CSC2515 Fall 2007
Introduction to Machine Learning

Lecture 7: Decision Trees and Mixtures of Experts

All lecture slides will be available as .ppt, .ps, & .htm at
www.cs.toronto.edu/~hinton

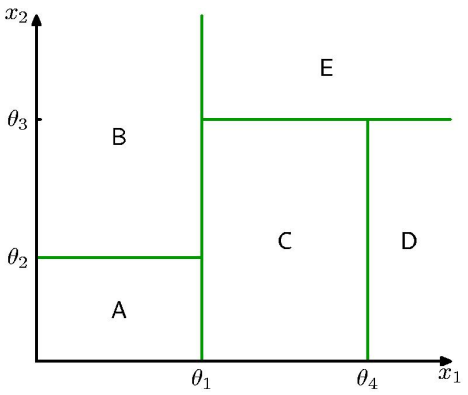
Many of the figures are provided by Chris Bishop
from his textbook: "Pattern Recognition and Machine Learning"

## Decision Trees: Non-linear regression or classification with very little computation

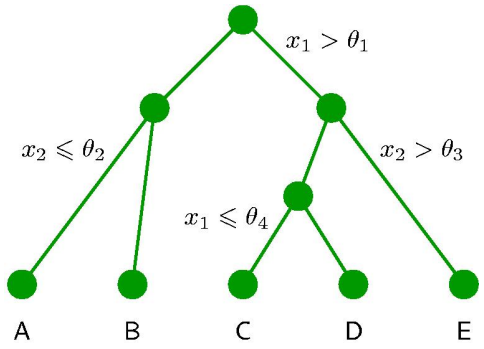
- The idea is to divide up the input space into a disjoint set of regions and to use a very simple estimator of the output for each region
  - For regression, the predicted output is just the mean of the training data in that region.
    - But we could fit a linear function in each region
  - For classification the predicted class is just the most frequent class in the training data in that region.
    - We could estimate class probabilities by the frequencies in the training data in that region.

## A very fast way to decide if a datapoint lies in a region

- We make the decision boundaries orthogonal to one axis of the space and parallel to all the other axes.
  - This is easy to illustrate in a 2-D space
- Then we can locate the region that a datapoint lies in using a number of very simple tests that is logarithmic in the number of regions.




## An axis-aligned decision tree



## How do we decide what tests to use?

- There are many variations (CART, ID3, C4.5) but the basic idea is to recursively partition the space by greedily picking the best test from a fixed set of possible tests at each step.
  - We need to decide on the set of possible tests
    - Consider splits at every coordinate of every point in the training data (i.e. all axis-aligned hyper-planes that touch datapoints)
    - We could consider all possible hyper-planes but it is usually too expensive in fitting time and model complexity.
  - We need a measure of how good a test is ("purity")
    - For regression, compute the resulting sum-squared error over all partitions.
    - For classification it is a bit more complicated.

## Two measures of classification "impurity"

- Treat the frequencies of classes in each partition as probabilities and compute the entropy.
  - This is a natural measure if we want to maximize the log probability of the correct answer.
- Treat the frequencies as probabilities and use an unprincipled measure called the Gini index.
  - It was invented by frequentist statisticians and it just happens to work pretty well.

$$\sum_{i=1}^{i=K} p_i(1-p_i)$$

## When should we stop adding nodes?

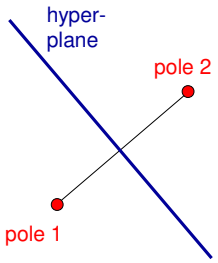
- Sometimes, the error stays constant for a while as nodes are added and then it falls
  - So we cannot stop adding nodes as soon as the error stops falling.
- It typically works best to fit a tree that is too large and then prune back the least useful nodes to balance complexity against error
  - We could use a validation set to do the pruning.

## Advantages and disadvantages of decision trees

- They are easy to fit, easy to use, and easy to interpret as a fixed sequence of simple tests. (Doctors like them.)
- They are non-linear, so they work much better than linear models for highly non-linear functions.
- They typically generalize less well than non-linear models that use adaptive basis functions, but its easy to improve them by averaging the predictions of many trees
  - Each tree is fitted to a training set produced by sampling the dataset with replacement ("Bagging")
  - So much for interpretability!

