Text Categorization

Foundations of Statistic Natural Language Processing

The MIT Press1999

Outline

- Introduction
- Decision Trees
- Maximum Entropy Modeling (optional)
- Perceptrons
- K Nearest Neighbor Classification

Introduction

- Categorization is the task of assigning objects from a universe to two or more classes or categories.

- The goal in text categorization is to classify the topic or theme of a document.

Problem	Object	Categorization
tagging	context of a word	the word's tags
disambiguation	context of a word	the word's senses
Author identification	document	authors
language identification	document	languages
text categorization	document	topics

Introduction

- A typical set of topic categories is the one used in the Reuters text collection.
- Some of its topics are “mergers and acquisition,” ”wheat,” “crude oil,” and “earnings and reports.”
- One application of text categorization is to filter a stream of news for a particular interest group.
- For example, a financial journalist may only want to see documents that have been assigned the category “mergers and acquisition.”

Introduction

- In general, the problem of statistical classification can be characterized as follow.

- Data representation model.

- We define model class.
  (a parameterized family of classifiers)

- We also define training procedure.
  (select one classifier from this family)

Data representation model

- Typically, each object in the training set is represented in the form $(\vec{x}, c)$,where $\vec{x} \in R^n$ is a vector of measurements and c is the class label.

- For text categorization, the information retrieval vector space model is frequently used as the data representation.

- That is, each document is represented as a vector of word counts.

Model class

- An example of such a family for binary classification is linear classifier which take the following form:

$$g(\vec{x}) = \vec{w} \cdot \vec{x} + w_0$$

where we choose class c_1 for $g(\vec{x}) > 0$ and class c_2 for $g(\vec{x}) \leq 0$.

This family is parameterized by the vector $\vec{w}$ and the threshold w_0.

Training procedure

- We can think of training procedures as algorithms for function fitting, which search for a good set of parameter values.
- Goodness is determined by an optimization criterion such as misclassification rate or entropy.
- Some training procedures are guaranteed to find he optimal set of parameters.
- However, many iterative training procedures are only guaranteed to find a better set in each iteration.

Evaluation

- Once we have chosen the parameters of the classifier (or, as we usually say, trained the classifier), it is a good idea to see how well it is doing on a test set.

- This training set should consist of data that was not used during training.

Evaluation

- For binary classification, an important measure is classification accuracy which is defined as $\frac{a+d}{a+b+c+d}$,the proportion of correctly classified objects.

Other measures are precision $\frac{a}{a+b}$, recall $\frac{a}{a+c}$, and fallout $\frac{b}{b+d}$

	YES is correct	NO is correct
YES was assigned	a	*b*
NO was assigned	c	***d***

Table 16.2 Contingency table for evaluating a binary classifier. For example, a is the number of objects in the category of interest that were correctly assigned to the category.

Evaluation

- In classification tasks with more than two categories, one begins by making a 2×2 contingency table for each category c_i separately.
- Macro-averaging: compute an evaluation measure like accuracy for each contingency table separately and then average the evaluation measure over categories to get an overall measure of performance.
- Micro-averaging: one first makes a single contingency table for all the data by summing the scores in each cell for all categories. The evaluation measure is then computed for this large table.

Evaluation

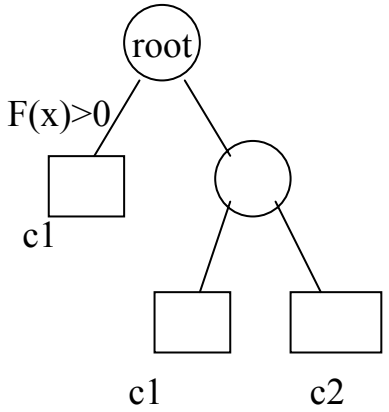
- Micro-averaged precision is dominated by the large categories.
- Macro-averaged precision will give a better sense of the quality of classification across all categories.

Decision tree

- A decision tree is a hierarchical model for supervised learning whereby the local region is identified in a sequence of recursive splits in a smaller number of steps.

- A decision tree is composed of internal decision nodes and terminal leaves.

- Each decision node m implements a test function $f_m(x)$ with discrete outcomes labeling the branches.



$F(\vec{x}) = \vec{w} \cdot \vec{x} + w_0$
where we choose class c_1 for $F(\vec{x}) > 0$ and class c_2 for $F(\vec{x}) \leq 0$.
This family is parameterized by the vector $\vec{w}$ and the threshold w_0.

Decision tree - dataset

- The text categorization task that we use as an example in this chapter is to build categorizers that distinguish the “earnings” category in the *Reuters* collection.

- The version we use consists of 9603 training articles and 3299 test articles that were sent over the *Reuters* newswire in 1987.

- The articles are categorized with more than 100 topics.

