

Lecture 19: Neural Networks

CS105: Great Insights in Computer Science
Michael L. Littman, Fall 2006

More Than A Program

- Usually, we think a program is something written by an experienced person.
- Often, the program isn't complete without "experience" of its own.
- Example: If I had to write a compression program, I could do ok. But, using the character-count statistics from the Gettysburg Address makes it possible to produce the most compact code.

Machine Learning

- *Machine learning* is the idea of writing programs that use data (experience) to create better programs than people can write directly.
- *Supervised learning* is a specific problem in which the experience is in the form of labeled examples and the learning system needs to learn to label novel examples.

Many Names

- classification learning
- statistical learning
- data mining
- concept learning
- discriminative learning
- logistic regression
- pattern recognition

Classification: Idea

- In its simplest form, a learner's job is to produce a *classifier*.
- A classifier takes objects as input and assigns each one to a *class*.
- Most simply, objects are represented as *vectors of features* and classes are 0/1.

Future Killer?

The image is a screenshot of the Slashdot website. At the top, there is a navigation bar with links to various sites: OSTG, SourceForge, ThinkGeek, ITMJ, Linux.com, NewsForge, freshmeat, Jobs, and PriceGrabber. Below this is the Slashdot logo and the tagline "NEWS FOR NERDS. STUFF THAT MATTERS." There are also links for "Login", "Create Account", "Subscribe", "Why Login?", and "Wh".

The main content area features a sidebar on the left with a "Sections" menu containing links to Main, Apple, AskSlashdot, Backslash, Books, Developers, Games, Hardware, Interviews, IT, Linux, and Politics. The main article is titled "Software Used To Predict Who Might Kill" and is posted by kdawson on Monday December 04, @02:11AM from the three-psychics-in-a-pool dept. The article text begins with "eldavojohn writes" and quotes "Richard Berk, a University of Pennsylvania criminologist, has worked with authorities to develop a software". To the right of the article is a yellow box with the text "Uses ethan" and a login form with fields for "Nickname", "Password", and "Public Terminal", along with a "Log in" button.

Example Classifier

- Input: A high school student
- Output: Will the student drop out of college?
- Vector of Features: Score on SATs, grades in Math / English / Science, age, parent's income, years at current address, height
- Such a classifier *might* be useful as a tool for admissions or financial aid.

Learning: The Problem

- Input: A *training set* consisting of *labeled instances*, each of which is a feature vector and a desired class (1 = yes, 0 = no).
- Output: A classifier, which we hope will accurately assign new feature vectors to classes.
- A *learning algorithm* is a program that addresses this problem.

Wearing White

- My mom told me “never wear white shoes after labor day”.
- Well, she didn’t so much *tell* me as humiliate me if I violated the rule.
- Top part is training set.
- Bottom lines are instances to be classified.

month	day	ok?
2	14	1
7	4	1
8	30	1
9	4	1
9	5	0
10	31	0
12	25	0
10	9	?
3	30	?

Informal to Formal

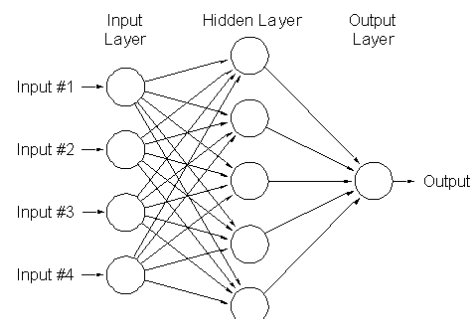
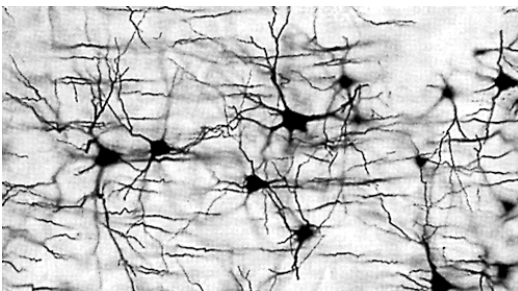
- Note that problem is not really defined at this point! Whether it does well on new examples isn’t directly measurable.
- One way to make it formal: find classifier with minimum number of mistakes on the training set.
- Usually not enough to create good classifiers: *overfitting*.

Learning Algorithms

- Decision trees
- Boosting
- Nearest neighbors
- Support Vector Machines
- Neural networks
- Naive Bayesian classifiers
- etc.

