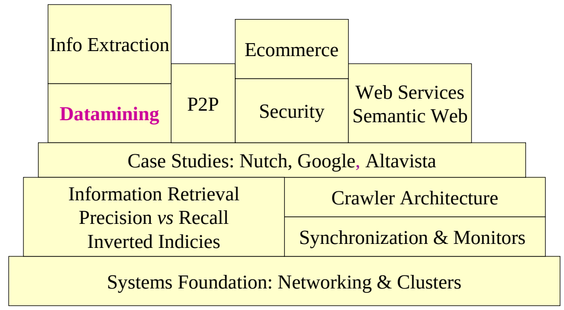
Text Categorization

CSE 454

Course Overview



Why is Learning Possible?

Experience alone never justifies any conclusion about any unseen instance.

Learning occurs when
PREJUDICE meets DATA!

Learning a "FOO"

Bias

- The nice word for prejudice is "bias".
- What kind of hypotheses will you consider?
  - What is allowable ***range*** of functions you use when approximating?
- What kind of hypotheses do you prefer?

Some Typical Bias: The World is Simple

- Occam's razor

  "It is needless to do more when less will suffice"

  – William of Occam,

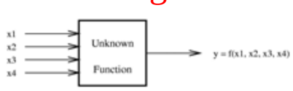
  died 1349 of the Black plague
- MDL – Minimum description length
- Concepts can be approximated by

  ... conjunctions of predicates

  ... by linear functions

  ... by short decision trees

A Learning Problem



Example	x_1	x_2	x_3	x_4	y
1	0	0	1	0	0
2	0	1	0	0	0
3	0	0	1	1	1
4	1	0	0	1	1
5	0	1	1	0	0
6	1	1	0	0	0
7	0	1	0	1	0

Hypothesis Spaces

- **Complete Ignorance.** There are $2^{16} = 65536$ possible boolean functions over four input features. We can't figure out which one is correct until we've seen every possible input-output pair. After 7 examples, we still have 2^9 possibilities.

x_1	x_2	x_3	x_4	y
0	0	0	0	?
0	0	0	1	?
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	?
1	0	0	0	?
1	0	0	1	1
1	0	1	0	?
1	0	1	1	?
1	1	0	0	0
1	1	0	1	?
1	1	1	0	?
1	1	1	1	?

Hypothesis Spaces (2)

- **Simple Rules.** There are only 16 simple conjunctive rules.

Rule	Counterexample
$\Rightarrow y$	1
$x_1 \Rightarrow y$	3
$x_2 \Rightarrow y$	2
$x_3 \Rightarrow y$	1
$x_4 \Rightarrow y$	7
$x_1 \wedge x_2 \Rightarrow y$	3
$x_1 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \Rightarrow y$	3
$x_2 \wedge x_4 \Rightarrow y$	3
$x_3 \wedge x_4 \Rightarrow y$	4
$x_1 \wedge x_2 \wedge x_3 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3
$x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Rightarrow y$	3

No simple rule explains the data. The same is true for simple clauses.

Terminology

- **Training example.** An example of the form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$.
- **Target function (target concept).** The true function f.
- **Hypothesis**. A proposed function h believed to be similar to f.
- **Concept**. A boolean function. Examples for which $f(\mathbf{x}) = 1$ are called **positive examples** or **positive instances** of the concept. Examples for which $f(\mathbf{x}) = 0$ are called **negative examples** or **negative instances.**
- **Classifier**. A discrete-valued function. The possible values $f(\mathbf{x}) \in \{1, \ldots, K\}$ are called the **classes** or **class labels**.
- **Hypothesis Space**. The space of all hypotheses that can, in principle, be output by a learning algorithm.
- **Version Space**. The space of all hypotheses in the hypothesis space that have not yet been ruled out by a training example.

Two Strategies for ML

- Restriction bias: use prior knowledge to specify a restricted hypothesis space.
  - Naïve Bayes
- Preference bias: use a broad hypothesis space, but impose an ordering on the hypotheses.
  - Decision Trees.