Decision tree – data representation

- The first task in text categorization is to find an appropriate data representation model.
- For simplicity, we will use a single data representation throughout this chapter.
- It is based on the 20 words whose χ^2 score with the category "earnings" in the training set was highest.

hypothesis	in earnings	not in earnings
x		
-x		

Decision tree

```
<REUTERS NEWID="11">
<DATE>26-FEB-1987 15:18:59.34</DATE>
<TOPICS><D>earn</D></TOPICS>
<TEXT>
<TITLE>COBANCO INC &1t;CBCO> YEAR NET</TITLE>
<DATELINE>    SANTA CRUZ, Calif., Feb 26 - </DATELINE>
<BODY>Shr 34 cts vs 1.19 dlrs
    Net 807,000 vs 2,858,000
    Assets 510.2 mln vs 479.7 mln
    Deposits 472.3 mln vs 440.3 mln
    Loans 299.2 mln vs 327.2 mln
    Note: 4th qtr not available. Year includes 1985
extraordinary gain from tax carry forward of 132,000 dlrs,
or five cts per shr.
 Reuter
</BODY></TEXT>
</REUTERS>
```

Decision tree – data representation

Word w^j	Term weight s_{ij}	Classification
vs	5	
mln	5	
cts	3	
	3	
6	3	
000	4	
loss	0	
	0	
"	0	
3	4	
profit	0	
dlrs	3	
1	2	
pct	0	
is	0	
s	0	
that	0	
net	3	
lt	2	
at	0	

$\vec{x} =$ (the vector of term weights above) $\quad c = 1$

Table 16.3 The representation of document **11, shown** in figure 16.3. This illustrates the data representation model which we use for classification in this chapter.

Decision tree – data representation

- Each document was then represented as a vector of K=20,

  $\vec{x}_j = (s_{1j},...,s_{Kj})$, where s_{ij} was computed as the following quantity :

  $$s_{ij} = \text{round}(10 \times \frac{1+\log(tf_{ij})}{1+\log(l_j)})$$

  tf_{ij} is the number of occurrences of term i in document j.
  l_j is the length of document j.

- We round values to make it easier to present and inspect data for pedagogical reasons.

$$S_{vs,11} = round\left(10 \times \frac{1+\log(6)}{1+\log(89)}\right) \approx round(5.09) = 5$$