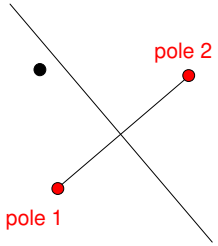
## An alternative to axis-aligned hyper-planes

- We want a fixed set of hyper-planes that is more flexible than axis-aligned, but not nearly as complex as arbitrary hyper-planes.
- Consider the N(N-1)/2 hyper-planes that are equidistant from two of the N training examples.
  - These are quite sensible candidates for decision boundaries.
  - It is very easy to compute on which side of one of these hyper-planes any of the training points lies




## Computing on which side a training point lies

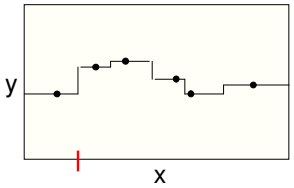
- First compute all the pairwise distances between training points.
- Then, just look up the distance from a training point to each of the two "poles" of the hyper-plane.
  - This is very efficient for high-dimensional spaces.
- The computational efficiency and the low complexity both come from the fact that we define 0(N^2) hyper-planes using only N points.




## A spectrum of models

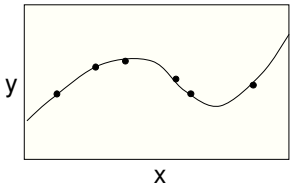
### Very local models

- e.g. Nearest neighbors
- Very fast to fit
  - Just store training cases
- Local smoothing obviously improves things




### Fully global models

- e. g. Polynomial
- May be slow to fit
  - Each parameter depends on all the data

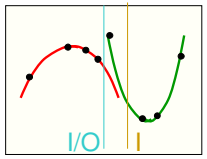



## Multiple local models

- Instead of using a single global model or lots of very local models, use several models of intermediate complexity.
  - Good if the dataset contains several different regimes which have different relationships between input and output.
  - But how do we partition the dataset into subsets for each expert?

## Partitioning based on input alone versus partitioning based on input-output relationship

- We need to cluster the training cases into subsets, one for each local model.
  - The aim of the clustering is NOT to find clusters of similar input vectors.
  - We want each cluster to have a relationship between input and output that can be well-modeled by one local model

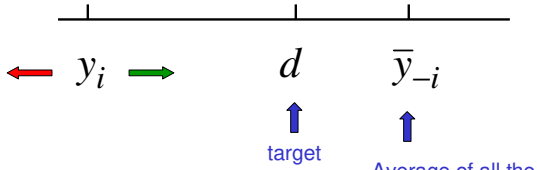



which partition is best:
I=input alone or I/O=input→output mapping?

## Mixtures of Experts

- Can we do better that just averaging predictors in a way that does not depend on the particular training case?
  - Maybe we can look at the input data for a particular case to help us decide which model to rely on.
    - This may allow particular models to specialize in a subset of the training cases. They do not learn on cases for which they are not picked. So they can ignore stuff they are not good at modeling.
- The key idea is to make each expert focus on predicting the right answer for the cases where it is already doing better than the other experts.
  - This causes specialization.
  - If we always average all the predictors, each model is trying to compensate for the combined error made by all the other models.

## A picture of why averaging is bad




Do we really want to move the output of predictor i away from the target value?

## Making an error function that encourages specialization instead of cooperation

- If we want to encourage cooperation, we compare the average of all the predictors with the target and train to reduce the discrepancy.
  - This can overfit badly. It makes the model much more powerful than training each predictor separately.

Average of all the predictors

$$E = (d - < y_i >_i)^2$$

- If we want to encourage specialization we compare each predictor separately with the target and train to reduce the average of all these discrepancies.
  - Its best to use a weighted average, where the weights, p, are the probabilities of picking that "expert" for the particular training case.

$$E = < p_i (d - y_i)^2 >_i$$

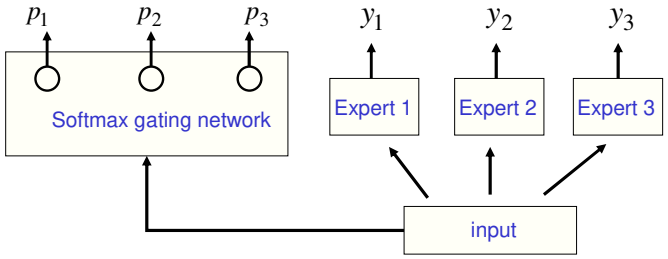
probability of picking expert i for this case

## The mixture of experts architecture

Combined predictor: $y = \sum_i p_i \, y_i$

Simple error function for training: $E = \sum_i p_i (d - y_i)^2$

(There is a better error function)