Key Issues for ML

- **What are good hypothesis spaces?**
  Which spaces have been useful in practical applications and why?
- **What algorithms can work with these spaces?**
  Are there general design principles for machine learning algorithms?
- **How can we optimize accuracy on future data points?**
  This is sometimes called the "problem of overfitting".
- **How can we have confidence in the results?**
  How much training data is required to find accurate hypotheses? (the *statistical question*)
- **Are some learning problems computationally intractable?**
  (the *computational question*)
- **How can we formulate application problems as machine learning problems?** (the *engineering question*)

Framework for Learning Algos

- **Search Procedure**.
  Direction Computation: solve for the hypothesis directly.
  Local Search: start with an initial hypothesis, make small improvements until a local optimum.
  Constructive Search: start with an empty hypothesis, gradually add structure to it until local optimum.
- **Timing**.
  Eager: Analyze the training data and construct an explicit hypothesis.
  Lazy: Store the training data and wait until a test data point is presented, then construct an ad hoc hypothesis to classify that one data point.
- **Online vs. Batch**. (for eager algorithms)
  Online: Analyze each training example as it is presented.
  Batch: Collect training examples, analyze them, output an hypothesis.

Categorization (review)

- Given:
  - A description of an instance, $x \in X$, where X is the *instance language* or *instance space*.
  - A fixed set of categories: $C=\{c_1, c_2, \ldots c_n\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a categorization function whose domain is X and whose range is C.

Learning for Categorization

- A training example is an instance $x \in X$, paired with its correct category $c(x)$: $<x, c(x)>$ for an unknown categorization function, c.
- Given a set of training examples, D.
- Find a hypothesized categorization function, $h(x)$, such that:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Consistency

Sample Category Learning Problem

- Instance language: <size, color, shape>
  - size $\in$ {small, medium, large}
  - color $\in$ {red, blue, green}
  - shape $\in$ {square, circle, triangle}
- C = {positive, negative}
- D:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

More to the Point

- C(X) = true if X is a Webcam page
- Features

  Words on page

  ….
- Hypothesis Language

Generalization

- Hypotheses must generalize to correctly classify instances ***not*** in the training data.
  - Simply memorizing training examples gives a consistent hypothesis that does not generalize.
- *Occam's razor*:
  - Finding a *simple* hypothesis helps ensure generalization.

Text Categorization

- Assigning documents to a fixed set of categories.
- Applications:
  - Web pages
    - Categories in search (see microsoft.com)
    - Yahoo-like classification
  - Newsgroup Messages / News articles
    - Recommending
    - Personalized newspaper
  - Email messages
    - Routing
    - Prioritizing
    - Folderizing
    - spam filtering

General Learning Issues

- Many hypotheses often consistent w/ training data.
- Bias
  - Any criteria other than consistency with the training data that is used to select a hypothesis.
- Classification accuracy
  - % of instances classified correctly
  - Measured on independent test data.
- Training time
  - Efficiency of training algorithm
- Testing time
  - Efficiency of subsequent classification

Learning for Text Categorization

- Manual development of text categorization functions is difficult.
- Learning Algorithms:
  - **Bayesian (naïve)**
  - Neural network
  - **Relevance Feedback (Rocchio)**
  - Rule based (C4.5, Ripper, Slipper)
  - **Nearest Neighbor (case based)**
  - Support Vector Machines (SVM)

Using Relevance Feedback (Rocchio)

- Adapt relevance feedback for text categorization.
- Use standard TF/IDF weighted vectors to represent text documents (normalized by maximum term frequency).
- For each category, compute a *prototype* vector by summing the vectors of the training documents in the category.
- Assign test documents to the category with the closest prototype vector based on cosine similarity.

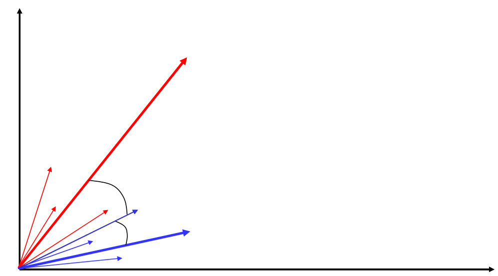
Rocchio Text Categorization Algorithm (Training)