Decision tree

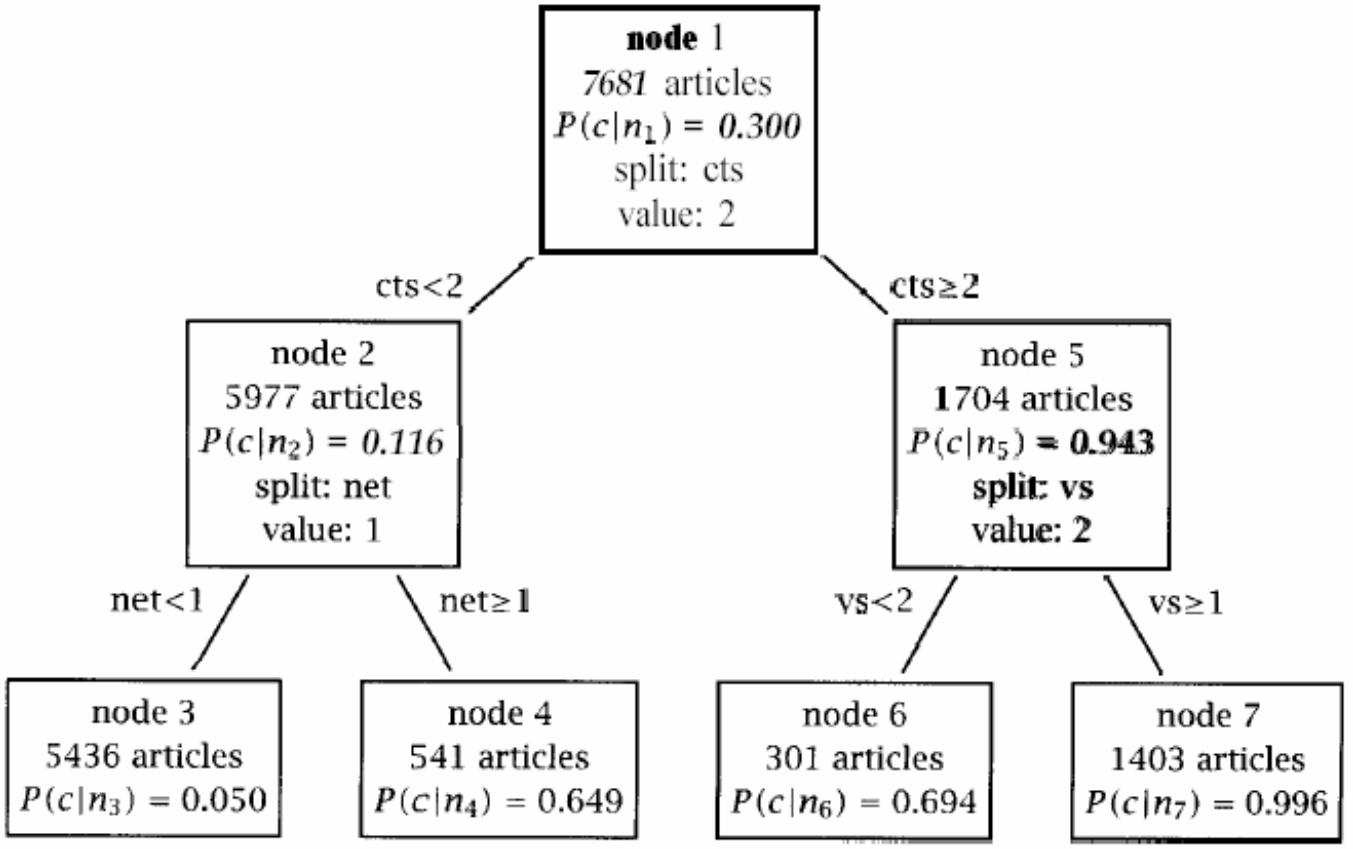


Figure 16.1 A decision tree. This tree determines whether a document is part of the topic category ••arnings•• or not. $P(c|n_i)$ is the probability of a document at node n_i to belong to the ••arnings•• category c.

Decision tree

- Another way to visualize the tree is shown in figure 16.2.
- The horizontal axis corresponds to the weight for cts, the vertical axis to the weight for net.
- Questions ask whether the value of some feature is less than some value or not.

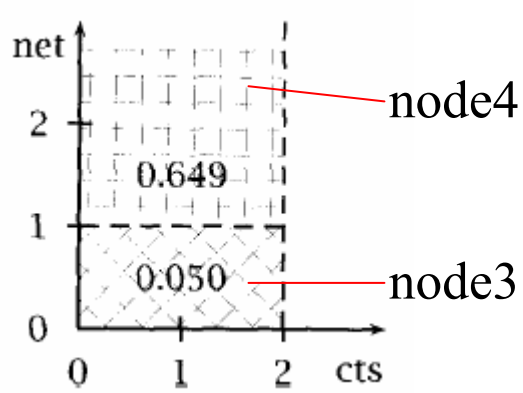


Figure16.2 Geometric interpretation of part of the tree in figure 16.1

Decision tree

- Now that we have a model class (decision trees) and a representation for the data (20-element vectors), we need to define the training procedure.

- Decision trees are usually built by first growing a large tree and then pruning it back to a reasonable size. The pruning step is necessary because very large trees overfit the training set.

- Overfitting occurs when classifiers make decisions based on accidental properties of the training set that will lead to errors on the test set.

Decision tree

- For growing the tree, we need a splitting criterion for finding the feature and its value that we will split on and stopping criterion which determines when to stop splitting.

- The stopping criterion can trivially be that all element at a node have an identical representation or the same category so that splitting would not further distinguish them.

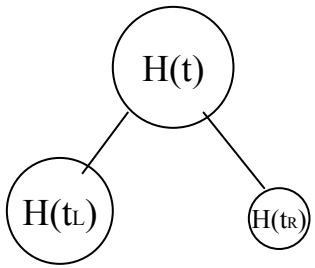
- The splitting criterion which we will use here is to split the objects at a node into two piles in the way that gives us maximum information gain.

Information gain

- Information gain is an information-theoretic measure defined as difference of the entropy of the mother node and the weighted sum of the entropies of the child nodes:

$$G(a, y) = H(t) - H(t \mid a) = H(t) - (p_L H(t_L) + p_R H(t_R))$$

where a is the attribute we split on, y is the value of a we split on, t is the distribution of the node that we split,
p_L and p_R are the proportion of elements that are passed on to the left and right nodes.
t_L and t_R are the distributions of the left and right nodes.



Information gain is intuitively appealing because it can be interpreted as measuring the reduction of uncertainty.

