What is Machine Learning?

Based on www.cs.toronto.edu/~hinton

What is Machine Learning?

- It is very hard to write programs that solve problems like recognizing a face.
 - We don't know what program to write because we don't know how our brain does it.
 - Even if we had a good idea about how to do it, the program might be horrendously complicated.
- Instead of writing a program by hand, we collect lots of examples that specify the correct output for a given input.
- A machine learning algorithm then takes these examples and produces a program that does the job.
 - The program produced by the learning algorithm may look very different from a typical hand-written program. It may contain millions of numbers.
 - If we do it right, the program works for new cases as well as the ones we trained it on.

A classic example of a task that requires machine learning: It is very hard to say what makes a 2

00011(1112

るえてみる 2 ス 3 み 3 344445555 467773888 888194999

Some more examples of tasks that are best solved by using a learning algorithm

- Recognizing patterns (classification):
 - Facial identities or facial expressions
 - Handwritten or spoken words
 - Medical images
- Prediction (regression):
 - Future stock prices or currency exchange rates

Some web-based examples of machine learning

- The web contains a lot of data. Tasks with very big datasets often use machine learning
 - especially if the data is noisy or non-stationary.
- Spam filtering, fraud detection:
 - The enemy adapts so we must adapt too.
- Recommendation systems:
 - Lots of noisy data. Million dollar prize!
- Information retrieval:
 - Find documents or images with similar content.
- Data Visualization:
 - Display a huge database in a revealing way

Displaying the structure of a set of documents using Latent Semantic Analysis (a form of PCA)



Each document is converted to a vector of word counts. This vector is then mapped to two coordinates and displayed as a colored dot. The colors represent the hand-labeled classes.

When the documents are laid out in 2-D, the classes are not used. So we can judge how good the algorithm is by seeing if the classes are separated.

Displaying the structure of a set of documents using a deep neural network



Types of learning

- Supervised learning (input examples and correct answers are given)
 - Learn to predict output when given an input vector
 - Who provides the correct answer?
- Reinforcement learning
 - Learn action to maximize payoff (reinforcement signal)
 - Not much information in a payoff signal
 - Payoff is often delayed
- Unsupervised learning (only input examples, no correct answers)
 - Discover/create an internal representation of the input e.g. form clusters; extract features
 - How do we know if a representation is good?
 - Clustering, density estimation, dimensionality reduction
 - This is the new frontier of machine learning because most big datasets do not come with labels.

Machine Learning & Statistics

- A lot of machine learning is just a rediscovery of things that statisticians already knew. This is often disguised by differences in terminology:
 - Ridge regression = weight-decay
 - Fitting = learning
 - Held-out data = test data
- But the emphasis is very different:
 - A good piece of statistics: Clever proof that a relatively simple estimation procedure is asymptotically unbiased.
 - A good piece of machine learning: Demonstration that a complicated algorithm produces impressive results on a specific task.
- Data-mining: Using very simple machine learning techniques on very large databases because computers are too slow to do anything more interesting with ten billion examples.

Hypothesis Space

- One way to think about a supervised learning machine is as a device that explores a "hypothesis space".
 - Each setting of the parameters in the machine is a different hypothesis about the function that maps input vectors to output vectors.
- The art of supervised machine learning is in:
 - Deciding how to represent the inputs and outputs
 - Selecting a hypothesis space that is powerful enough to represent the relationship between inputs and outputs but simple enough to be searched.

Searching a hypothesis space

- The obvious method is to first formulate a loss (error) function and then adjust the parameters to minimize the loss function.
 - This allows the optimization to be separated from the objective function that is being optimized.
- Bayesians do not search for a single set of parameter values that do well on the loss function.
 - They start with a prior distribution over parameter values and use the training data to compute a posterior distribution over the whole hypothesis space.

Some Loss Functions

- Squared difference between actual and target realvalued outputs.
- Number of classification errors
 - Problematic for optimization because the derivative is not smooth.
- Negative log probability (likelihood) assigned to the correct answer.
 - In some cases it is the same as squared error (regression with Gaussian output noise)
 - In other cases it is very different (classification with discrete classes needs cross-entropy error)

Generalization

- The real aim of supervised learning is to do well on test data that is not known during learning.
- Choosing the values for the parameters that minimize the loss function on the training data is not necessarily the best policy.
- We want the learning machine to model the true regularities in the data and to ignore the noise in the data.
 - But the learning machine does not know which regularities are real and which are accidental quirks of the particular set of training examples we happen to pick.
- So how can we be sure that the machine will generalize correctly to new data?

Trading off the goodness of fit against the complexity of the model

- It is intuitively obvious that you can only expect a model to generalize well if it explains the data surprisingly well given the complexity of the model.
- If the model has as many degrees of freedom as the data, it can fit the data perfectly but so what?
- There is a lot of theory about how to measure the model complexity and how to control it to optimize generalization.

A simple example: Fitting a polynomial

- The green curve is the true function (which is not a polynomial)
- The data points are uniform in x but have noise in y.
- We will use a loss function that measures the squared error in the prediction of y(x) from x. The loss for the red polynomial is the sum of the squared vertical errors.



Some fits to the data: which is best?









Using a validation set

- Divide the total dataset into three subsets:
 - Training data is used for learning the parameters of the model.
 - Validation data is not used of learning but is used for deciding what type of model and what amount of regularization works best.
 - Test data is used to get a final, unbiased estimate of how well the network works. We expect this estimate to be worse than on the validation data.
- We could then re-divide the total dataset to get another unbiased estimate of the true error rate (cross-validation).

The Bayesian framework

- The Bayesian framework assumes that we always have a prior distribution for everything.
 - The prior may be very vague.
 - The likelihood term takes into account how probable the observed data is given the parameters of the model.
 - When we see some data, we combine our prior distribution with a likelihood term to get a posterior distribution.

Bayes Theorem

joint probability p(D) p(W | D) = p(D, W) = p(W) p(D | W)

Prior probability of weight vector W $p(W | D) = \frac{p(W) \quad p(D | W)}{p(D)}$ Posterior probability of weight vector W given training data D $p(W) p(D | W) = \int_{W} p(W) p(D | W)$

A cheap trick to avoid computing the posterior probabilities of all weight vectors

- Suppose we just try to find the most probable weight vector.
 - We can do this by starting with a random weight vector and then adjusting it in the direction that improves p(W | D).
- It is easier to work in the log domain. If we want to minimize a cost we use negative log probabilities:

 $p(W | D) = p(W) \quad p(D | W) / p(D)$ Cost = -log p(W | D) = -log p(W) - log p(D | W) + log p(D)

Why we maximize sums of log probs

- We want to maximize the product of the probabilities of the outputs on the training cases
 - Assume the different training cases, c, are independent.

$$p(D|W) = \prod_{c} p(d_{c}|W)$$

 Because the log function is monotonic, it does not change where the maxima are. So we can maximize sums of log probabilities

$$\log p(D|W) = \sum \log p(d_c|W)$$

MAP & ML

- Suppose we completely ignore the prior over weight vectors
 - This is equivalent to giving all possible weight vectors the same prior probability density.
- Then all we have to do is to maximize:

$$\log p(D | W) = \sum \log p(D_c | W)$$

• Maximum likelihood (ML) learning.

 $\log p(W | D) = \log p(W) + \log p(D | W)$

 This is called maximum a posteriori (MAP) learning (prior is taken into account)