Assume the set of categories is $\{c_1, c_2, \ldots c_n\}$
For i from 1 to n let $\mathbf{p}_i = \langle 0, 0, \ldots, 0 \rangle$ *(init. prototype vectors)*
For each training example $\langle x, c(x) \rangle \in D$
  Let **d** = frequency normalized TF/IDF term vector for doc x
  Let $i = j$: $(c_j = c(x))$
  (sum all the document vectors in c_i to get $\mathbf{p}_i$)
  Let $\mathbf{p}_i = \mathbf{p}_i + \mathbf{d}$

Rocchio Text Categorization Algo (Test)

Given test document x
Let **d** be the TF/IDF weighted term vector for x
Let $m = -2$ *(init. maximum cosSim)*
For i from 1 to n:
  (compute similarity to prototype vector)
  Let $s = \text{cosSim}(\mathbf{d}, \mathbf{p}_i)$
  if $s > m$
    let $m = s$
    let $r = c_i$ *(update most similar class prototype)*
Return class r

Illustration of Rocchio Text Categorization



Rocchio Properties

- Does not guarantee a consistent hypothesis.
- Forms a simple generalization of the examples in each class (a *prototype*).
- Prototype vector does not need to be averaged or otherwise normalized for length since cosine similarity is insensitive to vector length.
- Classification is based on similarity to class prototypes.

25

Rocchio Time Complexity

- Note: The time to add two sparse vectors is proportional to minimum number of non-zero entries in the two vectors.
- Training Time: $O(|D|(L_d + |V_d|)) = O(|D|\, L_d)$
  where L_d is the average length of a document in D and V_d is the average vocabulary size for a document in D.
- Test Time: $O(L_t + |C||V_t|)$
  where L_t is the average length of a test document and $|V_t|$ is the average vocabulary size for a test document.
  - Assumes lengths of $\mathbf{p}_i$ vectors are computed and stored during training, allowing cosSim($\mathbf{d}$, $\mathbf{p}_i$) to be computed in time proportional to the number of non-zero entries in $\mathbf{d}$ (i.e. $|V_t|$)

26

Nearest-Neighbor Learning Algorithm

- Learning is just storing the representations of the training examples in *D*.
- Testing instance *x*:
  - Compute similarity between *x* and all examples in *D*.
  - Assign *x* the category of the most similar example in *D*.
- Does not explicitly compute a generalization or category prototypes.
- Also called:
  - Case-based
  - Memory-based
  - Lazy learning

27

K Nearest-Neighbor

- Using only the closest example to determine categorization is subject to errors due to:
  - A single atypical example.
  - Noise (i.e. error) in the category label of a single training example.
- More robust alternative is to find the *k* most-similar examples and return the majority category of these *k* examples.
- Value of *k* is typically odd to avoid ties, 3 and 5 are most common.

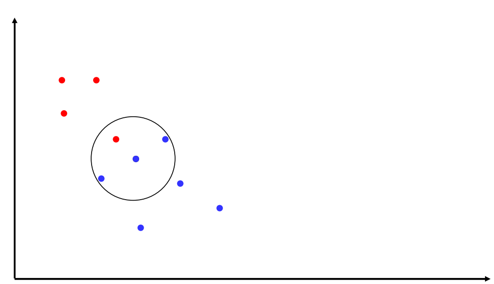
28

Similarity Metrics

- Nearest neighbor method depends on a similarity (or distance) metric.
- Simplest for continuous *m*-dimensional instance space is *Euclidian distance*.
- Simplest for *m*-dimensional binary instance space is *Hamming distance* (number of feature values that differ).
- For text, cosine similarity of TF-IDF weighted vectors is typically most effective.

29

3 Nearest Neighbor Illustration (Euclidian Distance)



30

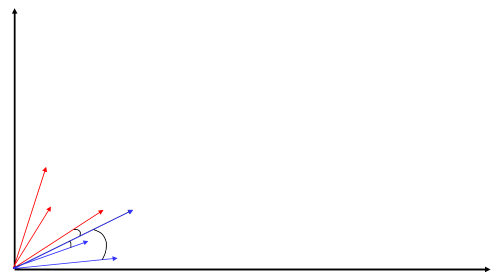
K Nearest Neighbor for Text

Training:
For each each training example <x, c(x)> ∈ D
    Compute the corresponding TF-IDF vector, $\mathbf{d}_x$, for document *x*