Neural Networks

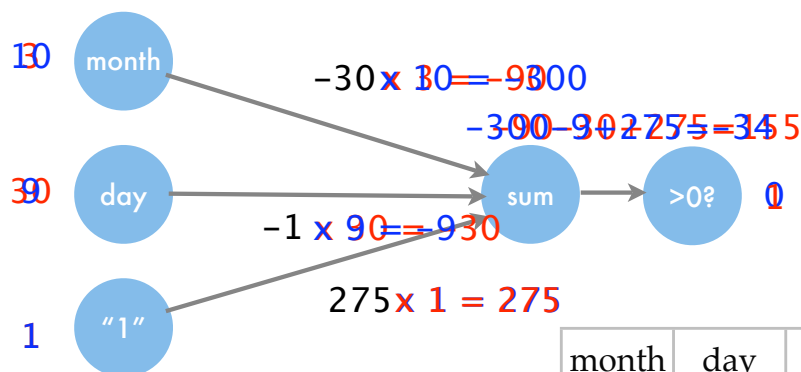
- Background: http://en.wikipedia.org/wiki/Artificial_neural_network
- Inspired by brains, but ultimately a simple and flexible mathematical representation.



Basic Elements

- Each component of the feature vector becomes an *input unit*.
- An additional input unit is added that always has an input of 1.
- Each input unit is connected to a sum unit with a *weight*. The input value is multiplied by the weight and all are summed.
- If the sum is greater than zero, the output is "1", else "0".

White Net: Handmade



month	day	ok?
10	9	0
3	30	1

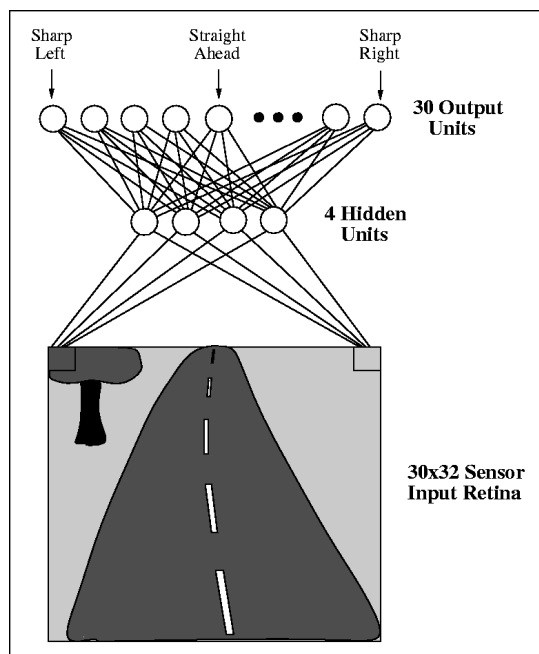
In "Math"

- i stands for a feature.
- x_i is the input on feature i (the i th component of the feature vector x).
- w_i is the weight for feature i .
- y is the output of the network.
 - $y = (\sum_i x_i \times w_i) > 0$

```
def classify(w,x):  
    return sum([x[i] * w[i] for i in range(len(x))]) > 0
```

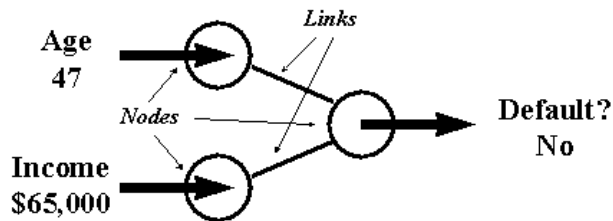
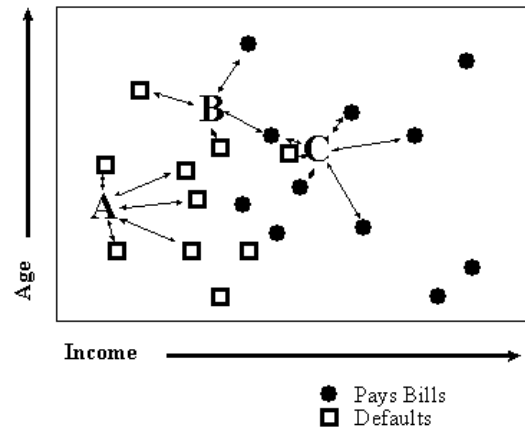
Autonomous Driving

- ALVINN could mimic a driver's decisions and, so, follow roads.



