# GENERAL NOTES

ANDREW S. HARD[1]

November 16, 2015

## CONTENTS


1 Introduction 2
2 Basic Statistics 2
  2.1 Deduction and Induction . . . . . 2
  2.2 Conditional Probability and Bayes' Theorem . . . . . 2
  2.3 Correlation and Covariance . . . . . 4
3 Data Structures 4
  3.1 Lists . . . . . 4
  3.2 Trees . . . . . 4
  3.3 Stacks and Queues . . . . . 4
  3.4 Graphs . . . . . 4
  3.5 Hash Maps . . . . . 5
4 Numerical Methods 5
  4.1 Root-Finding . . . . . 5
  4.2 Minimization . . . . . 5
  4.3 Regression . . . . . 5
  4.4 Matrix Algebra . . . . . 6
5 Machine Learning 6
  5.1 Introduction . . . . . 6
  5.2 Feature Selection . . . . . 6
  5.3 Regression (Again!) . . . . . 6
  5.4 Clustering . . . . . 6
  5.5 Support Vector Machines . . . . . 6
  5.6 Decision Trees . . . . . 6
  5.7 Lasso Method . . . . . 6
6 Deep Learning 6
  6.1 Overview . . . . . 6
  6.2 Perceptrons . . . . . 6
  6.3 MLP: Multi-Layer Perceptrons . . . . . 6
  6.4 RBM: Restricted Boltzmann Machines . . . . . 6
  6.5 CNN: Convolutional Neural Networks . . . . . 8
  6.6 RNN: Recurrent Neural Networks . . . . . 8
  6.7 LSTM: Long Short Term Memory . . . . . 8
  6.8 Auto-Encoders and Deep Belief Networks . . . . . 8
  6.9 Neural Turing Machines . . . . . 8
7 Lasso Method 8

## LIST OF FIGURES

Figure 1 A Boltzmann Machine diagram . . . . . 7

## LIST OF TABLES

1 *Department of Physics, University of Wisconsin, Madison, United States of America*

# 1 INTRODUCTION

As a fifth year graduate student, I have come to the realization that significant portions of my skillset and knowledge base are highly specialized. I intend to pursue a career outside of my field of study (experimental high-energy particle physics). In order to enhance my future job prospects, I decided that it would be useful to review basic concepts in computer science, statistics, mathematics, and machine learning. Most of the derivations and explanations will be borrowed from other sources. A list of references is currently being compiled.

# 2 BASIC STATISTICS

## 2.1 Deduction and Induction

**Deduction**: if $A \to B$ and $A$ is true, then $B$ is true.
**Induction**: if $A \to B$ and $B$ is true, then $A$ is more plausible.

## 2.2 Conditional Probability and Bayes' Theorem

The **conditional probability** of an event $A$ assuming that $B$ has occured, denoted $P(A|B)$, is given by:

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{1}$$

Via multiplication:

$$P(A \cap B) = P(A|B)P(B). \tag{2}$$

This makes sense. The probability of $A$ *and* $B$ is the probability of $A$ *given* $B$ times the probability of $B$. Or, just as reasonably, the probability of $A$ *and* $B$ is the probability of $B$ *given* $A$ times the probability of $A$. As an aside, the equation above can be generalized:

$$P(A \cap B \cap C) = P(A|B \cap C)P(B \cap C) = P(A|B \cap C)P(B|C)P(C). \tag{3}$$

Back to the derivation, since $P(A \cap B) = P(B \cap A)$, we can use Equation 2:

$$P(A|B)P(B) = P(B|A)P(A). \tag{4}$$

And re-arranging gives **Bayes' Theorem**:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} = \frac{P(A)}{P(B)}P(B|A). \tag{5}$$

In this equation, $P(A)$ is the **prior probability**, the initial degree of belief in $A$, or the probability of $A$ before $B$ is observed. It is essentially the probability that hypothesis $A$ is true before evidence is collected. The **posterior probability** $P(A|B)$ is the degree of belief after taking $B$ into account.