Test instance *y*:
Compute TF-IDF vector **d** for document *y*
For each <x, c(x)> ∈ D
    Let s_x = cosSim(**d**, $\mathbf{d}_x$)
Sort examples, *x*, in *D* by decreasing value of s_x
Let *N* be the first *k* examples in D.   *(get most similar neighbors)*
Return the majority class of examples in *N*

31

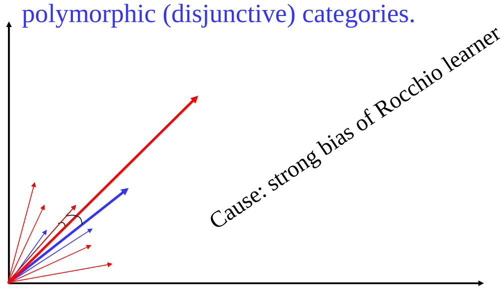
Illustration of 3 Nearest Neighbor for Text



32

Rocchio Anomaly

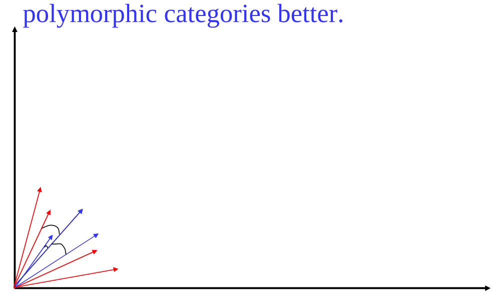
- Prototype models have problems with polymorphic (disjunctive) categories.



33

3 Nearest Neighbor Comparison

- Nearest Neighbor tends to handle polymorphic categories better.



34

Nearest Neighbor Time Complexity

- Training Time: O(|*D*| L_d) to compose TF-IDF vectors.
- Testing Time: O(L_t + |*D*||V_t|) to compare to all training vectors.
  - Assumes lengths of $\mathbf{d}_x$ vectors are computed and stored during training, allowing cosSim(**d**, $\mathbf{d}_x$) to be computed in time proportional to the number of non-zero entries in **d** (i.e. |V_t|)
- Testing time can be high for large training sets.

35

Nearest Neighbor with Inverted Index

- Determining *k* nearest neighbors is the same as determining the *k* best retrievals using the test document as a query to a database of training documents.
- Use standard VSR inverted index methods to find the *k* nearest neighbors.
- Testing Time: O(*B*|V_t|)
  where *B* is the average number of training documents in which a test-document word appears.
- Therefore, overall classification is O(L_t + *B*|V_t|)
  - Typically *B* << |*D*|

36

Bayesian Methods

- Learning and classification methods based on probability theory.
  - Bayes theorem plays a critical role in probabilistic learning and classification.
  - Uses ***prior*** probability of each category given no information about an item.
- Categorization produces a ***posterior*** probability distribution over the possible categories given a description of an item.

37

Axioms of Probability Theory

- All probabilities between 0 and 1

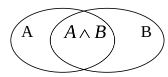
$$0 \le P(A) \le 1$$

- True proposition has probability 1, False has probability 0.

P(true) = 1 P(false) = 0.

- The probability of disjunction is:

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

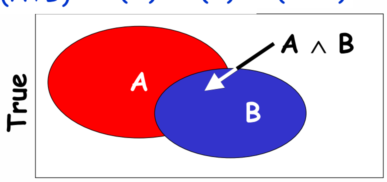


38

Probability: Simple & logical

The definitions imply that certain logically related events must have related probabilities

E.g. $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$



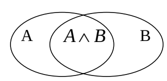
de Finetti (1931): an agent who bets according to probabilities that violate these axioms can be forced to bet so as to lose money regardless of outcome.