## The derivatives of the simple cost function

- If we differentiate w.r.t. the outputs of the experts we get a signal for training each expert.
- If we differentiate w.r.t. the outputs of the gating network we get a signal for training the gating net.
  - We want to raise p for all experts that give less than the average squared error of all the experts (weighted by p)

$$E = \sum_i p_i (d - y_i)^2$$

$$\frac{\partial E}{\partial y_i} = p_i (d - y_i)$$

$$\frac{\partial E}{\partial x_i} = p_i [(d - y_i)^2 - E]$$

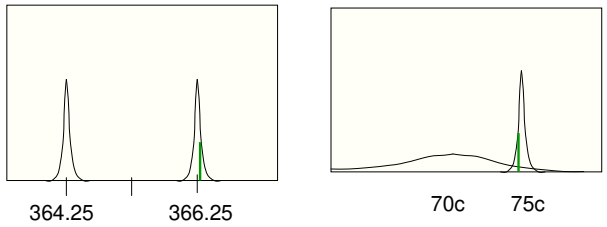
$$where \quad p_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

## Another view of mixtures of experts

- One way to combine the outputs of the experts is to take a weighted average, using the gating network to decide how much weight to place on each expert.
- But there is another way to combine the experts.
  - How many times does the earth rotate around its axis each year?
  - What will be the exchange rate of the Canadian dollar the day after the Quebec referendum?

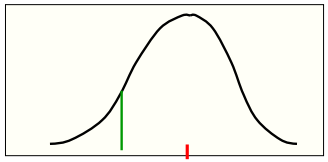
## Giving a whole distribution as output

- If there are several possible regimes and we are not sure which one we are in, its better to output a whole distribution.
  - Error is negative log probability of right answer



## The probability distribution that is implicitly assumed when using squared error

- Minimizing the squared residuals is equivalent to maximizing the log probability of the correct answers under a Gaussian centered at the model's guess.
  - If we assume that the variance of the Gaussian is the same for all cases, its value does not matter.




$$p(d) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(d-y)^2}{2\sigma^2}}$$

$$-\log p(d) = k + \frac{(d-y)^2}{2\sigma^2}$$

## The probability of the correct answer under a mixture of Gaussians

Mixing proportion assigned to expert i for case c by the gating network

$$p(d^c \mid MoG) = \sum_i p_i^c \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\|d^c - o_i^c\|^2}$$

Prob. of desired output on case c given the mixture

Normalization term for a Gaussian with $\sigma^2 = 1$

output of expert i

## A natural error measure for a Mixture of Experts

$$-\log p(d^c \mid MoE) = -\log \sum_i p_i^c \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\|d^c - o_i^c\|^2}$$

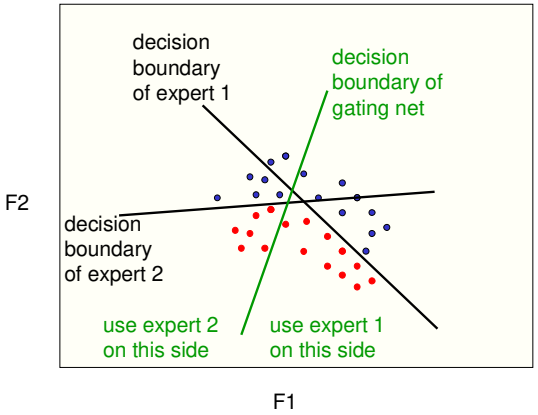
$$\frac{\partial E^c}{\partial o_i^c} = -2 \frac{p_i^c e^{-\frac{1}{2}\|d^c - o_i^c\|^2}}{\sum_j p_j^c e^{-\frac{1}{2}\|d^c - o_j^c\|^2}} (d^c - o_i^c)$$

This fraction is the posterior probability of expert i

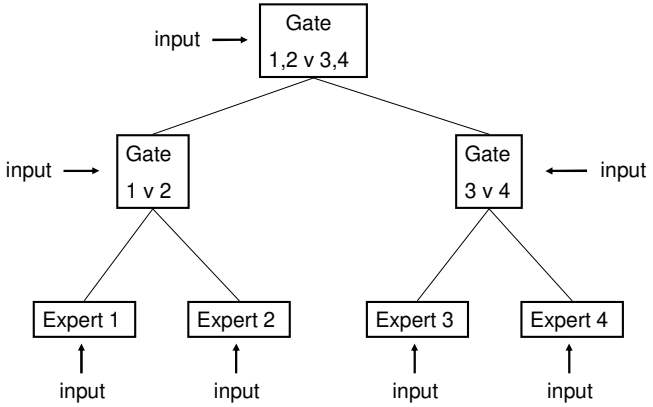
## What are vowels?