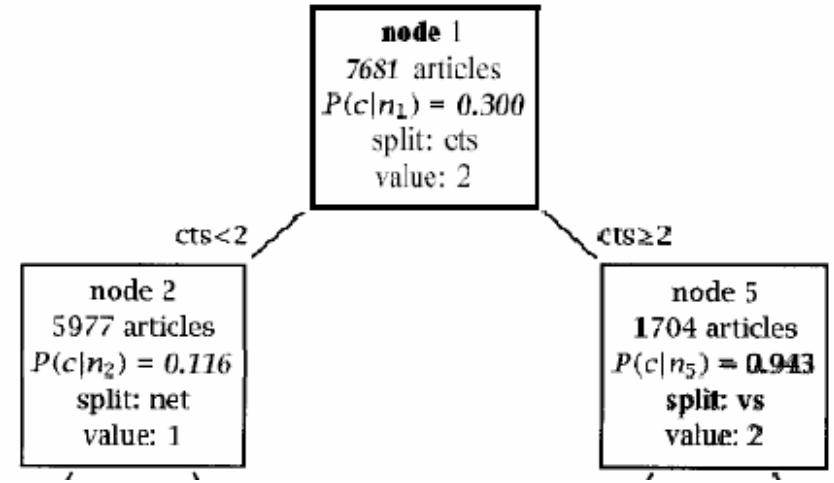
Information gain

$$H(x) = -\sum_{x \in X} p(x) \log p(x)$$

$$-(0.3 \times \log 0.3 + 0.7 \times \log 0.7) = 0.611$$

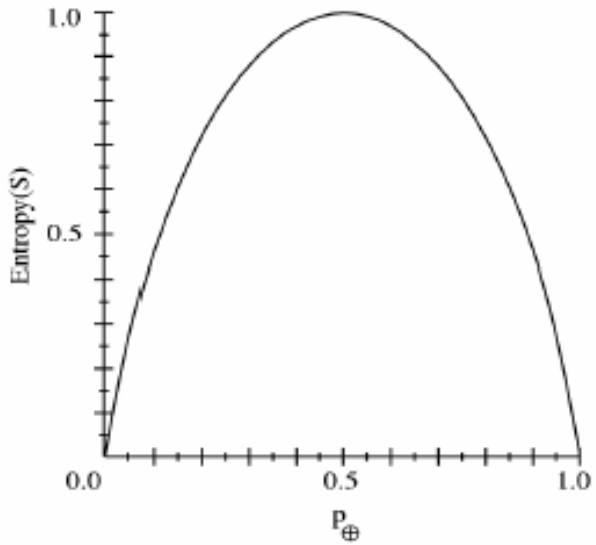
entropy at node 1, $P(C\|N) = 0.300$	0.611
entropy at node 2, $P(C\|N) = 0.116$	0.359
entropy at node 5, $P(C\|N) = 0.943$	0.219
weighted sum of 2 and 5	$\frac{5977}{7681} \times 0.359 + \frac{1704}{7681} \times 0.219 = 0.328$
information gain	$0.611 - 0.328 = 0.283$

Table 16.4 An example of information gain as a splitting criterion. The table shows the entropies for nodes 1, 2, and 5 in figure 16.1, the weighted sum of the child nodes and the information gain for splitting 1 into 2 and 5.



Review Entropy

- For Boolean classification (0 or 1)



$$Entropy\,(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2$$

- Entropy can be expressed as the minimum number of bits of information needed to encode the classification of an arbitrary number of examples
  - If c classes are generated, the maximum of Entropy can be

$$Entropy\,(X) = \log_2 c$$

ID3 algorithm

GenerateTree(X)

If NodeEntropy(X) $< \theta_I$ // $I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$ Stop criterion

Create leaf labelled by majority class in X

Return

$i \leftarrow$ SplitAttribute(X)

For each branch of x_i

Find X_i falling in branch

GenerateTree(X_i)

} Depth first search

SplitAttribute(X)

MinEnt $\leftarrow$ MAX

For all attributes $i = 1,\ldots,d$

If x_i is discrete with n values

Split X into $X_1,...,X_n$ by x_i

$e \leftarrow$ SplitEntropy($X_1,...,X_n$) // $I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^i \log p_{mj}^i$

If $e <$ MinEnt MinEnt $\leftarrow e$; bestf $\leftarrow i$

Else // x_i is numeric

For all possible splits

Split X into X_1, X_2 on x_i

$e \leftarrow$ SplitEntropy(X_1, X_2) //

If $e <$ MinEnt MinEnt $\leftarrow e$; bestf $\leftarrow i$

Return bestf

Pruning

- Avoid overfitting.
  too specific for the training set.
  can not to generalize to other data.

- Once the tree has been fully grown, we prune it to avoid overfitting.

- At each step, we select the remaining leaf node that we expect by some criterion to be least ‘helpful’ for accuracy classification.

- One common pruning criterion is to compute a measure of confidence that indicates how much evidence there is that the node is ‘helpful’.

