Machine Learning and Neural Networks

Many engineering tasks involve classification or prediction:

- **Parts Inspection**. Classify a part into "defective" or "ok".
- **Veneer Processing**. Identify flaws in plywood veneer and decide how to handle them: "ignore", "cut out and plug", or "unrepairable".
- **PAP Smear Analysis**. Given a cell from a PAP smear, estimate the probability that it is "normal", "precancerous", or "cancerous".
- **Document Understanding**. Given a rectangular region from a scanned document, classify it is "text", "graphic", "picture", "horizontal line", "vertical line".
- **Optical Character Recognition**. Given an image of a character, classify it as "A", "B", ..., "Z".

Building Classification Programs

There are two main methodologies for building classification programs:

- **Interview Experts**. Interview an expert and encode his/her expertise as a program. Often uses rule-based programming.
- **Learning From Examples**. Collect a database of cases and have the expert label each case with the correct classification.

  Then apply a machine learning program to analyze the database of cases and construct a *classification rule* or *discriminant function*.

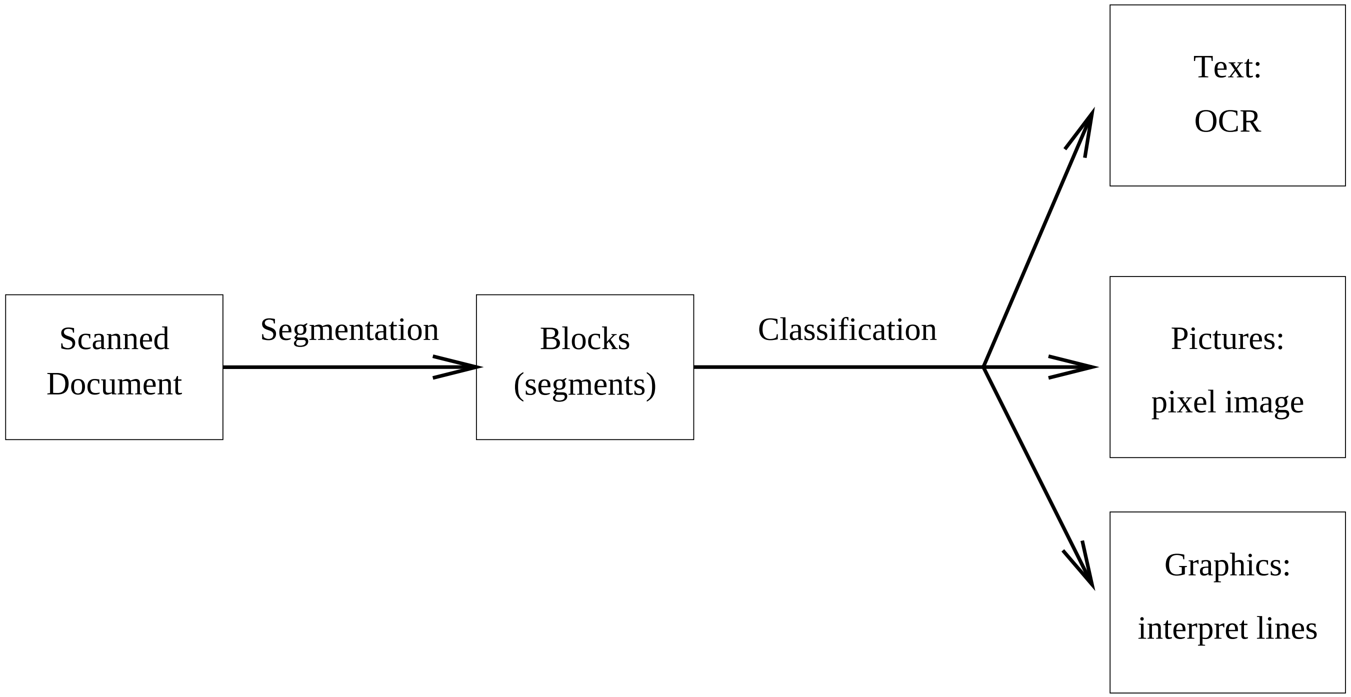
For tasks involving human sensory processing, it is often difficult or impossible to obtain useful information from interviewing experts. Humans can't introspect on these processes.

Even in other tasks, the ability of experts to introspect accurately is limited.

Machine Learning Programs

- **Represent each case as a vector of features:** $\mathbf{x} = (x_1, \ldots, x_n)$.
- **Represent the classification rule as a function that maps feature vectors into a discrete set of** k **classes.** $f(\mathbf{x}) \in \{0, 1, \ldots, k-1\}$.
- **Represent each training example as a pair:** $\langle \mathbf{x}, f(\mathbf{x}) \rangle$.

An Example: Learning to Classify Page Blocks



Two classification problems:

- **Classifying Segments:** text, graphics, picture, horizontal line, vertical line.
- **Classifying Individual Letters:** A, B, C, . . . , Z.

Features Describing a Block

- **height:** Height of the block.
- **length:** Length of the block.
- **area:** Area of the block ($height \times length$).
- **eccen:** Eccentricity of the block ($length/height$).
- **blackpix:** Total number of black pixels in bitmap of the block.
- **wbtrans:** Number of white-black transitions in the bitmap of the block.
- **meantr:** Mean number of white-black transitions ($blackpix/wbtrans$).
- **pblack:** Percentage of black pixels within the block ($blackpix/area$);
- **blackand:** Total number of black pixels in the bitmap of the block after Run Length Smoothing Algorithm (RLSA).
- **pand:** Percentage of black pixels after RLSA ($blackand/area$).

$$f(\langle 48, 402, 19296, 8.375, .055, .094, 4.13, 1058, 1814, 256 \rangle) = 5$$

How Can We Guess f?

Given a finite sample, it is impossible to guess the true function f:

x	$f(x)$
1	1
2	4
3	9
4	16
5	??

One Approach: Find some pattern that is true of all of the training examples, and assume that the pattern will hold for future examples too.

What is a Pattern?

- **Assume f has some simple functional form**

$$f(x) = a + bx + cx^2$$

  for some values of a, b, and c.

- **Assume f can be expressed as a set of simple rules**

- **Assume f can be expressed as a small neural network**

Let $\mathcal{F}$ be the space of patterns that a learning program considers. The learning program must choose some hypothesis $\hat{f}$ from $\mathcal{F}$.

How Can We Choose Which Pattern is Best?

Our real goal is to find an $\hat{f}$ that will correctly classify new examples. So the ideal thing would be to choose $\hat{f}$ to minimize

$$Pr(\hat{f}(\mathbf{x}) \neq f(\mathbf{x}) | \mathbf{x} \text{ is a new case}).$$

But we can't compute this probability, because we don't have the new cases.

- **Minimize number of errors on the training data**
- **Use the training data to estimate new data errors. Minimize number of estimated new data errors.**

Fitting Hypotheses to Data

Given: Training examples of the form $\langle \mathbf{x}, f(\mathbf{x}) \rangle$,
for some *unknown function* f.

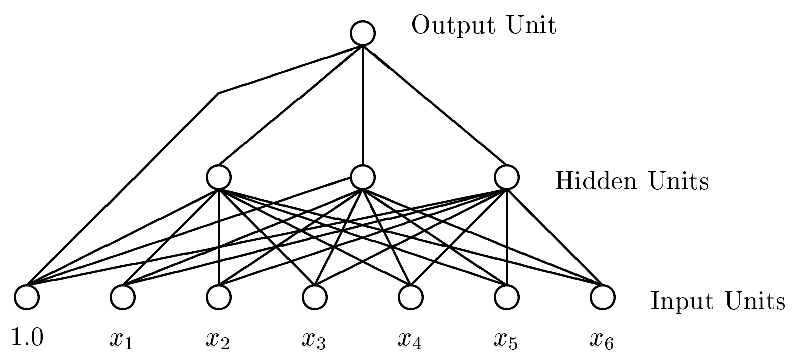
A space of possible functions $\mathcal{F}$ to consider.

An error function $J(\hat{f})$ that measures the error rate of $\hat{f} \in \mathcal{F}$.

Find: The function $\hat{f} \in \mathcal{F}$ that minimizes the error $J(\hat{f})$.

Neural Networks

We represent our hypotheses by so-called "three layer networks"



The circles are called "units".