- The vocal tract has about four resonant frequencies which are called formants.
  - We can vary the frequencies of the four formants.
- How do we hear the formants?
  - The larynx makes clicks. We hear the dying resonances of each click.
  - The click rate is the pitch of the voice. It is independent of the formants. The relative energies in each harmonic of the pitch define the envelope of a formant.
- Each vowel corresponds to a different region in the plane defined by the first two formants, F1 and F2. Diphthongs are different.

## A picture of two imaginary vowels and a mixture of two linear experts after learning




## Hierarchical mixtures of experts




## The generative model for an HMoE

- First let the top-level gate choose one branch of the gating tree, using the input to determine the relative probabilities
- Then use the next gating network down in the chosen branch, etc.
- Finally, generate an output from the expert at the chosen leaf node.

## Making predictions once the tree has been learned

- If we are doing regression and our loss function is the squared error, then average the outputs of all the experts using the probabilities of paths through the gating tree as the weights.
- This avoids discontinuities at boundaries between regions, because the probabilities are soft.
  - Its like the use of sigmoid units to allow gradient descent in training feed-forward nets

## Learning a simplified HMoE

- There is a very efficient way to learn an HMoE if we make two assumptions:
- Linear experts: make every expert give an output that is a linear function of the input, and use a squared error.
  - This makes it possible to fit an expert non-iteratively if we know how much responsibility it has for each training case.
- Generalized linear gating networks: make each expert be a softmax applied to a linear transformation of the input vector.
  - This makes it possible to fit each gating network quickly if we know what probabilities it should output for each case. The cost function is convex.
  - The fitting uses IRLS – iterative recursive least squares.

## Using EM to fit an HMoE

- E-step: Compute the output of each expert and the "prior" probabilities provided by each gating net. Then combine the prior probabilities with the probability that each expert assigns to the correct answer. This gives posterior probabilities for each expert and each gating net.
  - (see the Jordan and Jacobs paper for the math)
- M-step: Refit each expert to the data weighted by the posterior probability that each datapoint came from that expert. Refit each gating network to minimize the cross-entropy between the "prior" that it provides and the posterior distribution computed in the E-step.
  - This requires IRLS which is iterative but converges rapidly to the global optimum (of this sub-problem).
  - See the textbook page 207 for IRLS

## IRLS

- For a linear model with a squared error, the optimal weights are given by

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$$

- This can be derived as a single update on the initial weight vector in which the gradient vector is pre-multiplied by the inverse of the curvature of the error surface to decide the direction and magnitude of the weight update:

$$\mathbf{w}^{new} = \mathbf{w}^{old} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

## Newton updates for a logistic output with cross-entropy error

- Using the same Newton-Raphson method of pre-multiplying the gradient by the inverse of the curvature, we get:

$$\mathbf{w}^{new} = (\mathbf{X}^T\mathbf{R}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{R}(\mathbf{X}\mathbf{w}^{old} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})$$

*where* $\mathbf{R}$ *is a diagonal matrix with*

$R_{nn} = y_n(1-y_n)$

*i.e. the slope of the sigmoid*

## Is an HMoE better than a flat MoE?

- If we use the simple gating networks that can be fitted rapidly by IRLS, is an HMoE really more powerful than a flat MoE?
  - The textbook says it is but doesn't say why.
  - An HMoE that uses a binary tree has the same number of degrees of freedom in the path probabilities as a single flat softmax over all experts.
  - But does the dependence on the input vector make the two ways of doing the gating different?

## A different (and better?) type of hierarchy for a mixture of experts

- Instead of just using a hierarchy of gating nets, also use a hierarchy of experts.
- Learn the whole system by greedy divide-and-conquer.
  - Start by learning a single expert.
  - Then make two slightly different copies of the expert, and use EM to rapidly fit an MoE with one gating network and two experts.
  - Now split each of these two experts. Use the previous gating network as the initial top-level gating net and add two new gating nets (with zero weights) at the next level down.

## The advantage of "speciation"

- The knowledge that is shared by different experts does not need to be learned separately by the different experts as it does in an HMoE.
  - Think how inefficient it would be for humans and chimpanzees to separately invent eyeballs.

## Does speciation work better than a standard HMoE?

- Even though the speciation algorithm was invented (by Hinton and Nowlan) before HMoE's it has never been compared with HMoE's (so far as I know).
  - Maybe it works better.