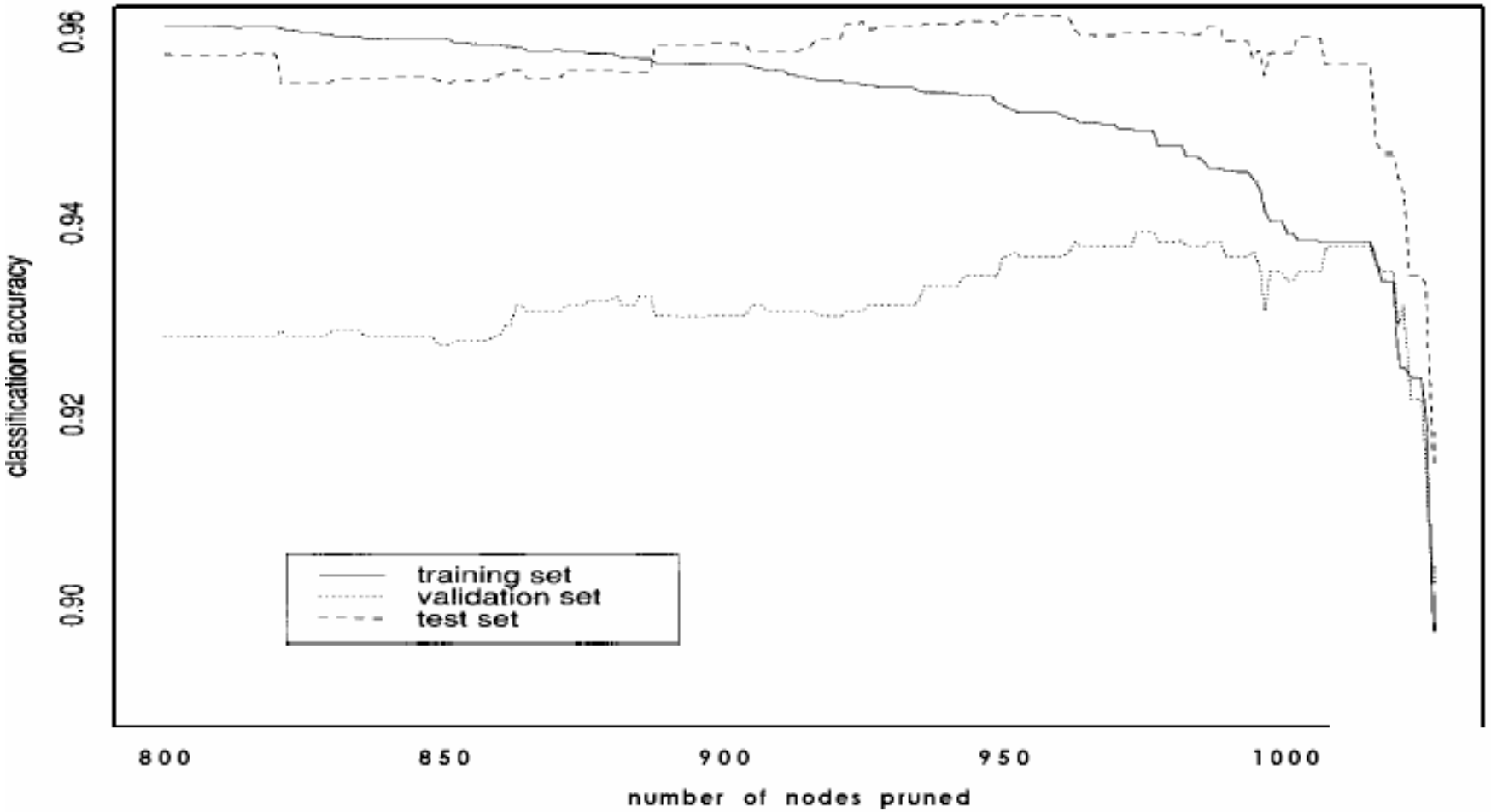


Figure 16.4 Pruning a decision tree. The graph shows how classification accuracy of a decision tree depends on pruning. Optimal performance on the test set (96.21% accuracy) is reached for 951 nodes pruned. Optimal performance on the validation set (93.91% accuracy) is reached for 974-977 nodes pruned. For these four pruned trees, performance on the test set is **96.00%,** close to optimal performance. Performance on the training set is monotonically decreasing.

Validation

- Validation evaluates a classifier on a held out data set (validation set) to assess its accuracy.
  {prune the decision tree to smaller size.}

- For the same reason as needing independent test data, in order to evaluate how much to prune a decision tree we need to look at a new set of data – which is what evaluation on the validation set does.

- Rather than using held out data for pruning, held out data can be used to train the parameters of a linear interpolation of all these distributions for each leaf node, and these interpolation distributions can then be used as the final classification functions.

N-fold cross validation

- Estimate a good validation set size.
- 5-fold cross-validation for example:
  - split the data into 5 parts
    - reserve 1 part as validation set and train the tree on other 4 parts.
    - prune the tree based on validation set.
    - repeat this process 4 times using each of the other 4 parts as a validation set.
  - Average size for optimal performance.