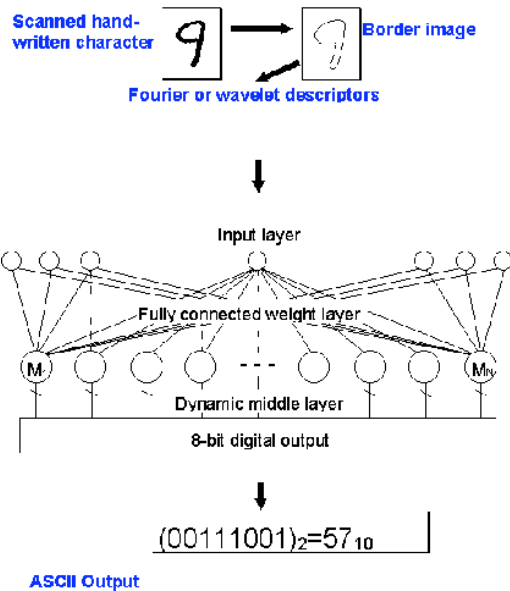
Loan Applications

- Will loan applicant default on the loan?



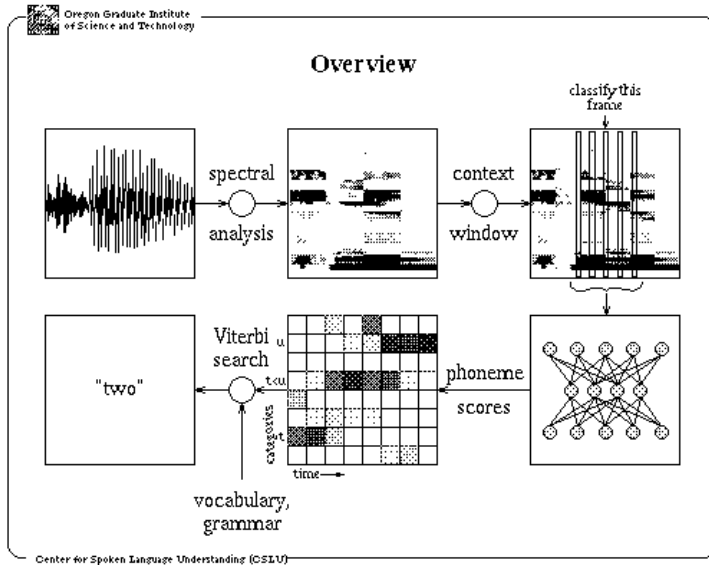
Character Recognition

- Is it a "9" I see?
- NN systems in place reading roughly half of all checks in US.



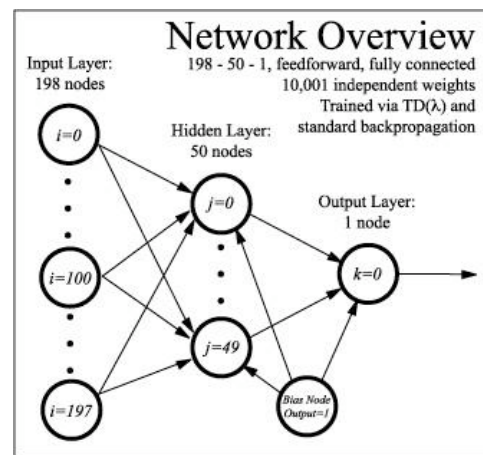
Speech Recognition

- Learning systems (sometimes NNs) play an important role in high-quality speech recognizers.



Game Playing

- One of the first uses was in a checker playing program (learn the evaluation function).
- Central part of the world's best backgammon playing program.



Training A Perceptron

- Given a training set, would like a way to set the weights to correctly classify the labeled instances.
- Networks with one layer of nets are very well understood at this point.
- I'll describe a demonstrate one simple rule: the perceptron training procedure.
 - http://en.wikipedia.org/wiki/Delta_rule
 - <http://en.wikipedia.org/wiki/Perceptron>

Start With "Math"

- i is a feature.
- x is the input (x_i is the i th component).
- w_i is weight vector.
- $y = (\sum_i x_i \times w_i) > 0$.
- t is the *target* (right output for x).
- α is the *learning rate* (amount to change the weights).
- Δw_i is the amount we plan to change weight i .
- For all features i :
 - $\Delta w_i = \alpha(t - y)x_i$

Understanding The Rule

t	y	action of $\Delta w_i = \alpha(t - y)x_i$
0	0	$t - y$ is zero (right answer). Nothing needs to change.
0	1	$t - y$ is -1 (output too big). If $x_i > 0$, w_i is decreased to make sum smaller.
1	0	$t - y$ is 1 (output too small). If $x_i > 0$, w_i is increased to make sum bigger.
1	1	$t - y$ is zero (right answer). Nothing needs to change.