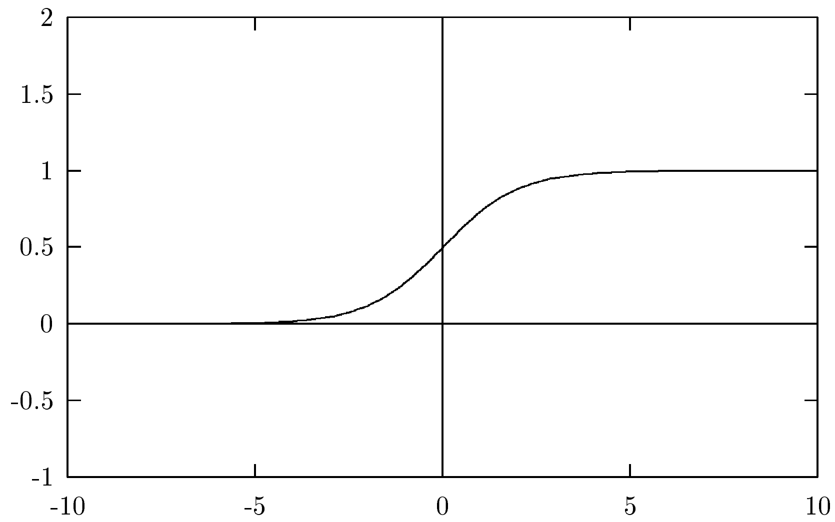
Each edge in the graph corresponds to a parameter value or "weight".

The input units do no computation.

The hidden and output units compute $\sigma(\mathbf{W} \cdot \mathbf{Y})$.

Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$



A three-layer network in algebra:

$$\hat{f}(\mathbf{X}, W) = \sigma\left(\mathbf{W}_o \cdot [\sigma(\mathbf{W}_1 \cdot \mathbf{X}), \sigma(\mathbf{W}_2 \cdot \mathbf{X}), \ldots, \sigma(\mathbf{W}_h \cdot \mathbf{X})]\right)$$

The Bias Input

The sigmoid function can approximate a linear function:

$$y = w_0 + w_1 x_1 + \ldots + w_n x_n$$

The weight w_0 is the *intercept term*. To make the computation uniform, we can pretend there is an x_0 feature whose value is fixed at 1.0.

Using this convention, w_0 is called the "bias term."

Error Measures

For a network with a single output unit:

- **Squared Error**

$$J(S,W) = \sum_{\mathbf{X}\in S} [f(\mathbf{X}) - \hat{f}(\mathbf{X},W)]^2$$

- **Cross Entropy**

$$J(S,W) = \sum_{\mathbf{X}\in S} [-f(\mathbf{X}) \log \hat{f}(\mathbf{X},W) \ - \ (1 - f(\mathbf{X})) \log(1 - \hat{f}(\mathbf{X},W))]$$

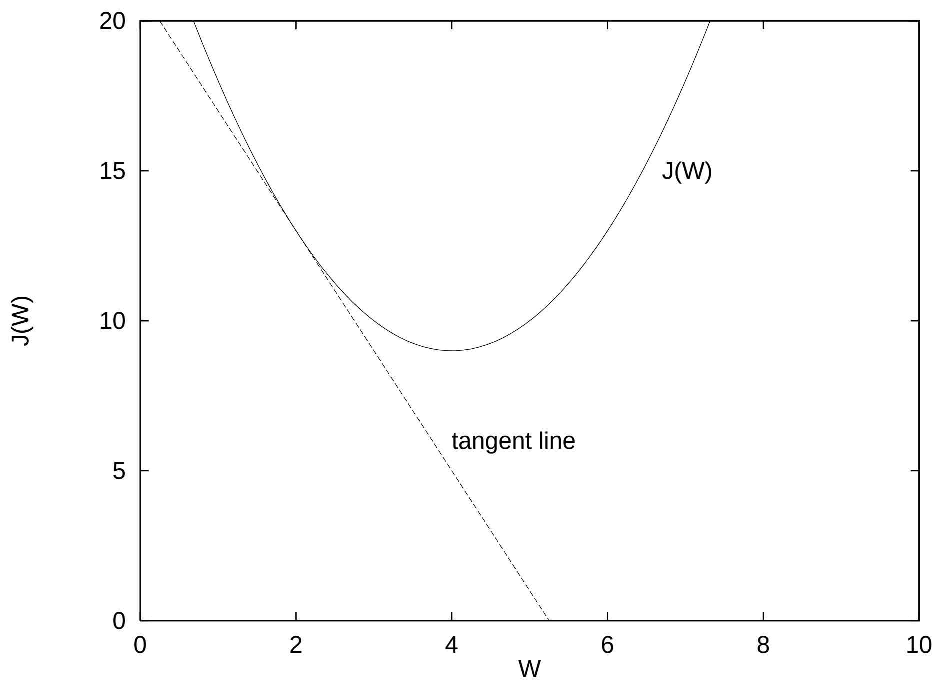
For multiple outputs, sum for each output.

Gradient Search Methods

Approximate J(W) by the first two terms of the Taylor's series:

$$J(S,W) \doteq J(S,W_0) + \frac{dJ(S,W)}{dW}(W - W_0).$$

To decrease J, W should move in the direction opposite the first derivative of $J(S,W)$.



Simple Gradient Descent

The gradient is the vector of first partial derivatives:

$$\nabla_{\mathbf{W}} J(S,W) = \left(\frac{\partial J(S,W)}{\partial w_1}, \ \ldots, \ \frac{\partial J(S,W)}{\partial w_N} \right).$$

Take a step in weight space in direction of negative derivative.

$$W_{new} := W_{old} - \eta \nabla_W J(S,W)$$

The parameter η is the learning rate or step size parameter.

Computing the gradient requires one complete pass through all of the training data. This is called an *epoch*.

Initializing Gradient Descent

The search must start from some starting point.

Goal: All units should be in their "linear" region.

For each unit, the weights can be initialized to

$$[-1, +1] \times \frac{1}{\sqrt{\text{fan-in}}}.$$

Stochastic Gradient Descent

Update the weights for each training example one-at-a-time, rather than summing over all of S:

$$W := W - \eta \nabla_W J(\mathbf{X}_i, W)$$

- Much faster to compute.
- Converges faster in practice.
- Injects some noise into the search—escapes local minima.

Stochastic Gradient Descent with Momentum

To reduce the "noise", we can smooth the weight updates.

$$\Delta W := -\eta \nabla_W J(\mathbf{X}_i, W) + \mu \Delta W.$$

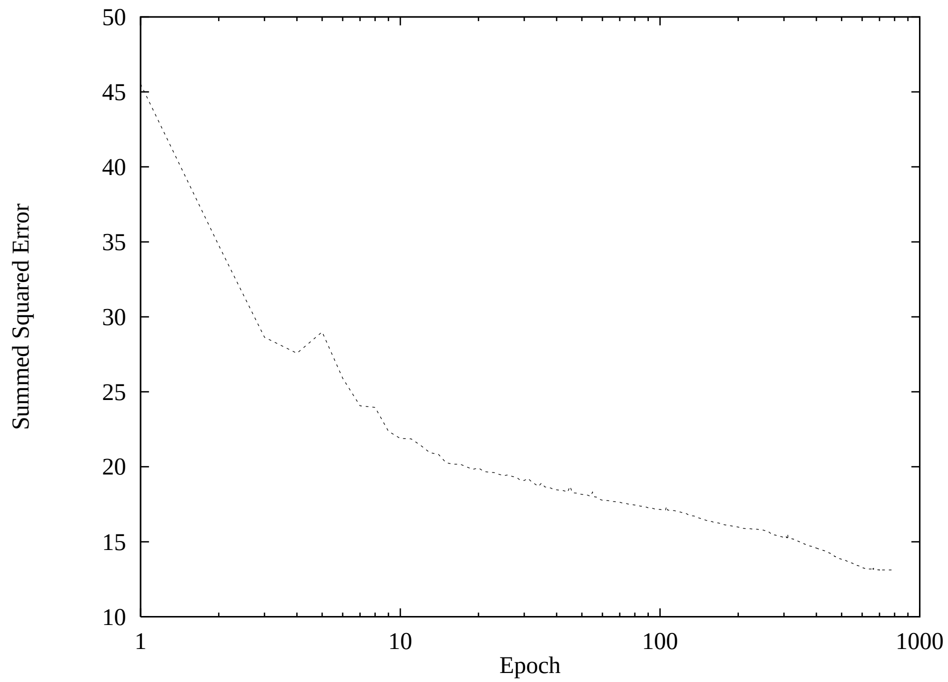
μ is typically set in the range [0.7,0.95].

Conjugate Gradient Search

1. Compute full gradient.
2. Search along $J(S, W - \alpha \nabla_W J(S, W))$ to find the value of α that minimizes J.
3. Repeat from step 1 until J ceases to improve.