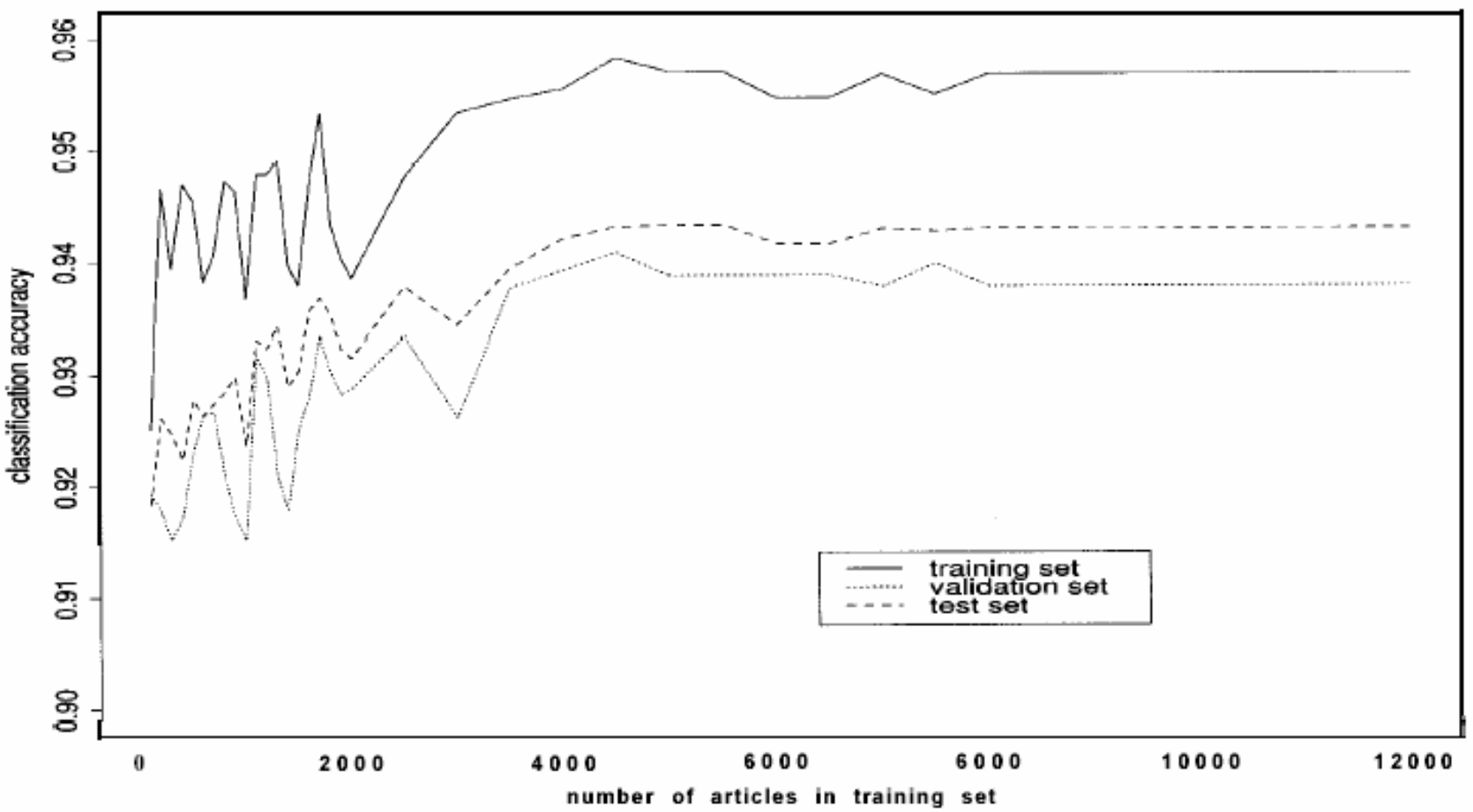


Figure 16.5 Classification accuracy depends on the amount of training data available. The x axis corresponds to the number of training documents the decision tree was trained on. The y axis corresponds to accuracy on the test set for a decision tree selected based on a constant size validation set. Classification accuracy is highly variable for small training set sizes and increases and levels off for larger sets.

Learning curve

- Computing learning curves like above is important to determine the size of an appropriate training set.
- Many training procedures are computationally expensive, so it is advantageous to avoid overly large training sets.
- On the other hand, insufficient training data will result in sub optimal classification accuracy.

Decision tree - advantage

- Easily to interpret and explain.
- Easily to trace the path from root to leaf node.

Perceptrons

- We present perceptrons here as a simple example of a *gradient descent* (or reversing the direction of goodness, *hill climbing*) algorithm, an important class of iterative learning algorithms.

- Perceptrons : linear separators.

- In gradient descent, we attempt to optimize a function of the data that computes a goodness criterion like square error or likelihood.

Perceptrons

- In each step, we compute the derivation of the function and change the parameters of the model in the direction of the steepest gradient (steepest ascent or descent, depending on the optimality function).

- This is a good idea because the direction of steepest gradient is the direction where we can expect the most improvement in the goodness criterion.

Perceptrons

- Text documents are represented as term vectors.
  Our goal is to learn a weight vector $\vec{w}$ and a threshold θ .

- We decide “yes” (the article is in the “earnings” category) if the inner product of weight vector and document vector is greater than the threshold and “no” otherwise:

$$\text{Decide "yes" iff } \vec{w} \cdot \vec{x}_j = \sum_{i=1}^{K} w_i x_{ij} > \theta$$

where K is the number of features ($K = 20$ as before)
x_{ij} is the component i of vector $\vec{x}_j$

Perceptrons

- The basic idea of the perceptron learning algorithm is simple.