Bayes' theorem describes the probability of an event, based on conditions that might be relevant to the event. With the Bayesian probability interpretation the theorem expresses how a subjective degree of belief should rationally change to account for evidence (**Bayesian inference**). Note: in the

**Frequentist Interpretation**, probability measures a proportion of outcomes, not probability of belief.

Bayes' theorem can also be interpreted in the following manner: $A$ is some hypothesis, and $B$ is the evidence for that hypothesis.

The equation makes intuitive sense, particularly in the last form in which it is presented. If $P(B) \gg P(A)$, the probability $P(A|B)$ will be small because $B$ *usually* occurs without $A$ occurring. Even if $P(B|A)$ were 100%, the posterior probability would still be small. On the other hand, if $P(A) \gg P(B)$, $P(A|B)$ is high, even if the conditional probability $P(B|A)$ is smaller.

Nate Silver has a discussion of Bayesian statistics in "The Signal And The Noise", p245. Let $x$ be the initial estimate of the hypothesis likelihood (**prior probability**). In the book, $x$ is the initial estimate that your spouse is cheating. $y$ is a conditional probability - the probability of an observation being made given that the hypothesis is true. So $y$ in his case was the probability of underwear appearaing conditional on the spouse cheating. $z$ is the probability of an observation being made given that the hypothesis is false. So $z$ was the probability of underwear appearing if the spouse was not cheating. The **posterior probability** is the revised estimate of the hypothesis likelihood. In Silver's case, the likelihood that the spouse is cheating on you, given that underwear was found. The posterior probability is given by the expression:

$$x' = \frac{xy}{xy + z(1-x)}. \tag{6}$$

In Silver's formulation, the denominator is equivalent to the total probability of an observation being made (total probability of underwear being found). So it is the probability of the hypothesis being true times the associated conditional probability of the observation of underwear plus the probability of the hypothesis being false times the associated conditional probability of the observation.

Note that this method can be applied recursively by substituting $x'$ for $x$ in the expression. We can also come up with a recurrent form of Bayes theorem as formulated in Equation 5 by letting $P(A_{n+1}) = P(A_n|B)$.

The derivation below is from mathworld's article on Bayes' Theorem. In this example, let $A$ and $S$ be sets. Furthermore, let

$$S \equiv \bigcup_{i=1}^{N} A_i, \tag{7}$$

so that $A_i$ is an event in $S$, and $A_i \cap A_j = \emptyset \; \forall \; i \neq j$. Then:

$$A = A \bigcap S = A \bigcap (\bigcup_{i=1}^{N} A_i) = \bigcup_{i=1}^{N} (A \cap A_i) \tag{8}$$

Then we can compare the probabilities using the Law of Total Probability:

$$P(A) = P(\bigcup_{i=1}^{N} (A \cap A_i)) = \sum_{i=1}^{N} P(A \cap A_i) = \sum_{i=1}^{N} P(A_i)P(A|A_i) \tag{9}$$

Using substitution into Bayes' Theorem (Equation 5), we have a new form:

$$P(A_j|A) = \frac{P(A_j)P(A|A_j)}{\sum_{i=1}^{N} P(A_i)P(A|A_i)}. \tag{10}$$

What is the meaning of this? The denominator is simply $P(A)$, expressed as the sum of its constituents. So this reduces exactly to Bayes theorem as in Equation 5, with $A \to A_j$ and $B \to A$.

### 2.3 Correlation and Covariance

Correlation and covariance are similar. Both concepts describe the degree to which two random variables or sets of random variables tend to deviate from their expected values in similar ways.

Let $X$ and $Y$ be two random variables, with means $\mu_X$ and $\mu_Y$ and standard deviations $\sigma_X$ and $\sigma_Y$, respectively. The **covariance** is defined (using $\langle\rangle$ to denote expectation value):

$$\sigma_{XY} = \langle (X - \langle X \rangle)(Y - \langle Y \rangle) \rangle. \tag{11}$$

Similarly, the **correlation** of the two variables is defined:

$$\rho_{XY} = \frac{\langle (X - \langle X \rangle)(Y - \langle Y \rangle) \rangle}{\sigma_X \sigma_Y}. \tag{12}$$

So correlation is dimensionless, while covariance is the multiple of the units of the two variables. **Variance** is simply the covariance of a variable with itself ($\sigma_{XX}$ is usually denoted $\sigma_X^2$, the square of the standard deviation). The correlation of a variable with itself is always 1, except in the degenerate case where the two variances are zero and the correlation does not exist.