- **Advantage:** Automatically chooses learning rate.
- **Disadvantage:** Slow if S is large.

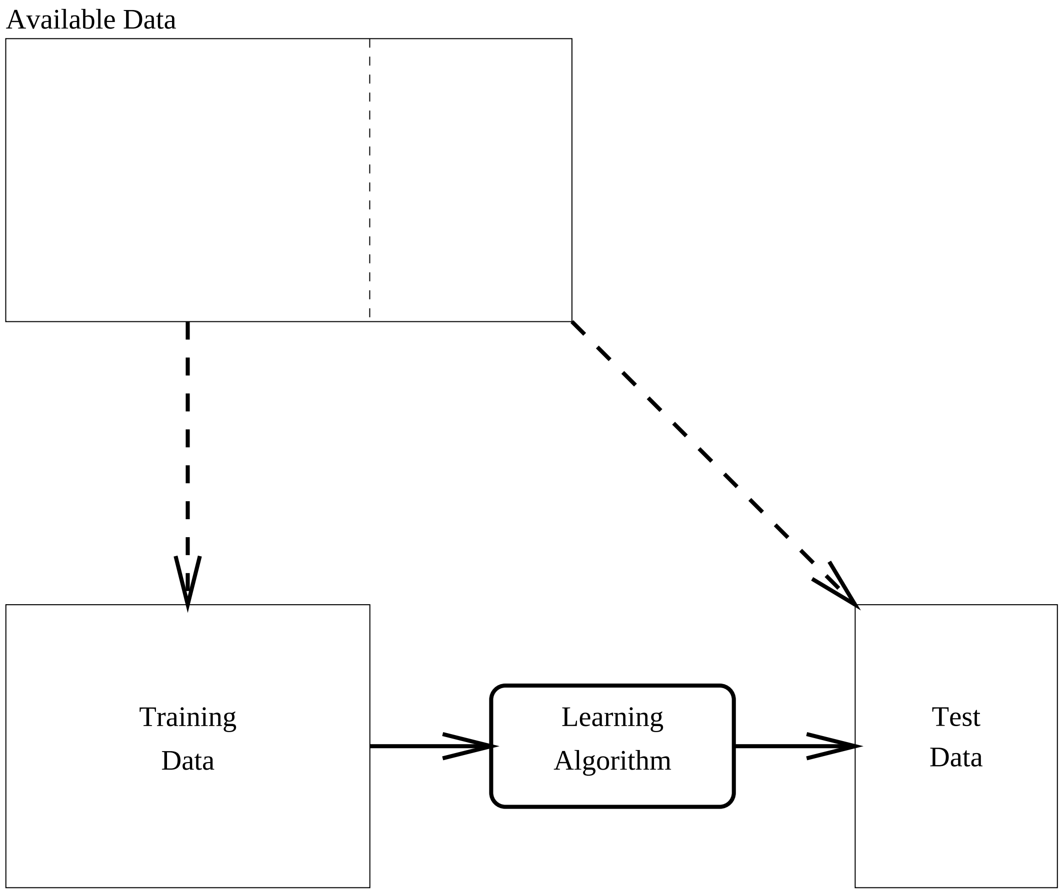
Conjugate Gradient Training on Page Blocks



How Can We Measure The Quality of $\hat{f}$?

Question: How can you validate the quality of the learned neural network?

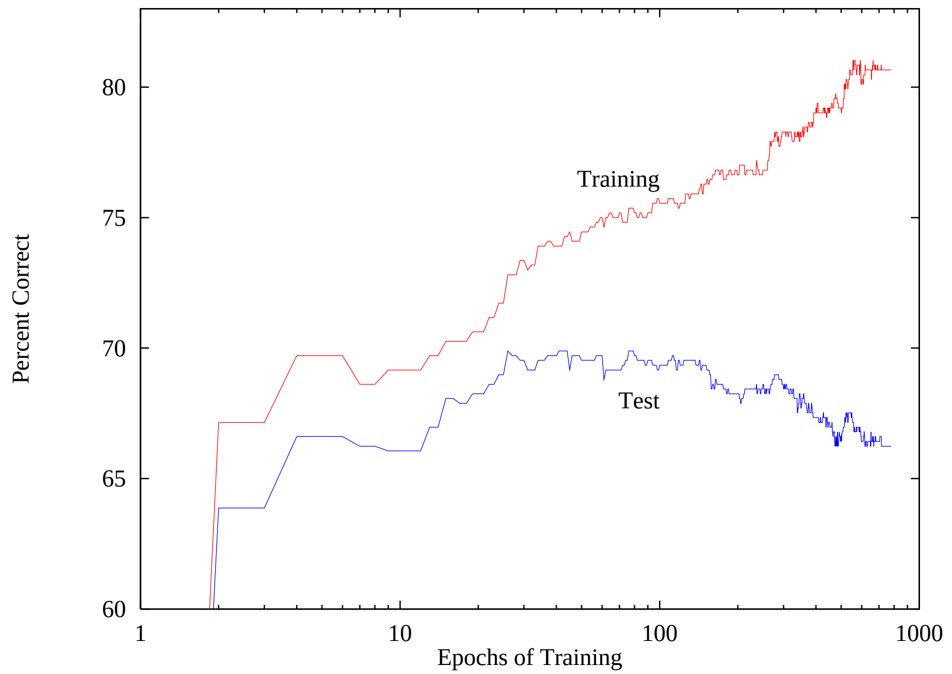
Answer: Set aside a random sample of your training data to use for testing.



The Problem of Overfitting

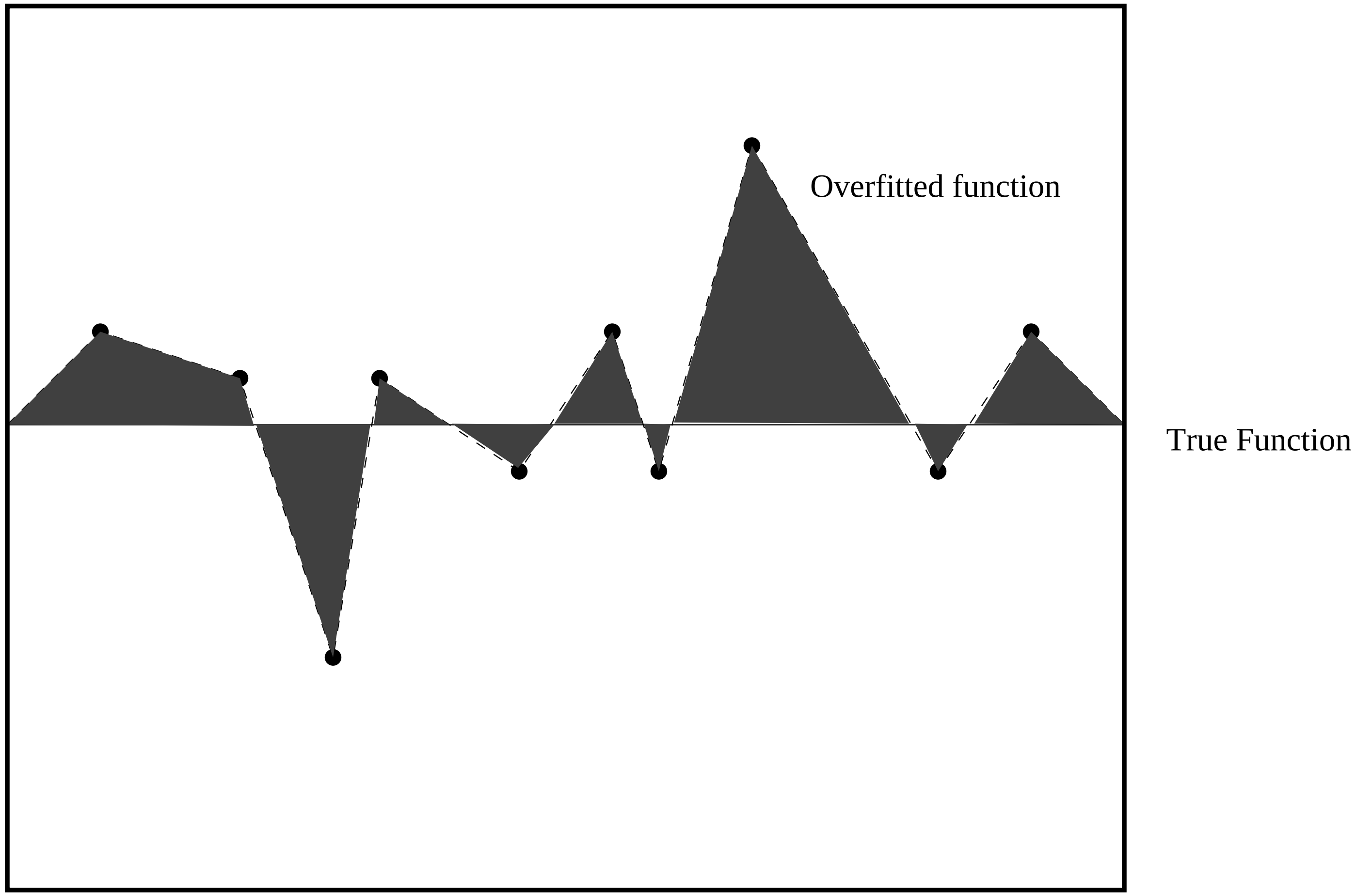
So far we have only considered minimizing the *training set error*. However, what we really want to do is minimize the error on new examples.