Using The Rule

- Start off with the weight vector set to an arbitrary value (all zeros, say).
- For each example in the training set, apply the rule, changing the weights as needed.
 - If weights have changed, repeat.

Simple Example

```
def train(d):
    w = [-1 for i in d[0]]
    wNew = [0 for i in d[0]]
    while w != wNew:
        w = wNew
        print errorSet(d,w)
        wNew = perceptronRule(d,w)
    return w

def perceptronRule(d,w):
    for l in d:
        n = len(l)
        t = l[n-1]
        x = l[0:n-1]+[1]
        y = classify(w,x)
        w = [w[i]+(t-y)*x[i] for i in range(n)]
    return w

orSet = [[0,0,0],[0,1,1],[1,0,1],[1,1,1]]
w = train(orSet)
3
1
1
0
print w
[1, 1, 0]
```

Rule Properties

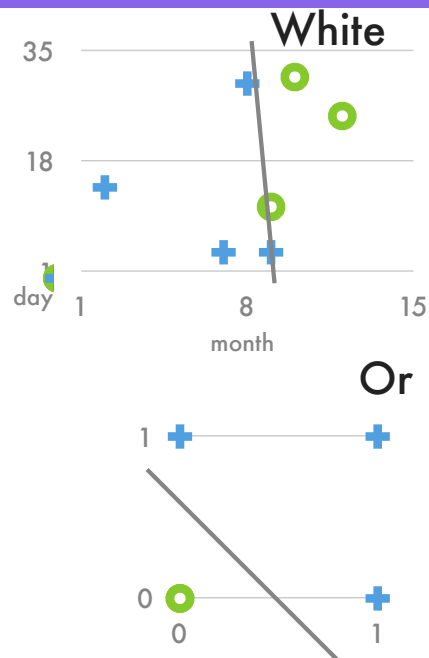
- Stops changing when stops making mistakes.
- If there is a way of setting the weights that makes *no* mistakes, it will be found eventually.
- Otherwise, might bounce around.
- Very related to gradient descent and hill-climbing: local changes to reduce error score.

More Examples

- Run code on “orSet”, “notASet”, “bDaySet”, “whiteSet”, “hotSet”.
- What about “comfortableSet”, “virgoSet”, and “xorSet”?

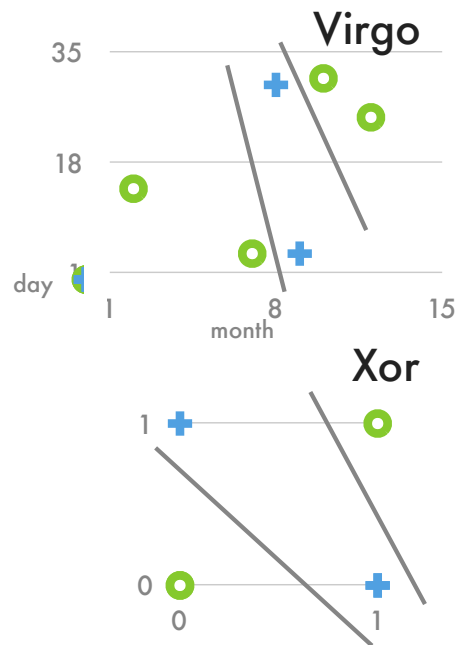
Linear Separability

- Perceptron can only achieve perfect score if data is *linearly separable*. That is, if we visualize the instances as points in a high dimensional space, there needs to be linear surface that separates the positive and negative examples.



Not Linearly Separable

- Minsky and Papert pointed out that data sets that are not linearly separable can cause the perceptron much headaches.



Elaborations

- Backprop can train multilayer networks and learn xor.
- Some work on recurrent networks (outputs sent back as inputs).
- Lots of statistically grounded algorithms now available.

Decision Trees

- Decision trees are another kind of classifier.
- Branch depending on test on attribute value.
- Can build such a tree incrementally and automatically using examples. DEMO.