- If the weight vector makes a mistake, we move it (and threshold) in the direction of greatest change for our optimality criterion.

$$\sum_{i=1}^{K} w_i x_{ij} - \theta$$

```
1  comment: Categorization Decision
2  funct decision(x⃗, w⃗, θ) ≡
3     if w⃗ · x⃗ > θ then
4                          return yes
5                    else
6                          return no
7     fi.
9  comment:  Initialization
10 w⃗ = 0
11 θ = 0
12 comment: Perceptron Learning Algorithm
13 while not converged yet do
14          for all elements x⃗_j in the training set do
15              d = decision(x⃗_j, w⃗, θ)
16              if class(x⃗_j) = d then
17                                    continue
18              elsif class(x⃗_j) = yes and d = no then
19                                                    θ = θ − 1
20                                                    w⃗ = w⃗ + x⃗_j
21              elsif class(x⃗_j) = no and d = yes then
22                                                    θ = θ + 1
23                                                    w⃗ = w⃗ − x⃗_j
24              fi
25          end
26 end
```

Error-correcting

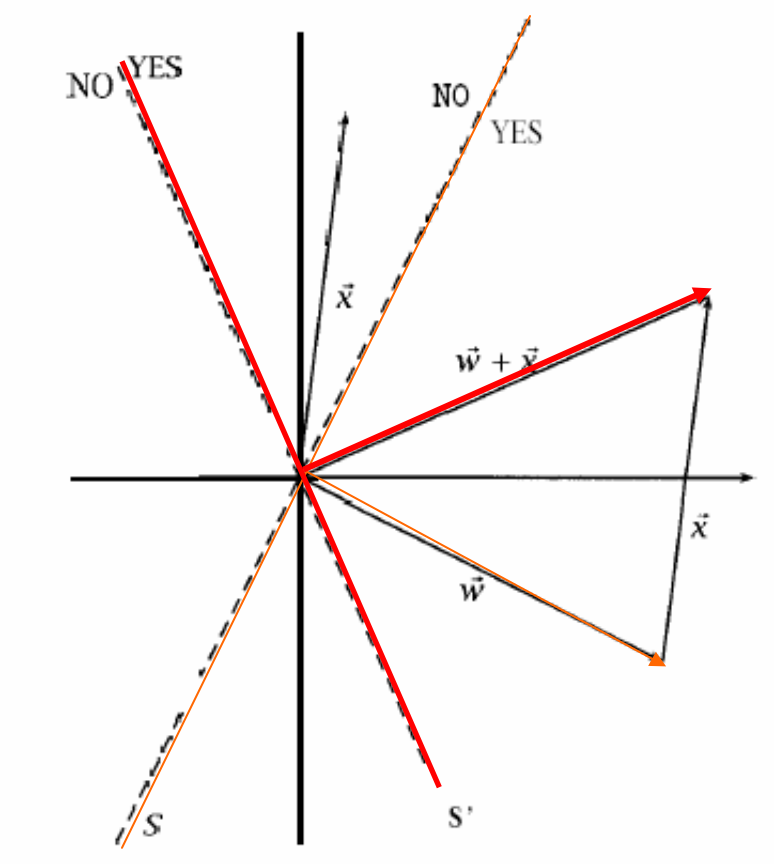


Figure 16.8 One error-correcting step of the perceptron learning algorithm. Data vector $\vec{x}$ is misclassified by the current weight vector $\vec{w}$ since it lies on the ••o•• side of the decision boundary S. The correction step adds $\vec{x}$ to $\vec{w}$ and (in this case) corrects the decision since $\vec{x}$ now lies on the ••es••-side of S•• the decision boundary of the new weight vector $\vec{w}+\vec{x}$.

Perceptrons

- The weights learned by the perceptron learning algorithm for the “earnings” category after about 1000 iterations.

Word	w^l	Weight
vs		11
mln		6
cts		24
,		2
&		12
000		- 4
loss		19
		- 2
"		7
3		- 7
profit		31
dlrs		1
1		3
pct		- 4
is		- 8
s		- 12
that		- 1
net		8
It		11
at		- 6
θ		37

Perceptrons

- A perceptron in 20 dimensions is hard to visualize, so we reran the algorithm with just two dimensions, *mln* and *cts*.

