumptions and local conditional probabilities
- Inference possible for any feature / target, based on any set of observed variables
  - has to be done approximately to be tractable (NP-hard)
- **Use case**: conveniently **encode prior causal knowledge** in form of conditional (in)dependencies

- A simple Bayes net for medical diagnosis
- One node per random variable
  - Attached is a conditional probability table with the distribution of that node’s values given its parents
- A Link between 2 nodes exists if there is a direct conditional (causal) dependence
The EM algorithm
A general-purpose, *unsupervised* learning algorithm

**EM** (expectation maximization)
- Iterative method to **learn in the presence of unobserved variables**
  - A typical hidden variable is some sort of group/cluster membership
- Good convergence guarantees (finds local maximum)

**Example**
- A given dataset is known to be generated by either of 2 Gaussians (with equal probability)
- Only the data is observed
  - Which Gaussian generated a certain point is unobserved
  - The Gaussians’ parameters are unknown
- The means & variances of these Gaussians shall be learned
  - Needs an estimation of the membership probability of each point to either Gaussian

---

EM algorithm used to iteratively optimize the parameters of 2 Gaussians (animated)
Source: [https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm](https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm)
The EM algorithm (contd.)

Algorithm

1. Start with a random initial hypothesis
   Example: Pretend to know the parameters $\mu, \sigma^2$ of the 2 Gaussians (e.g., pick random values)

2. **E-Step**: Estimate expected values of unobserved variables, assuming the current hypothesis holds
   Example: Compute probabilities $p_{ti}$ that feature vector $x_t$ was produced by Gaussian $i$
   $p_{ti} = p(G = i|x_t) = \frac{p(x_t|G=i)p(G=i)}{p(x_t)} \approx p(x_t|G = i) = g_i(x_t, \mu_i, \sigma_i)$ with $g_i$ being the Gaussian pdf and $G$ the unobserved random variable indicating membership to one of the Gaussians)

3. **M-Step**: Calculate new Maximum Likelihood (ML) estimate of hypothesis, assuming the expected values from (2) hold
   Example: Calculate the $\mu_i, \sigma_i^2$, given the currently assigned membership
   (i.e., using standard ML estimation: $\mu_i = \frac{1}{T} \sum_{t=1}^{T} p_{ti} \cdot x_t$, $\sigma_i^2 = \frac{1}{T} \sum_{t=1}^{T} p_{ti} \cdot (x_t - \mu_i)^2$

4. Repeat with step 2 until convergence
   Always replacing old estimates with new ones
More on Bayesian learning

- [Mitchell, 1997], ch. 6: Concise introduction to Bayesian learning
- [Murphy, 2012] and [Bishop, 2006]: Two text books embracing the Bayesian perspective
QUICK INTRODUCTION TO IPYTHON
A quick introduction to IPython
Web-based enhanced Python console for explorative analysis

Features
• Runs in the browser
• **Code and markup** (e.g., descriptions, explanations) in the same «file»
• Concept of «cells»
  • The code in a cell is run on demand («play» button on highlighted cell)
  • Results are directly rendered below (text output, plots, sound, videos, formulae, …)
  • Order of execution is top-down (self-defined functions are possible)

➡ Easy to follow (because of explanations), easy to manipulate
➡ Often starting point for autonomous scripts

**IP[y]:** **IPython**
Interactive Computing
How to run an IPython notebook from github?

1. View source
2. Right click → “save page as …” on your local computer
3. Launch an IPython window in your browser (and start a kernel on your local machine) from the Anaconda launcher
4. Browse to the location of the saved notebook on your computer to open it
5. There you are (→ see next slide)

This is just the IPython kernel running in the background. It performs the computation.
How to operate an IPython notebook?

Run the highlighted cell; results are computed live and directly displayed below.

Double click markdown to edit

Click an individual cell in order to highlight it

Click an individual cell in order to highlight it

Code cells share the Python scope with the previously run cells above (i.e., all names from there are also known here; names not introduced and run earlier are not known)