Minimizing training set error does not minimize error on new cases.

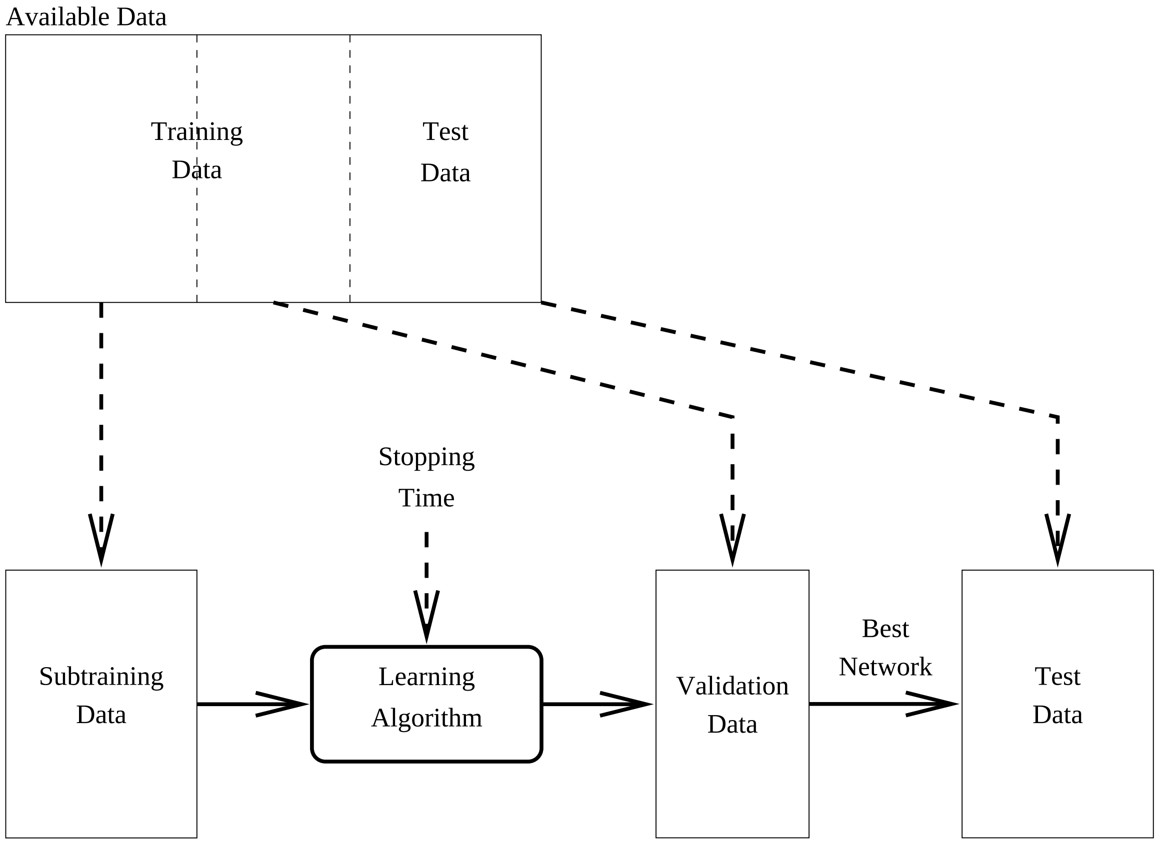


Why Overfitting Happens

The longer you train a network, the better it fits the training data. Ultimately, it will memorize the data exactly. If there is any noise in the data, the network will *fit the noise*.

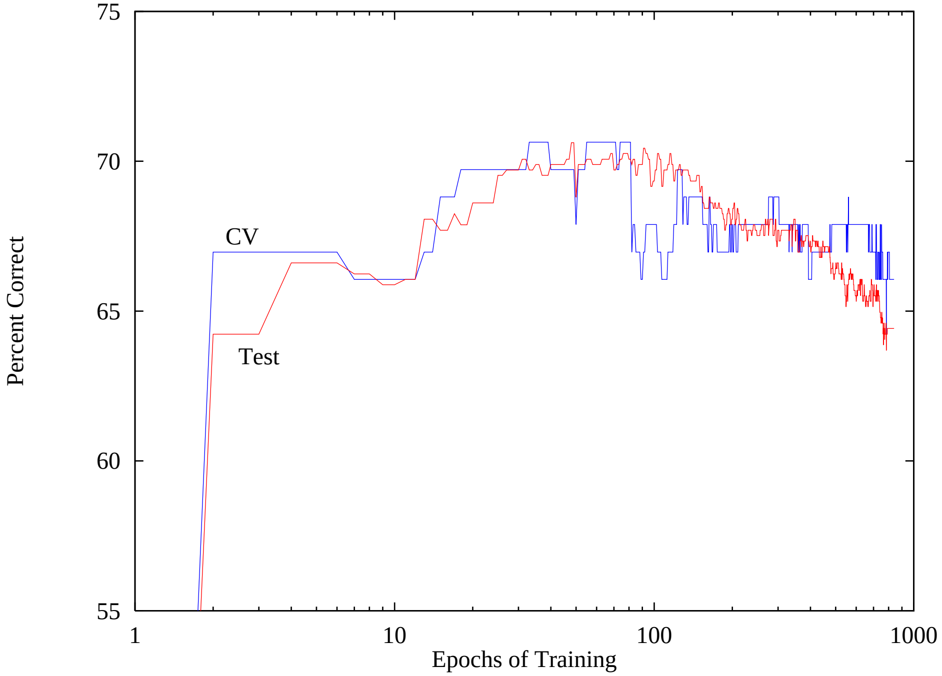


Using A Validation Set to Estimate New Data Error Rate



Subdivide the training data. Train on the "subtraining" data and test $\hat{f}$ using the "validation" data. Keep the network that minimizes validated error. The "validation set" is sometimes called the "halting set".