## 3 DATA STRUCTURES

### 3.1 Lists

Basic List
- Linked Lists
- Circular Lists

### 3.2 Trees

Binary trees
- Balancing
- Red-Black trees

### 3.3 Stacks and Queues

Stacks
- Queue
- Priority Queue
- Max/Min Heap

### 3.4 Graphs

**Djikstra's algorithm** is an algorithm for finding the shortest paths between nodes in a graph [1].

### 3.5 Hash Maps

# 4 NUMERICAL METHODS

This section describes a number of methods for solving common problems in scientific programming, such as root-finding, minimization, regression, and matrix algebra. The discussions of each algorithm contain information on the algorithm complexity, the implementation, and the relative merits of the approach.

Many of the methods discussed provide numerical solutions rather than analytic solutions. This means that results will be approximations, and as such will have an associated error. **Approximation error** or **absolute error** is the discrepancy between an exact value and an approximation to that exact value. Such errors can occur because approximations are used instead of real data, or because the measurement of data is imprecise.

Absolute error is given by:

$$\epsilon = |v - v_{\text{approx.}}| \tag{13}$$

Relative error is defined:

$$\eta = \frac{\text{epsilon}}{|v|} = |1 - \frac{v_{\text{approx.}}}{v}| \tag{14}$$

## 4.1 Root-Finding

The **bisection method** is a simple root-finding method that repeatedly bisects an interval and selects the subinterval containing the root for further bisection. Advantages: sthe algorithm is simple to code recursively, and it is robust (always converges). Disadvantages: it is a relatively slow algorithm.

Given a function $f$ that is continuous on the interval $[a, b]$ with $f(a)$ and $f(b)$ having opposite signs, the bisection method is guaranteed to converge. The absolute error is reduced by half during each application of the method. As a result, the bisection method *converges linearly*. It is significantly slower than other available algorithms. An upper bound on the absolute error (difference between the true and numerical result) can be determined by assuming the worst case scenario, that the root is at the midpoint of the interval $c = (a + b)/2$:

$$|c_n - c| \leqslant \frac{|b - a|}{2^n} \tag{15}$$

where $n$ is the number of iterations. The number of iterations needed to meet a particular error tolerance $\epsilon$ is given by

$$\eta = \log_2(\frac{\epsilon_0}{\epsilon}) = \frac{\log \epsilon_0 - \log \epsilon}{\log 2} \tag{16}$$

where $\epsilon_0$ is the size of the initial interval. The convergence of this algorithm is described as *linear* because $\epsilon_{n+1} = \text{constant} \times \epsilon_n$.

Next on the list: Newton-Raphson!

## 4.2 Minimization

## 4.3 Regression

Discuss linear regression

Nonlinear regression

### 4.4 Matrix Algebra

## 5 MACHINE LEARNING

### 5.1 Introduction

This section should contain a general description of data science principles, as well as use cases for different algorithms. Many tools for many problems.

Basic problems in machine learning are classification (labeling) and regression (function estimation).

How to do importance ranking of features? Feature importance, global loss function.

### 5.2 Feature Selection

Garbage in gives garbage out. Interactions between features. Interactions are not the same as correlations. Filters versus wrappers.

### 5.3 Regression (Again!)

### 5.4 Clustering

### 5.5 Support Vector Machines

### 5.6 Decision Trees

### 5.7 Lasso Method

## 6 DEEP LEARNING

### 6.1 Overview

Explain the use cases of each type of algorithm.