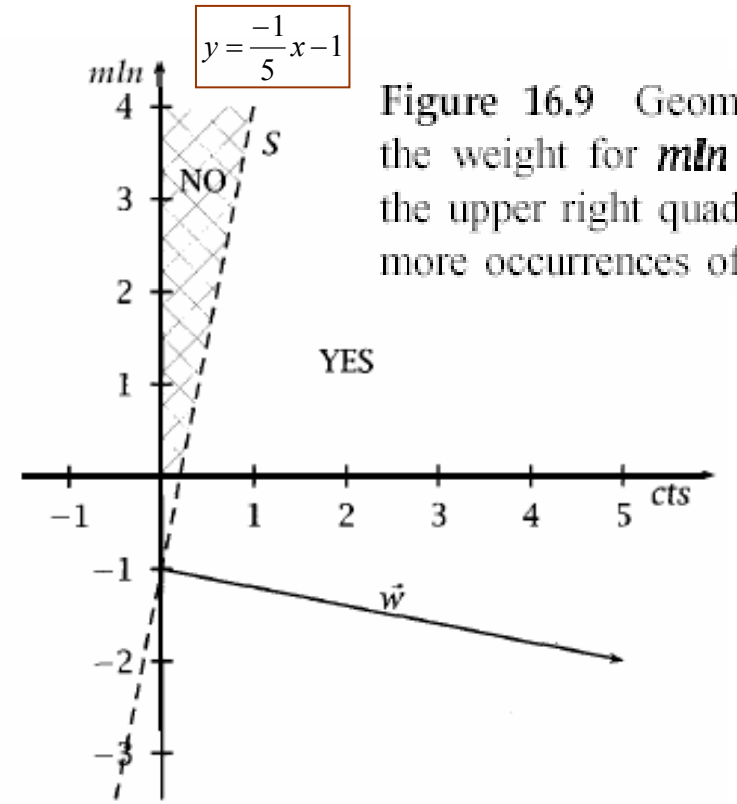


Figure 16.9 Geometric interpretation of a perceptron. The weight for ***cts*** is 5, the weight for ***mln*** is -1, and the threshold is 1. The linear separator ***S*** divides the upper right quadrant into a NO and a YES region. Only documents with many more occurrences of ***mln*** than ***cts*** are categorized as not belonging to "earnings."

Perceptrons - conclusion

- Perceptrons have not been used much in NLP because most NLP problems are not linearly separable and the perceptron learn algorithm does not find a good approximation separator in such cases.

- However, in cases where a problem is linearly separable, perceptron can be an appropriate classification method due to their simplicity and ease of implement.

k Nearest Neighbor classification

- The rationale for the nearest neighbor classification is remarkably simple. To classify a new object, find the object in the training set that is most similar. Then assign the category of this nearest neighbor.

- The basic idea is that if there is an identical article in the training set (or at least one with the same representation), then the obvious decision is to assign the same category. If there is no identical article, then the most similar one is our best bet.

k Nearest Neighbor classification

- A generalization of the nearest neighbor rule is k nearest neighbor or KNN classification.
- Instead of using only one nearest neighbor as the basis for our decision, we consult k nearest neighbors.
- KNN for k>1 is more robust than the “1 nearest neighbor”.
- The complexity of KNN is in finding a good measure of similarity.
- If one does not have a good similarity metric, one can not use KNN.

k Nearest Neighbor classification

- For the "earnings" data, we implemented cosine similarity, and chose k=1. This "1NN algorithm" for binary categorization can be stated as follows.
  - Goal: categorize $\vec{y}$ based on the training set X.
  - Determine the largest similarity with any element in the training set: $\mathbf{sim}_{\max}(\vec{y}) = \max_{\vec{x} \in X} \mathbf{sim}(\vec{x}, \vec{y})$
  - Collect the subset of X that has highest similarity with $\vec{y}$:

    $A = \{\vec{x} \in X | \mathbf{sim}(\vec{x}, \vec{y}) = \mathbf{sim}_{\max}(\vec{y})\}$
  - Let n_1 and n_2 be the number of elements in A that belong to the two classes c_1 and c_2, respectively. Then we estimate the conditional probabilities of membership as follows:

    $$P(c_1|\vec{y}) = \frac{n_1}{n_1 + n_2} \qquad P(c_2|\vec{y}) = \frac{n_2}{n_1 + n_2}$$
  - Decide c_1 if $P(c_1|\vec{y}) > P(c_2|\vec{y})$, c_2 otherwise.

KNN - conclusion

- The main difficulty with KNN is that its performance is very dependent on the right similarity metric.

- Computing similarity with all training exemplars takes more time than computing a linear classification function or determining the appropriate path of a decision tree.

Comparison

••arnings" assigned?	••arnings•• correct? YES	NO
YES	*1024*	69
NO	*63*	*2143*

Table 16.5 Contingency table for a decision tree for the Reuters category ••arnings.•• Classification accuracy on the test set is 96.0%.

"earnings" assigned?	"earnings" correct? YES	NO
YES	1059	521
NO	28	1691

Table 16.11 Classification results for the perceptron in table 16.10 on the test set. Classification accuracy is 83.3%.

"earnings" assigned?	"earnings" correct? YES	NO
YES	1022	91
NO	65	2121

Table 16.12 Classification results for an 1NN categorizer for the ••arnings" category. Classification accuracy is 95.3%.