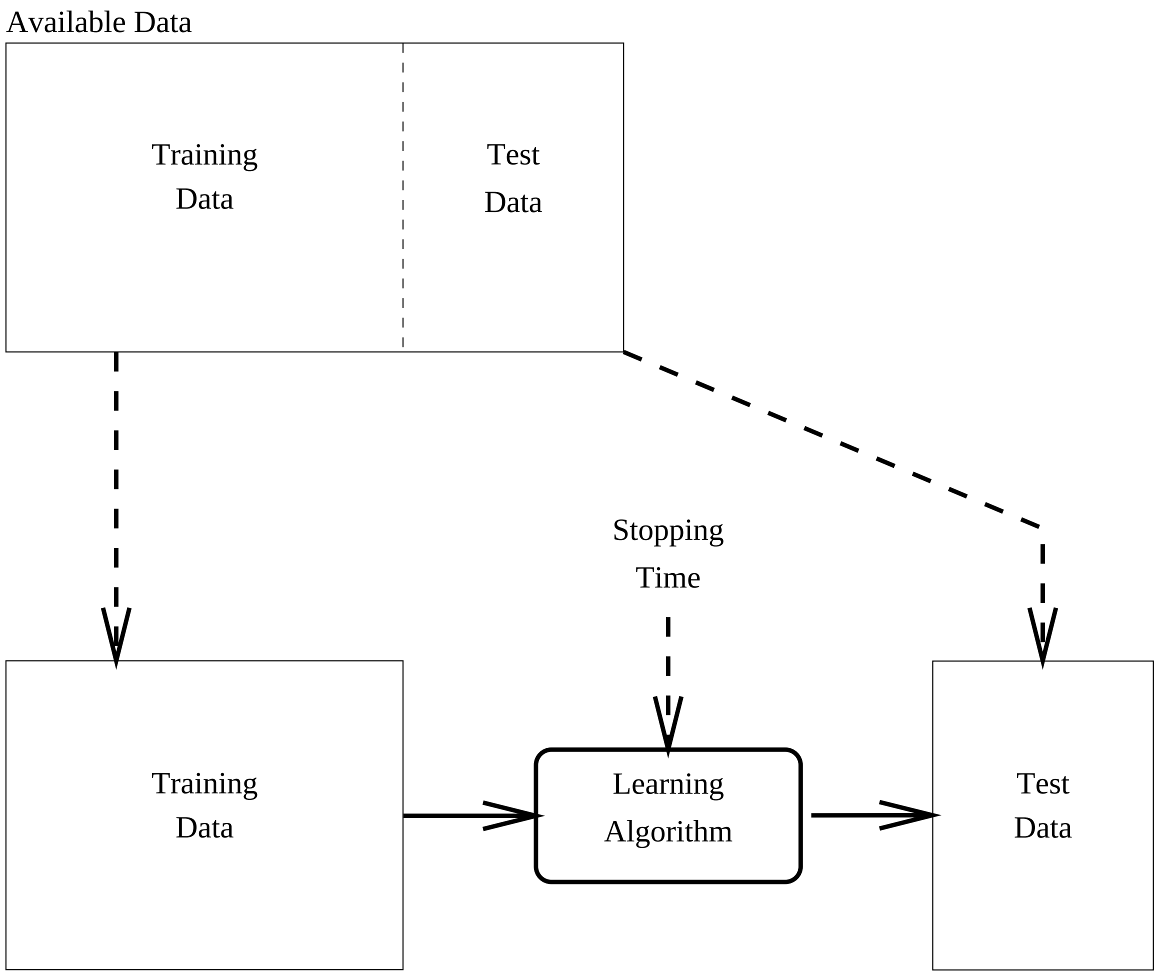
Early Stopping



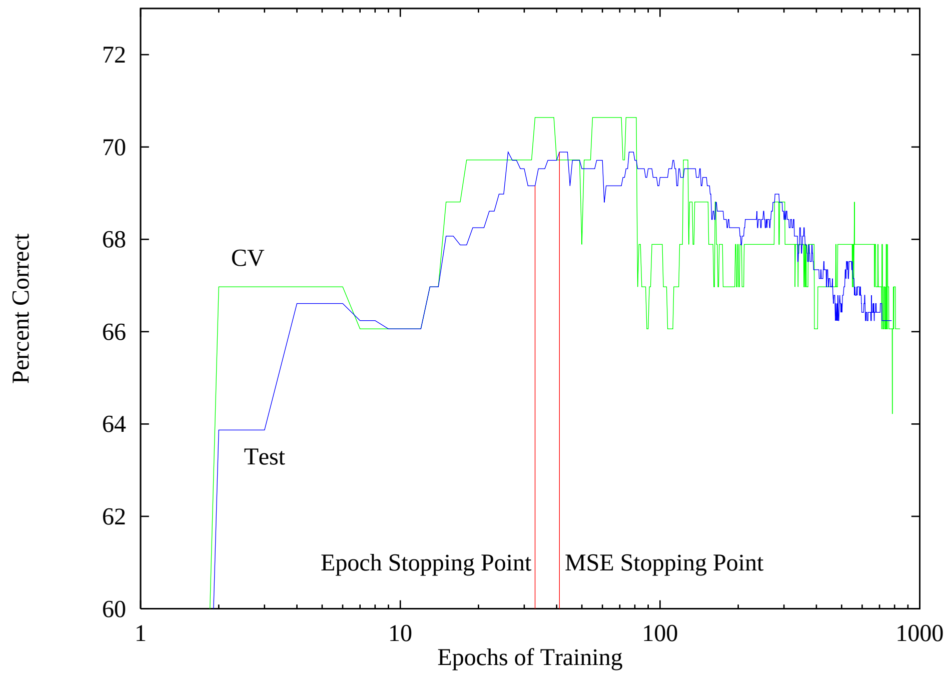
We can see that the validation performance is a good predictor of the test set performance. In a real application, we would not plot the test set on this picture. We choose the optimal stopping point at the first point where the validation performance achieves its optimal value.

Using A Halting Set to Choose Full Stopping Point

A problem with this approach is that the validation data are not used very much during training. One solution is the "reconstitute" the training set and use the previous validation run to decide when to halt training.



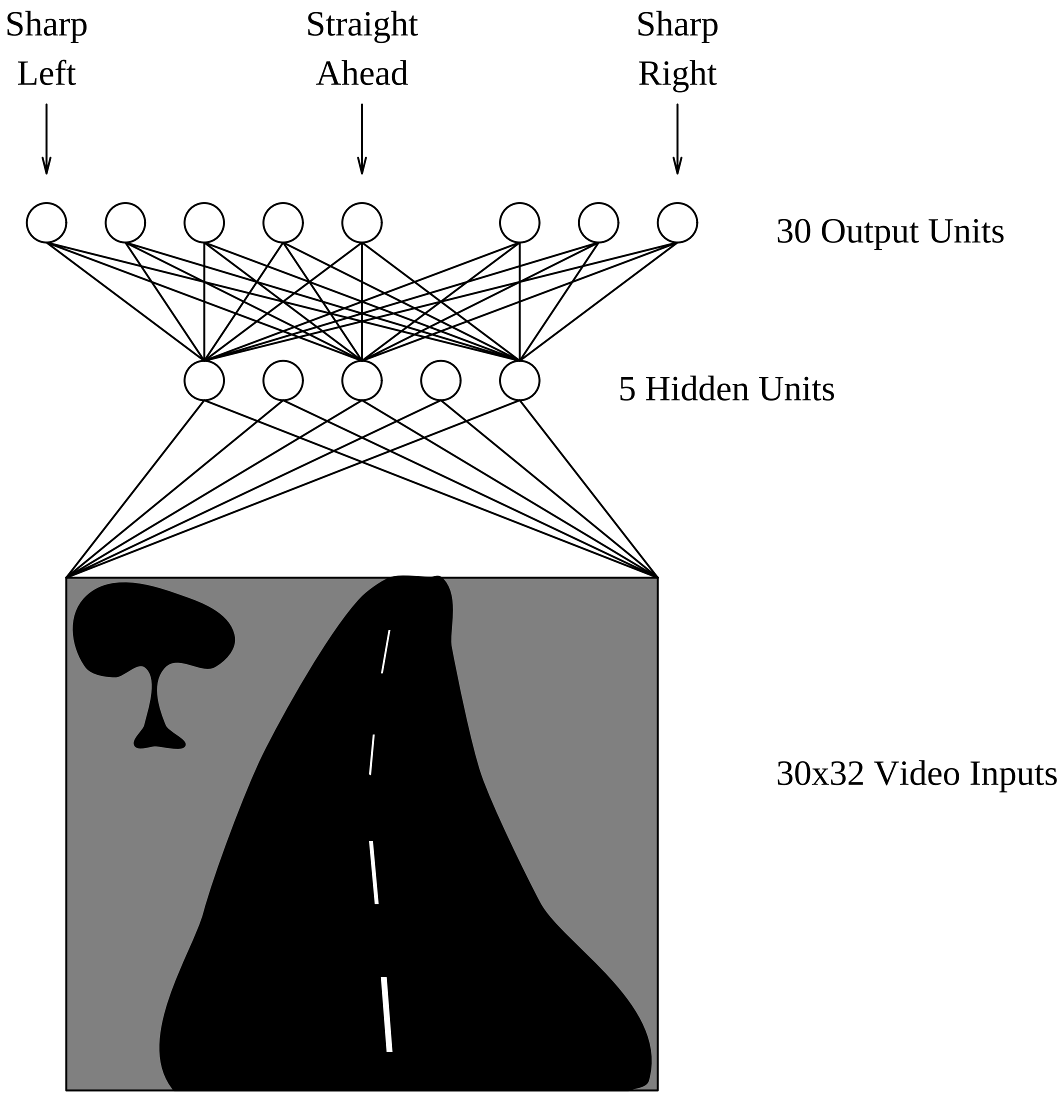
Reconstituted Early Stopping



We can stop either when the MSE on the training set matches the predicted optimal MSE or when the number of epochs matches the predicted optimal number of epochs.

