Basic Machine Learning

What is Machine Learning?

- Arthur Samuel (1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998) Well-posed Learning Problem: A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Example Applications

- Spam Filters
- Handwriting and speech recognition
- Product Recommendations (Netflix, Amazon, etc.)
- Computer Vision
- Data mining of large databases in fields like biology, medicine, engineering, etc.

Types of Machine Learning

- Thee Major Types
  - Supervised Learning: Algorithm is trained using data where the desired output for a given input is known (labeled data).
  - Unsupervised Learning: Algorithm is trained on unlabeled data and left to determine structure on its own.
  - Reinforced Learning: Algorithm is trained by interaction with dynamic environment.

Supervised Learning

- Arguable the simplest (relatively) paradigm and the focus of this presentation.
- Two major uses:
  - Regression: Measure relationships between variables, eg. house price vs square footage.
  - Classification: Assign a class to an input, eg sentiment analysis or determining digits of zip code on a letter.

Regression

- We want to find a line that fits out data well (without over fitting) to give us predictive power.
- I'm assuming some basic familiarity with the technique outside of its use in ML.
  - http://en.wikipedia.org/wiki/Linear_regression
  - Least Squares Estimation in particular.

(Non)linear Regression

- Least Squares Estimation is often used
  - May remember $w = (X^T X)^{-1} X^T y$
  - If X is a large matrix, ie a database containing millions of rows, then the matrix inversion can be intractable.
  - Solution: Use Machine Learning

(Non)linear Regression

- We start with some training data vector
  - $D = \{D_1, D_2, \ldots, D_n\}$ where each example D_n is an input/output pair $<\mathbf{x}_i, y_i>$.
  - $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$ is an input vector with d features.
    - Features for a house may be square footage, number of bedrooms, etc.
  - y_i is the known output paired with some $\mathbf{x}_i$.
    - House price

(Non)linear Regression

- We want to optimize a hypothesis function h with output z.
  - $h(x) = w_0 + w_1x_1 + w_2x_2 \ldots + w_dx_d$
  - w_i is a weight
- As the algorithm is shown examples it will produce a predicted output z which is compared to the expected output y. Our goal is to fine weights that minimize the error $(y - h(x))^2$.
  - Should look familiar (Least Squares)

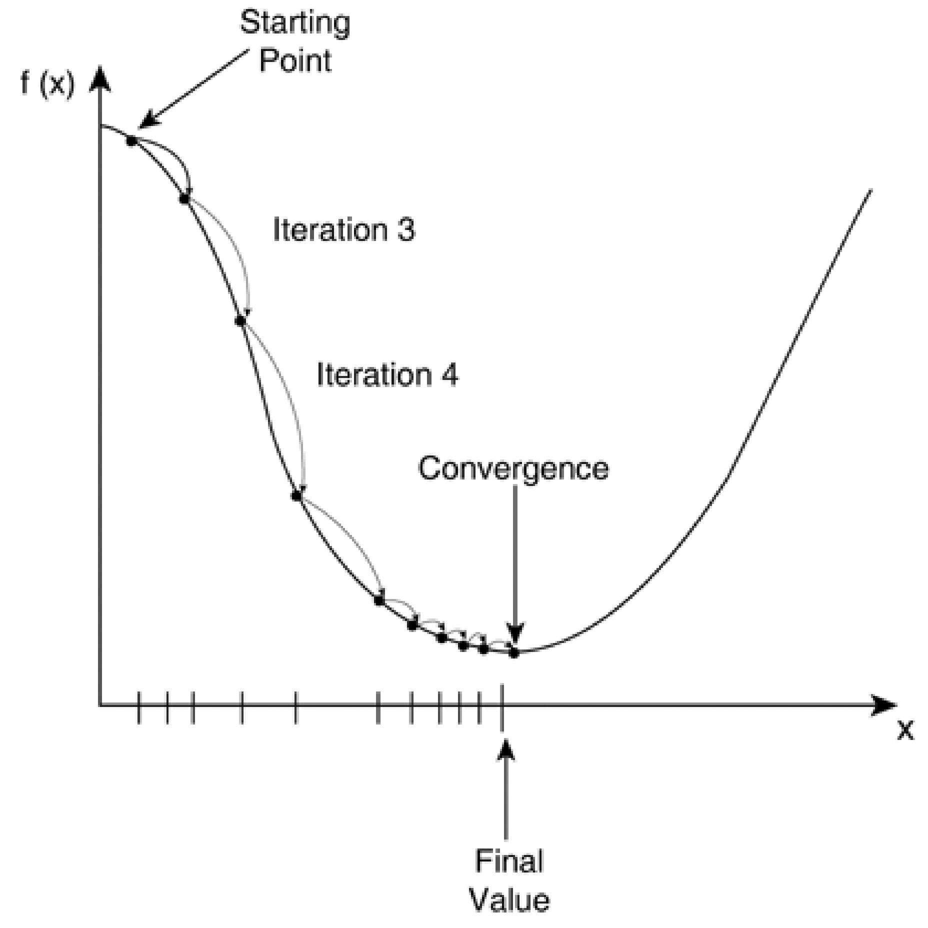
Gradient Descent

- We achieve this optimization via gradient descent.
  - We have an error function $E_n = sum_{0,\ldots,n} (h(\mathbf{x}_n) - y_n)^2$
  - We keep changing out weights in h(**x**) until we find the the minimum of E.
  - We change the weights by calculating the gradient and taking a step in proportion to the negative of that gradient.

Gradient Descent

- Each iteration the weights are changed according to the gradient and we move down the hill toward a minimum.

Picture: http://bryannotes.blogspot.com/2014/11/algorithm-stochastic-gradient_4.html



Gradient Descent Algorithm

- The weight update rule is simple:
  - For each example we update each weight according to:
    - $w_i := w_i - alpha*d/dw_i(E_i)$ (Note these are partial derivatives)
    - For $h(x) = w_0 + w_1x$ the updates look like:
      - $W_0 := W_0 - a*E$
      - $W_1 := W_1 - a*E*x$
    - alpha is the learning rate. It determines the step size. Larger alpha results in faster descent, but if it's too large the algorithm fails to converge because it oversteps the minimum and ends up bouncing back and forth over it.
  - Repeat until convergence

Adding the nonlinearity

- A straight line can't fit every data set well.
  - Adding nonlinearity is easy:
    - $h(x) = w_0 + w_1x^2$
  - This is just a simple example. Can use other non linearities as desired.
  - As with any regression over fitting can be a problem, so be careful with how you much nonlinearity you add.

Other Topics

- Logistic Regression uses gradient descent to determine a function for classification. It is very similar to linear regression except it uses discrete outputs and targets.
- Neural Networks are statistical learning algorithms that are significantly more powerful and advanced architectures used for classification and regression.

Resources

- Many great resources:
  - Wikipedia
  - Coursera Machine Learning Course:
    - https://class.coursera.org/ml-006
  - Andrew Ng's Stanford Lecutres:
    - https://www.youtube.com/course?list=ECA89DCFA6ADACE599
  - CalTech's ML MOOC:
    - http://work.caltech.edu/previous.html