### 6.2 Perceptrons

### 6.3 MLP: Multi-Layer Perceptrons

### 6.4 RBM: Restricted Boltzmann Machines

https://www.cs.toronto.edu/ hinton/absps/guideTR.pdf

"**Restricted Boltzmann machines (RBMs)** have been used as generative models of many different types of data including labeled or unlabeled images [2], windows of mel-cepstral coefficients that represent speech (Mohamed et al., 2009), bags of words that represent documents (Salakhutdinov and Hinton, 2009), and user ratings of movies (Salakhutdinov et al., 2007). In their conditional form they can be used to model high-dimensional temporal sequences such as video or motion capture data (Taylor et al., 2006) or speech (Mohamed and Hinton, 2010). Their most important use is as

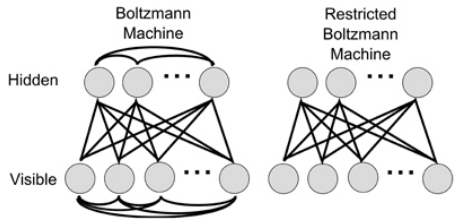


**Figure 1:** An example of a generic Boltzmann Machine versus a Restricted Boltzmann Machine, from http://www.frontiersin.org/files/Articles/58710/fnins-07-00178-r2/image_m/fnins-07-00178-g001.jpg).

learning modules that are composed to form deep belief nets [2]."

A general Boltzmann machine is a network with symmetric weights fully connecting all input and hidden nodes. The details of the weights and nodes are described later in this section. A restricted Boltzmann machine separates nodes into layers and has no intra-layer connections. General Boltzmann machines, while theoretically providing a more diverse architecture, are difficult to train in practice. The difference between a general Boltzmann Machine and the Restricted Boltzmann Machine is illustrated in Figure 1.

The training procedure for RBMs is more complicated than back-propagation. They typically use a procedure called constrastive divergence [3]. There are several meta-parameters including learning rate, momentum, weight-cost, sparsity target, initial weights, number of hidden units, and mini-batch sizes.

For simplicity, the outline below will consider a two-layer network consisting of the visible unit layer representing the input data and a layer of hidden units called **feature detectors**. Deep neural networks can be trained by successively stacking RBMs, with the hidden layer of the first RBM used as an input for the second. Greedy pairwise training of layers has been demonstrated to be effective in learning deep architectures (reference greedy pair training). Input data are stochastic binary pixels (or continuous variables scaled to $[0, 1]$). Feature detectors are also stochastic binary units. The visible units are connected to the feature detectors using symmetrtically weighted connections. A configuration of visible and hidden units has an energy (Hopfield, 1982) given by:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i h_i w_{ij} \tag{17}$$

for $i \in visible$ and $j \in hidden$. In this equation, $v_i$ and $h_j$ are the binary states of the visible unit $i$ and hidden unit $j$, respectively. $a_i$ and $b_j$ are their biases and $w_{ij}$ is the weight between them.

The energy can be used to define a probability of this pairing of visible and hidden units (hence the name "Boltzmann Machine":

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} \tag{18}$$

$Z$ is the partition function:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{19}$$

The network assigns a probability to a visible vector $\mathbf{v}$, which can be calculated by summing $p(\mathbf{v}, \mathbf{h})$ over all possible hidden vectors:

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{20}$$

A higher energy input is a less-probable input, according to the energy function. The network is trained to increase the probability (lowering the energy) assigned by the network to signal inputs while lowering the probability (increasing the energy) assigned to noise. To get the change in weights:

$$\frac{\delta \log p(\mathbf{v})}{\delta w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{21}$$

The angle brackets denote expectation values under the specified distribution (data or model). A learning rule for weights using stochastic gradient ascent on the log probability, and a learning rate $\epsilon$, is then simply:

$$\delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}). \tag{22}$$

Now all that remains is to get an unbiased sample of $\langle v_i h_j \rangle_{data}$ and $\langle v_i h_j \rangle_{model}$. The first one is straightforward, since there are no connections between hidden units of the same layer in an RBM. Given a randomly selected training image, $\mathbf{v}$, the binary state, $h_j$ , of each hidden unit, j, is set to 1 with probability

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{23}$$

### 6.5 CNN: Convolutional Neural Networks

### 6.6 RNN: Recurrent Neural Networks

### 6.7 LSTM: Long Short Term Memory

### 6.8 Auto-Encoders and Deep Belief Networks

### 6.9 Neural Turing Machines

## 7 LASSO METHOD

## REFERENCES

[1] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[2] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[3] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, August 2002.