ALVINN (Pomerleau)

Task: Learn to steer the CMU NavLab van

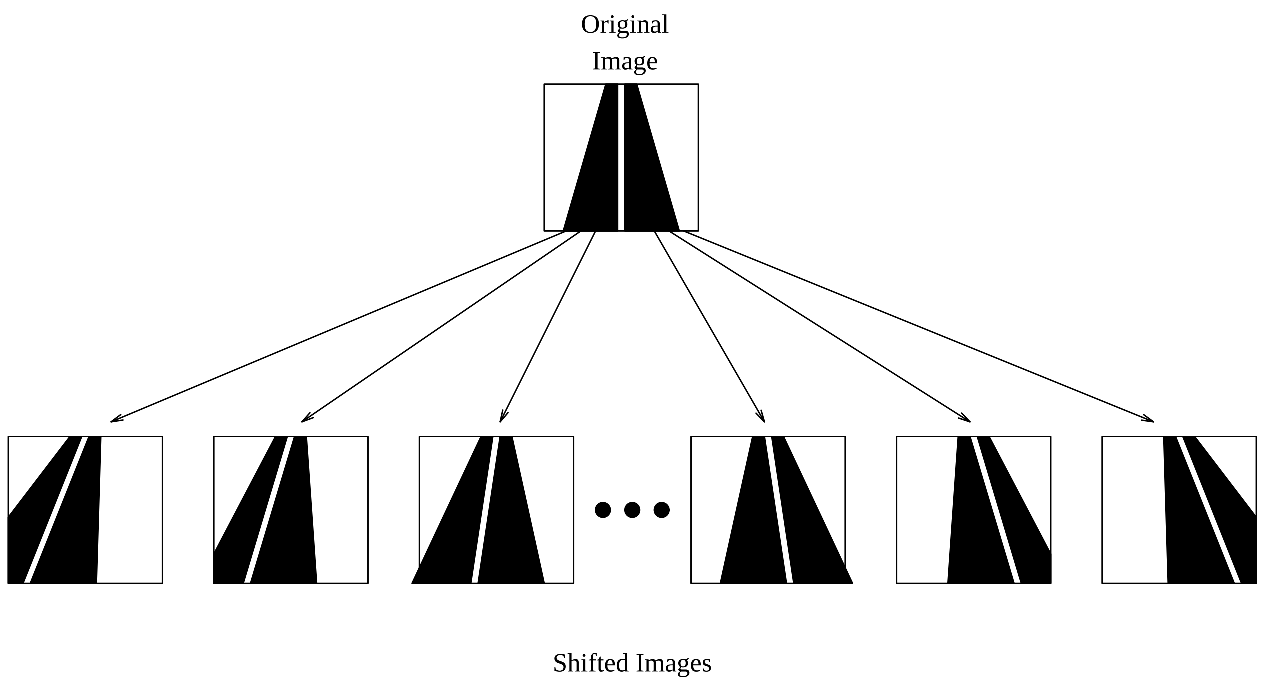


ALVINN (2)

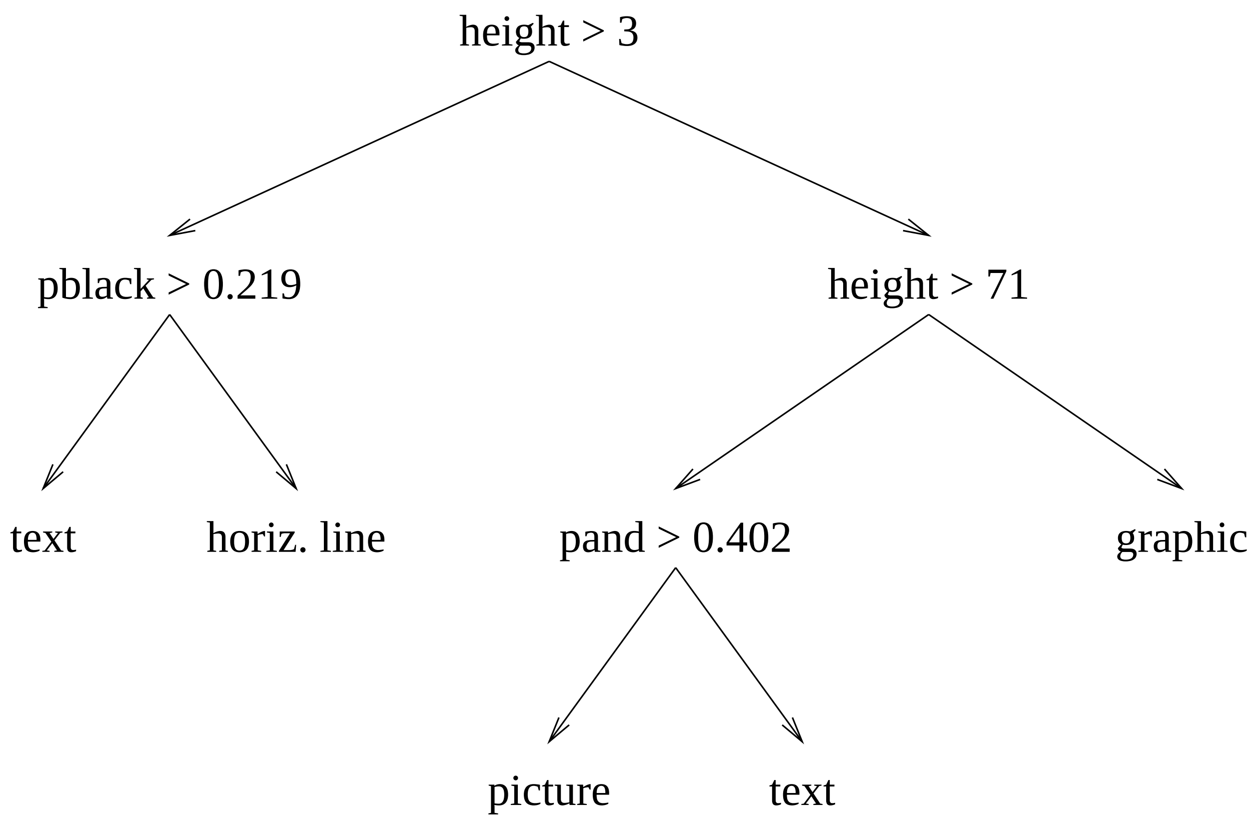
- **Outputs encoded as overlapping range units.**
- **Supervised training from human drivers.**

  Does not provide “off road” experiences.
- **Training from perturbed images.**

  Apply camera model to shift image and driving signal.



Decision Tree Learning



To classify a case, we start at the root and apply the indicated test. If the test is satisfied, we take the right branch. Otherwise, we take the left branch. When we reach a leaf node, it gives the value of $\hat{f}$.

Divide-and-Conquer Algorithm

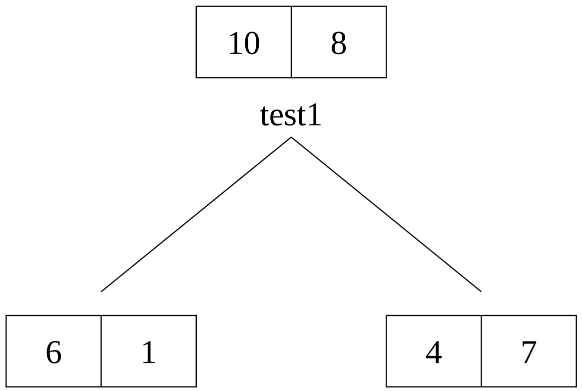
The algorithm C4.5 (and closely related methods called CART and CHAID) grows the tree starting at the root. Given a set of training examples, it evaluate every possible splitting test at the root and chooses one (using a scoring function). It then splits the data according to the chosen test and processes the two child nodes recursively.