39

Conditional Probability

- $P(A \mid B)$ is the probability of A given B
- Assumes that B is all and only information known.
- Defined by:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$



40

Independence

- A and B are ***independent*** iff:

$$P(A \mid B) = P(A)$$

$$P(B \mid A) = P(B)$$

These two constraints are logically equivalent

- Therefore, if A and B are independent:

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)} = P(A)$$

$$P(A \wedge B) = P(A)P(B)$$

41

Bayes Theorem

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$

Simple proof from definition of conditional probability:

$$P(H \mid E) = \frac{P(H \wedge E)}{P(E)}$$ (Def. cond. prob.)

$$P(E \mid H) = \frac{P(H \wedge E)}{P(H)}$$ (Def. cond. prob.)

$$P(H \wedge E) = P(E \mid H)P(H)$$ (Mult both sides of 2 by P(H).)

QED: $$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)}$$ (Replace 3 in 1.)

42

Bayesian Categorization

- Let set of categories be $\{c_1, c_2, \ldots c_n\}$
- Let E be description of an instance.
- Determine category of E by determining for each c_i

$$P(c_i \mid E) = \frac{P(c_i)P(E \mid c_i)}{P(E)}$$

- $P(E)$ can be determined since categories are complete and disjoint.

$$\sum_{i=1}^{n} P(c_i \mid E) = \sum_{i=1}^{n} \frac{P(c_i)P(E \mid c_i)}{P(E)} = 1$$

$$P(E) = \sum_{i=1}^{n} P(c_i)P(E \mid c_i)$$

43

Bayesian Categorization (cont.)

- Need to know:
  - Priors: $P(c_i)$
  - Conditionals: $P(E \mid c_i)$
- $P(c_i)$ are easily estimated from data.
  - If n_i of the examples in D are in c_i, then $P(c_i) = n_i / |D|$
- Assume instance is a conjunction of binary features:

$$E = e_1 \wedge e_2 \wedge \cdots \wedge e_m$$

- Too many possible instances (exponential in m) to estimate all $P(E \mid c_i)$

44

Naïve Bayesian Categorization

- If we assume features of an instance are independent given the category (c_i) (*conditionally independent*).

$$P(E \mid c_i) = P(e_1 \wedge e_2 \wedge \cdots \wedge e_m \mid c_i) = \prod_{j=1}^{m} P(e_j \mid c_i)$$

- Therefore, we then only need to know $P(e_j \mid c_i)$ for each feature and category.

45

Naïve Bayes Example

- C = {allergy, cold, well}
- e_1 = sneeze; e_2 = cough; e_3 = fever
- E = {sneeze, cough, ¬fever}

Prob	Well	Cold	Allergy
$P(c_i)$	0.9	0.05	0.05
$P(\text{sneeze} \mid c_i)$	0.1	0.9	0.9
$P(\text{cough} \mid c_i)$	0.1	0.8	0.7
$P(\text{fever} \mid c_i)$	0.01	0.7	0.4

46

Naïve Bayes Example (cont.)

Probability	Well	Cold	Allergy
$P(c_i)$	0.9	0.05	0.05
$P(\text{sneeze} \mid c_i)$	0.1	0.9	0.9
$P(\text{cough} \mid c_i)$	0.1	0.8	0.7
$P(\text{fever} \mid c_i)$	0.01	0.7	0.4

E={sneeze, cough, ¬fever}

P(well | E) = (0.9)(0.1)(0.1)(0.99)/P(E)=0.0089/P(E)
P(cold | E) = (0.05)(0.9)(0.8)(0.3)/P(E)=0.01/P(E)
P(allergy | E) = (0.05)(0.9)(0.7)(0.6)/P(E)=0.019/P(E)

Most probable category: allergy
P(E) = 0.089 + 0.01 + 0.019 = 0.0379
P(well | E) = 0.23
P(cold | E) = 0.26
P(allergy | E) = 0.50

47

Estimating Probabilities

- Normally, probabilities are estimated based on observed frequencies in the training data.
- If D contains n_i examples in category c_i, and n_{ij} of these n_i examples contains feature e_j, then:

$$P(e_j \mid c_i) = \frac{n_{ij}}{n_i}$$

- However, estimating such probabilities from small training sets is error-prone.
- If due only to chance, a rare feature, e_k, is always false in the training data, $\forall c_i : P(e_k \mid c_i) = 0$.
- If e_k then occurs in a test example, E, the result is that $\forall c_i$: $P(E \mid c_i) = 0$ and $\forall c_i$: $P(c_i \mid E) = 0$

48

Smoothing

- To account for estimation from small samples, probability estimates are adjusted or *smoothed*.
- Laplace smoothing using an *m*-estimate assumes that each feature is given a prior probability, *p*, that is assumed to have been previously observed in a "virtual" sample of size *m*.

$$P(e_j \mid c_i) = \frac{n_{ij} + mp}{n_i + m}$$

- For binary features, *p* is simply assumed to be 0.5.

