Learning

1 Neural nets

Every input to a neron in the neural net has an input value $x_1 \in 0, 1$ which is multipled by a weight w_i.
We sum all the input weights, and if reach a certain degree, we fire the neron and output its value.
We are modeling:

1. All or none
2. cumulative influence
3. Sympatic weight

A neural net is a funciton approximation. $z = f(x, w, T)$ where x is in input, w are the weights, and T are the neruons.
For activation, we use the sigmoid function $\frac{1}{1+e^{-}\alpha}$ to see if we are zero or one and what we use for the next input.
Performance is measured by $\frac{1}{2} * (expected - given)^2$
Computation is linear in depth, quadratic in width.

Auto Coding: is a process of using a hidden layer that's a fraction the size of the input, and trying to get the input ot match the output. The effect of this is that the hidden layer finds an encoding of a generaliztion of the input.
soft max: To find the probability of a particular output in the function, the probability is $P(c_1) = \frac{f(c_1)}{\sum_x f(c_i)}$

2 Genetic algorithms

Every chromosone is defined by an x and y value.
Mutation changes the x and y values slightly.
Crossing over combines x1 and y2 and y1 and x2.
Every generation we are going to: breed, map from genotype to phenotype, determine fitness, find a survivor
We could use the rank space method, and first each indivudal have a probability they reach the next generation, and move down until someone reaches. In addition to using rank space, we could also consider how different each one is from the people that have already been selected.

3 Near misses, felicity conditions:

When identifying symbols, we can keep track of examples to see what's close to an arch.
This keeps track of what additional information changes the classification and what information doesn't.
Heuristics we can do to keep track of things:

1. Drop link: decide something doesn't matter, have it point to the "anything" marker, generlization and example
2. Extend set: extend set of allowable values, generalization and this is an example
3. require link: decide something is require, near miss and specilizatin
4. forbidden link: decide that something can't matter, near miss and specilization
5. clinb tree: climb up tree of categories one step, generalization and example (correct value)

You can't learn unless you almost already know the answer.
There's a lot of value in talking to yourself.

4 support vector machines

Framework: collection of positive and negative examples, divide them with a straight line.
We draw the line that makes the widest "street" between positive and negative examples.
Draw a line that's perpendicular to the stree, measure the dot product between that line and the line to an unknow.
If $u \cdot v + b \geq 0$ then it's a positive example
$wx_+ + b \geq 1$ and $w \cdot x_- + b \leq -1$ for any positive and negative example x
Introduce y_i such that it's +1 for positive samples and -1 for negative examples. Now we have that $y_i(x_i \cdot *w_i + b) - 10$. This become equal to zero for all examples on edges of street.
Width of street = $(x_+ - x_-) * fracw||w||$, and we can use some algebra to find out that the width of the street is $\frac{2}{\|w\|}$ so we want to maximize the magnitude of w. Instead, we can minimize $\frac{1}{2}\|w\|^2$

When we can't solve a problem, we can use a kernal to transform it into another space.

5 boosting

Idea: combine several weak classifiers and combine their results to have a strong classifier.
First run the algorithm and find the one that gives the loweest error, call that h_1. Afterwards, weight everything that h_1 got wrong highly, then call that h_2.
We can use decision tree stumps to generate more tests.
The error is the $\sum_w rongw_i$ note: $\sum w_i = 1$.
20 minutes