Top-Down Divide-And-Conquer

```
void buildTree(node & N)
{
  if(N.pure()) {
      N.leaf = 1; N.class = N.majorityClass();
      }
  else {
      float bestScore = 0.0, thisScore = 0.0;
      test * bestTest;
      listIterator<test *> itr(N.possibleTests());
      for(itr.init(); !itr; ++itr) {
        thisScore = itr().score(N);
        if(thisScore > bestScore) {bestScore = thisScore; bestTest = itr();}
      }
      N.test = bestTest;
      N.left = new node(N.examplesSatisfying(bestTest));  // create node and initialize its training examples
      N.right = new node(N.examplesFailing(bestTest)); // create node and initialize its training examples
      buildTree(*N.left); buildTree(*N.right);
      }
}
```

Scoring a Test Using Mutual Information

Suppose there are only two classes, and our current node `N` contains 10 examples from class 0 and 8 examples from class 1. If we performed `test1`, the decision tree would look like this:



Let

$$H(i,j) = -\frac{i}{i+j}\log\frac{i}{i+j} - \frac{j}{i+j}\log\frac{j}{i+j}.$$

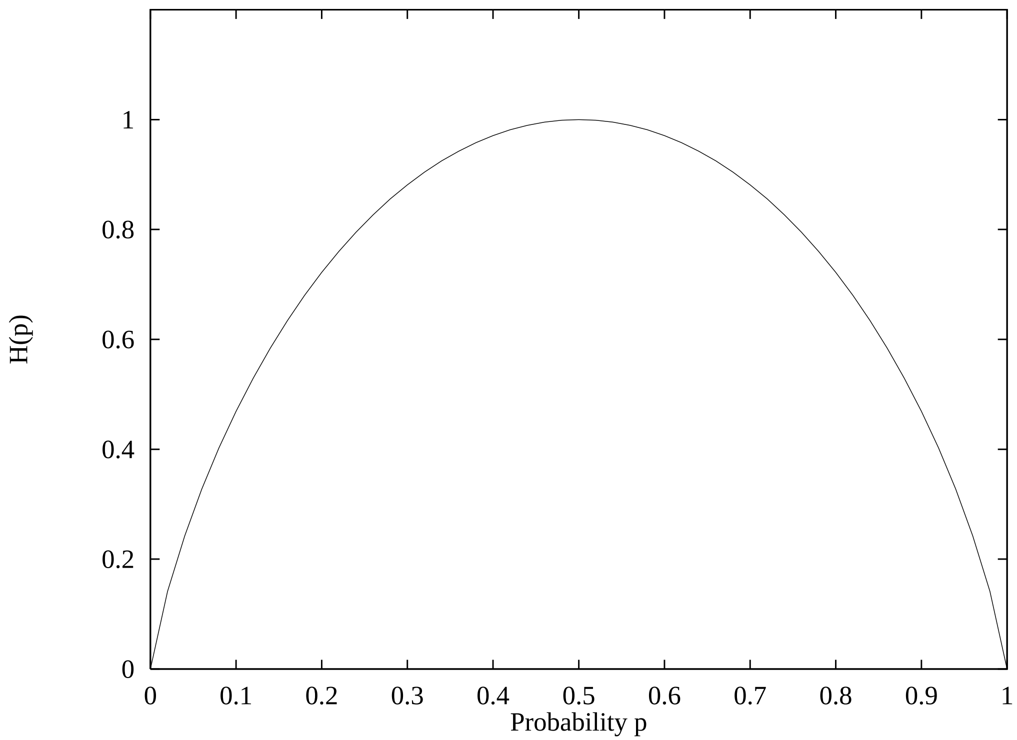
Then

$$score = H(10,8) - \left(\frac{7}{18}H(6,1) + \frac{11}{18}H(4,7)\right) = 0.1830.$$

This estimates how much information `test1` tells us about the class.

H is the Entropy Function

$$H(i,j) = H\left(\frac{i}{i+j}\right)$$



$$H(p) = -p\log p - (1-p)\log(1-p)$$

Controlling Overfitting via Tree Pruning

If we run C4.5 on the Page Block data, we get the following tree:

```
p_black <= 0.09 : 5 (7.0/2.0)
p_black > 0.09 :
|   height <= 3 :
|   |   p_black <= 0.219 : 1 (4.0)
|   |   p_black > 0.219 :
|   |   |   blackpix <= 7 :
|   |   |   |   lenght <= 10 : 1 (2.0/1.0)
|   |   |   |   lenght > 10 : 3 (2.0)
|   |   |   blackpix > 7 :
|   |   |   |   mean_tr > 71.13 : 2 (2.0/1.0)
|   |   |   |   mean_tr <= 71.13 :
|   |   |   |   |   p_black <= 0.302 : 2 (2.0/1.0)
|   |   |   |   |   p_black > 0.302 :
|   |   |   |   |   |   blackpix <= 11 :
|   |   |   |   |   |   |   area <= 9 : 2 (2.0)
|   |   |   |   |   |   |   area > 9 : 1 (4.0/1.0)
|   |   |   |   |   |   blackpix > 11 :
|   |   |   |   |   |   |   lenght <= 20 : 2 (4.0/1.0)
|   |   |   |   |   |   |   lenght > 20 :
|   |   |   |   |   |   |   |   lenght <= 38 : 2 (6.0)
|   |   |   |   |   |   |   |   lenght > 38 :
|   |   |   |   |   |   |   |   |   lenght <= 59 : 1 (3.0/1.0)
|   |   |   |   |   |   |   |   |   lenght > 59 : 2 (12.0/1.0)
|   height > 3 :
|   |   height > 71 : 3 (3.0/1.0)
|   |   height <= 71 :
|   |   |   p_and <= 0.402 :
|   |   |   |   height <= 8 : 5 (4.0)
|   |   |   |   height > 8 :
|   |   |   |   |   height <= 10 : 1 (2.0)
```

```
|   |   |   |   |   height > 10 : 5 (4.0/1.0)
|   |   |   p_and > 0.402 :
|   |   |   |   p_and <= 0.496 :
|   |   |   |   |   area <= 189 : 1 (5.0)
|   |   |   |   |   area > 189 :
|   |   |   |   |   |   lenght > 97 : 1 (4.0/1.0)
|   |   |   |   |   |   lenght <= 97 :
|   |   |   |   |   |   |   eccen <= 7.9 : 4 (5.0/1.0)
|   |   |   |   |   |   |   eccen > 7.9 : 3 (2.0)
|   |   |   |   p_and > 0.496 :
|   |   |   |   |   p_and <= 0.53 :
|   |   |   |   |   |   p_and <= 0.51 : 1 (3.0)
|   |   |   |   |   |   p_and > 0.51 : 5 (3.0)
|   |   |   |   |   p_and > 0.53 :
|   |   |   |   |   |   p_black <= 0.921 :
|   |   |   |   |   |   |   height <= 25 :
|   |   |   |   |   |   |   |   mean_tr <= 16.63 :
|   |   |   |   |   |   |   |   |   p_and <= 0.981 :
|   |   |   |   |   |   |   |   |   |   height <= 4 : 1 (5.0/2.0)
|   |   |   |   |   |   |   |   |   |   height > 4 :
|   |   |   |   |   |   |   |   |   |   |   p_black <= 0.204 :
|   |   |   |   |   |   |   |   |   |   |   |   blackand <= 2741 :
|   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.2 : 3 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.2 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 119 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 19 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 19 : 3 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 119 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 29 : 4 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 29 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.769 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.93 : 1 (3.0/2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.93 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.28 : 1 (8.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.28 :
```

```
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.42 : 1 (6.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.42 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 161 : 4 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 161 : 1 (4.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.769 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   height <= 7 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   height > 7 : 3 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   blackand > 2741 :
|   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 2980 : 2 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 2980 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   p_black > 0.204 :
|   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.526 :
|   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.626 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.618 : 1 (22.0/3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.618 : 3 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.626 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.653 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.312 : 1 (5.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.312 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   height <= 12 : 4 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   height > 12 : 2 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.653 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 5 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.238 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.232 : 2 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.232 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.44 : 1 (9.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.44 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.53 : 2 (4.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.53 : 1 (5.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.238 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.407 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area > 3980 : 1 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area <= 3980 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   eccen > 13.444 : 1 (30.0)
```

```
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   eccen <= 13.444 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area <= 119 : 1 (27.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area > 119 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 17 : 1 (5.0/2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 17 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 2.73 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 114 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 110 : 1 (6.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 110 : 2 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 114 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   wb_trans <= 38 : 1 (25.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   wb_trans > 38 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 173 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area <= 182 : 1 (3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area > 182 : 4 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 173 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 26 : 1 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 26 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 2.58 : 1 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 2.58 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.76 : 1 (30.0/2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.76 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.776 : 2 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.776 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.83 : 1 (23.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.83 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.294 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 82 : 2 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 82 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.294 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 219 : 2 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 219 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 445 : 1 (18.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 445 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand <= 465 : 2 (2.0)
```

```
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackand > 465 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.356 : 1 (6.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.356 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.325 : 1 (3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.325 : 2 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 2.73 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 3.09 : 1 (5.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 3.09 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area <= 738 : 5 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   area > 738 : 1 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.407 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.408 : 2 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.408 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 1.97 : 1 (5.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 1.97 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 2.15 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 2.03 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 2.03 : 2 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 2.15 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.456 : 1 (22.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.456 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   height > 8 : 1 (8.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   height <= 8 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght <= 6 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   lenght > 6 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.476 : 2 (3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.476 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 5 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 7 : 1 (8.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 7 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr <= 6.5 : 4 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   mean_tr > 6.5 : 2 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.526 :
|   |   |   |   |   |   |   |   |   |   |   |   |   wb_trans <= 52 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and <= 0.903 : 1 (14.0)
```