49

Naïve Bayes for Text

- Modeled as generating a bag of words for a document in a given category by repeatedly sampling with replacement from a vocabulary $V = \{w_1, w_2, \ldots w_m\}$ based on the probabilities $P(w_j \mid c_i)$.
- Smooth probability estimates with Laplace *m*-estimates assuming a uniform distribution over all words ($p = 1/|V|$) and $m = |V|$
  - Equivalent to a virtual sample of seeing each word in each category exactly once.

50

Text Naïve Bayes Algorithm (Train)

Let V be the vocabulary of all words in the documents in D
For each category $c_i \in C$
  Let D_i be the subset of documents in D in category c_i
  $P(c_i) = |D_i| / |D|$
  Let T_i be the concatenation of all the documents in D_i
  Let n_i be the total number of word occurrences in T_i
  For each word $w_j \in V$
    Let n_{ij} be the number of occurrences of w_j in T_i
    Let $P(w_i \mid c_i) = (n_{ij} + 1) / (n_i + |V|)$

51

Text Naïve Bayes Algorithm (Test)

Given a test document X
Let n be the number of word occurrences in X
Return the category:

$$\underset{c_i \in C}{\operatorname{argmax}}\, P(c_i) \prod_{i=1}^{n} P(a_i \mid c_i)$$

where a_i is the word occurring the ith position in X

52

Naïve Bayes Time Complexity

- Training Time: $O(|D|L_d + |C||V|))$ where L_d is the average length of a document in D.
  - Assumes V and all D_i, n_i, and n_{ij} pre-computed in $O(|D|L_d)$ time during one pass through all of the data.
  - Generally just $O(|D|L_d)$ since usually $|C||V| < |D|L_d$
- Test Time: $O(|C|\ L_t)$ where L_t is the average length of a test document.
- Very efficient overall, linearly proportional to the time needed to just read in all the data.
- Similar to Rocchio time complexity.

53

Underflow Prevention

- Multiplying lots of probabilities, which are between 0 and 1 by definition, can result in floating-point underflow.
- Since $\log(xy) = \log(x) + \log(y)$, it is better to perform all computations by summing logs of probabilities rather than multiplying probabilities.
- Class with highest final un-normalized log probability score is still the most probable.

54

Naïve Bayes Posterior Probabilities

- Classification results of naïve Bayes (the class with maximum posterior probability) are usually fairly accurate.
- However, due to the inadequacy of the conditional independence assumption, the actual posterior-probability numerical estimates are *not* accurate.
  – Output probabilities are generally very close to 0 or 1.

Evaluating Categorization

- Evaluation must be done on test data that are independent of the training data (usually a disjoint set of instances).
- *Classification accuracy*: c/n where n is the total number of test instances and c is the number of test instances correctly classified by the system.
- Results can vary based on sampling error due to different training and test sets.
- Average results over multiple training and test sets (splits of the overall data) for the best results.

N-Fold Cross-Validation

- Ideally, test and training sets are independent on each trial.
  – But this would require too much labeled data.
- Partition data into N equal-sized disjoint segments.
- Run N trials, each time using a different segment of the data for testing, and training on the remaining N–1 segments.
- This way, at least test-sets are independent.
- Report average classification accuracy over the N trials.
- Typically, N = 10.

Learning Curves

- In practice, labeled data is usually rare and expensive.
- Would like to know how performance varies with the number of training instances.
- *Learning curves* plot classification accuracy on independent test data (Y axis) versus number of training examples (X axis).

N-Fold Learning Curves

- Want learning curves averaged over multiple trials.
- Use N-fold cross validation to generate N full training and test sets.
- For each trial, train on increasing fractions of the training set, measuring accuracy on the test data for each point on the desired learning curve.

Sample Learning Curve (Yahoo Science Data)