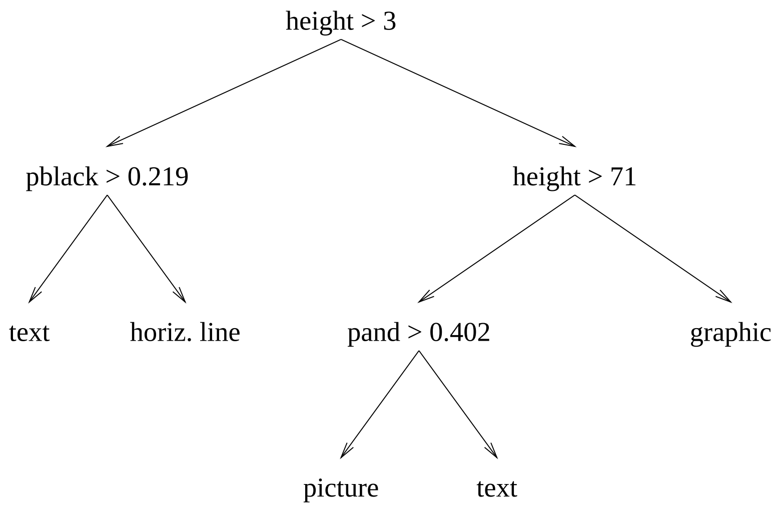
```
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_and > 0.903 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   wb_trans <= 16 : 1 (3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   wb_trans > 16 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   eccen <= 2.9 : 3 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   eccen > 2.9 : 1 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   wb_trans > 52 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   height <= 17 : 3 (3.0/2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   height > 17 : 1 (2.0)
|   |   |   |   |   |   |   |   |   p_and > 0.981 :
|   |   |   |   |   |   |   |   |   |   p_and <= 0.991 :
|   |   |   |   |   |   |   |   |   |   |   mean_tr > 2.7 : 2 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   mean_tr <= 2.7 :
|   |   |   |   |   |   |   |   |   |   |   |   wb_trans <= 19 : 5 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   wb_trans > 19 : 3 (5.0/1.0)
|   |   |   |   |   |   |   |   |   |   p_and > 0.991 :
|   |   |   |   |   |   |   |   |   |   |   blackpix <= 11 : 2 (2.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   blackpix > 11 :
|   |   |   |   |   |   |   |   |   |   |   |   blackpix <= 17 :
|   |   |   |   |   |   |   |   |   |   |   |   |   blackpix <= 14 : 1 (3.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   blackpix > 14 : 5 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   blackpix > 17 :
|   |   |   |   |   |   |   |   |   |   |   |   |   height <= 6 : 1 (9.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   height > 6 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.385 : 1 (5.0/2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.385 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black > 0.592 : 1 (3.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   p_black <= 0.592 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackpix > 45 : 1 (11.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackpix <= 45 :
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackpix <= 31 : 1 (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   blackpix > 31 : 2 (2.0)
|   |   |   |   |   |   |   |   mean_tr > 16.63 :
|   |   |   |   |   |   |   |   |   lenght <= 115 : 1 (2.0/1.0)
|   |   |   |   |   |   |   |   |   lenght > 115 : 2 (2.0/1.0)
|   |   |   |   |   |   |   height > 25 :
```

```
|   |   |   |   |   |   |   |   height <= 34 : 3 (3.0/1.0)
|   |   |   |   |   |   |   |   height > 34 : 1 (3.0)
|   |   |   |   |   |   p_black > 0.921 :
|   |   |   |   |   |   |   height <= 10 : 2 (2.0/1.0)
|   |   |   |   |   |   |   height > 10 :
|   |   |   |   |   |   |   |   mean_tr <= 17.04 : 4 (3.0/1.0)
|   |   |   |   |   |   |   |   mean_tr > 17.04 : 1 (2.0)
```

On the test data, this tree only gets 57.8% correct. It has overfit badly.

Pruning

Using a validation set, we can compare the error rate of the unpruned tree to the error rate of a tree with this node replaced by a leaf. Start at the leaves and work up the tree, making pruning decisions as we go.



The resulting tree gets 68.8% correct on the test data.

As with networks, internal validation can be performed to estimate the optimal amount of pruning. Then a tree can be grown on the entire data set and pruned to that level.

Converting Trees to Rules

Every path from the root to a leaf node defines a rule:

```
IF( (height > 3) && (height <= 71) && (pand > 0.402) )
THEN ( class = 1 )
```

However, these rules often have unnecessary tests in them. A simple greedy algorithm can be applied to find the one test that, if deleted, would improve the estimated error rate of the rule (e.g., using a validation set). Delete it and repeat as long as the estimated error rate is decreasing.

This often produces duplicate (or very similar) rules from different initial rules. So C4.5 chooses a subset of the rules sufficient to classify the training data.

Rules for Page Blocks

```
Rule 23:  height > 3 & length <= 161 & pblack <= 0.204 & pand > 0.496 &
          pand <= 0.769 & meantr > 1.28 & meantr <= 1.42  ->  class 4 [83.3%]
Rule 12:  height > 3 & height <= 71 & pand <= 0.402  ->  class 5  [72.2%]
Rule 4:   height <= 3 & pblack > 0.219 & blackpix > 7  ->  class 2 [73.0%]
Rule 18:  height <= 25 & pand > 0.53 & pand <= 0.981 & meantr <= 16.63
          ->  class 1 [80.4%]
Rule 76:  height > 3 & height <= 6 & pand > 0.991 & blackpix > 11
          ->  class 1 [84.6%]
Rule 81:  height > 6 & pblack > 0.385 & pblack <= 0.921 & pand > 0.991 &
          blackpix > 17  ->  class 1 [75.0%]
Rule 13:  area <= 189 & pand > 0.402 & pand <= 0.496  ->  class 1 [85.7%]
Default class: 1
```

These rules give 68.2% correct classifications on the test data.

The NETTalk Application

Given a spelled English word, convert it to a sequence of phonemes (and stresses) that can be played through a speech synthesizer.

$$f(\text{letter-string}) \longrightarrow \text{phoneme} + \text{stress string}$$

$$f(\texttt{"think"}) \longrightarrow \texttt{"T-IGk" + ">>1<<"}$$

Sejnowski and Rosenberg (1987) developed a dictionary of 20,003 words. The goal is to train on a subset of these words and correctly predict the pronunciation of new words.

Classifying Individual Letters

Idea: train a classifier to take as input a single letter (and the 3 letters on either side) and produce as output a phoneme and a stress:

```
"___thin"  -> "T", ">"
"__think"  -> "-", ">"
"_think_"  -> "I", "1"
"think__"  -> "G", "<"
"hink___"  -> "k", "<"
```

We encode each 7-letter “window” using 203 binary features and apply decision-tree techniques.

Our Top-Performing System

- **15-letter window**
- **Scan from back of word forwards**
- **Phonetic Context**. Input features include the classification decisions of the 7 letters to the right of the current letter.
- **Error-Correcting Output Coding**. Multiple trees are learned and then combined by a kind of voting based on error-correcting coding.

Comparison by Human Subjects

System and Configuration	% Words Judged Correct
ID3 & Error Correcting Codes	83.3
Wolpert & Error Correcting Codes	79.5
Multiclass ID3 (combined-ps)	76.2
DECtalk's rule-base	73.9

Three human subjects judging differences as "significant" between these methods and the dictionary pronunciation.

Demo

Klatt (1987) lists three major roadblocks to applying machine learning:

1. **Long distance interactions**
   "photograph" and "photography"
2. **Letter pairs like CH**
   misalign letters in the window
3. **Compound words**
   houseboat

Long Distance Effects

photograph
photography

commend
commendation
commendatory

chromatogram
chromatograph
chromatography

psyche
psychiatric
psychiatrist

The letter-pair CH

chocolate

purchasable

cliche

cytochemistry

dachshund

disaccharide

chlorophyll

chaise

synchrotron

	DECtalk	
	OK	Bad
ID3 OK	490	120
Bad	43	74

Compound Words

house + boat = houseboat

head + start = headstart

bird + watch = birdwatch

nut + hatch = nuthatch

steeple + chase = steeplechase

Machine Learning Summary

- **Use training data to choose among possible hypotheses**. Only works if the hypothesis space contains good hypotheses.

- **Choosing good features is critical**.

- **More data gives better performance**.

- **Overfitting must be controlled**. Early stopping, tree pruning.

- **Optimal search is intractable**. Heuristic methods must be used.