Computational Learning Theory **Fall Semester, 2012**

Lecture 1: October 21, 2012

Lecturer: Yishay Mansour
Scribe: Oren Ish-Am, Hila Peleg, Nir Hemed[1]

1.1 Introduction and Overview

1.1.1 What is machine learning?

The goal of machine learning is to develop computer programs that automatically improve with experience. Such programs adapt themselves to changing conditions and extract meaningful information from raw data.
Machine learning (as a field of study) grew out of AI and has strong ties to Algorithms and Statistics.
Machine learning is ubiquitous today, and is found for example in Google search, Facebook friends suggestions, Amazon book suggestions and many more. Here is a definition of Machine Learning by one of the poineers in the field:

> "Field of study that gives computers the ability to learn without being explicitly programmed"

`-Arthur Samuel`

1.1.2 Typical applications

There are numerous applications of machine learning. These applications are divided into several categories:

- **Classification**
  Each data item has a type (also called a label). There could be two or more types and the objective is to fit each item with the correct type.
  For example, items with a binary label could be emails (is it Spam or not?), credit card transactions (legal or fraud?) and patient tests (has/doesn't have cancer). An example of an item with multiple labels is a news article, where the labels are the news categories (political, science, culture, sports etc.).
  Classification algorithms use known data and label sets (also called "training data") to predict the labels on new data items.

[1]Based on scribes written by Aviv Gruber, Shai Hertz, Chavatzelet Tryster and Rinat Pichker (October 17, 2010)

- **Clustering**
  In these problems we wish to automatically partition the data into meaningful data groups. Usually there is more than one way of partitioning the data into groups, therefore there is no right or wrong answer.
  Examples: document classification, news classification.
  It is possible for a person to tag the documents or news items, but we would like the tagging to be done automatically without human intervention.

- **Control**
  In these problems current decisions affect future conditions, for example, flying a toy helicopter. These problems are hard to program explicitly due to the large amount of corner cases and options.

- **Collaborative Filtering**
  In these problems we wish to filter information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc.
  Examples: computer vision, speech recognition, translation, friend recommendation (Facebook).

1.1.3 *Types of Machine Learning*

Supervised Vs. unsupervised

Supervised learning: The algorithm is granted access to data accompanied by the right classification (true labels). The machine attempts to build a model (function) which will classify new data with minimum loss (this will be discussed later).
For example - data regarding the financial details of people who received a loan and whether the loan was returned. Based on this information we will try to predict whether other people, given their financial details, will return the loan.
Unsupervised learning: The algorithm is given access to data and has no knowledge of how the data should be classified. The objective is to split the data into groups that "make sense". For example, given a large set of hand-written digits, to group them into 10 equivalence classes, one per digit.

Active Vs. Passive

Passive learning: Given a data set, we build a model and then use it to predict something regarding new data.
Active learning: Given a data set we can selectively choose on which data to build the model. We can also create a new example and request its classification, called membership query.

For example: based on information about other customers we will build a profile of a new customer and ask the algorithm to classify its behavior.

Teacher Learning

The machine tries to mimic an expert and classify examples similar to the way the expert does. An implementation of teacher learning are Expert Systems. An expert system is a software that attempts to provide an answer to a problem, or clarify uncertainties where normally one or more human experts would need to be consulted.
The problem with this is that human knowledge is hard to systematically program and also, sometimes, we wish to surpass human performance.

Online Vs. Batch Learning

Batch: The algorithm has unlimited access to the data. It can go through it several times and has no storage limitation.
Online: The algorithm gets one example at a time, classifies it, learns from its own answer, and repeats this process with each additional example. It is assumed that the stream of examples is infinite so the machine cannot store all the data and is somehow restricted to a narrow "window" in the data stream.

1.1.4 Why do we need Machine Learning?

1. Tasks in which it is difficult to define precisely how they should be executed:
   When driving a car, one uses human generalization capabilities. We would like the machine to act as the driver would. Many times it's much simpler to give examples rather then rules for how to act (drive).

2. Tasks that are beyond human capability:
   The analysis of huge database or discovering interesting and unknown phenomena.

In both cases writing a program that will solve these problems is difficult and standard programming does not provide tools to deal with these difficulties.

In this course:

- classification
- supervised
- passive

- no teacher
- batch and online model

1.2 Building a machine learning model

In order to build a machine learning model we will have to answer the following questions:

1. We need to hypothesize how the examples (including the classification) are created. This is an assumption regarding the environment.
2. We need to decide how to compare the classifications of different algorithms. Obviously we would like to minimize the number of errors, but perhaps many small errors are preferable to a few big errors.
3. We need to introduce assumptions about the mapping between an example and its classification.

1.2.1 Machine learning model example: Complete Information

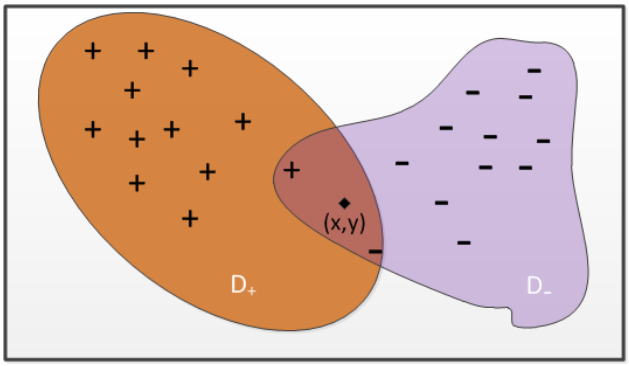

Figure 1.1: Classified Data Set

From the domain of two dimensional space ($\mathbb{R}^2$) we are given a set of points which has been classified into two groups: '+' and '-' as in Figure 1.1. Notice that the '+' and '-' here are simply labels given to each point.
Now, given a new point (x, y), we would like to be able to classify it, that is, give it one of the two labels.
In this scenario, we assume that we are provided with **Complete Information:** the distribution that generates the '+'s is D_+, the distribution that generates the '-'s is D_- . In this example, the shaded regions are schematic representations of the probability density of D_+

and D_-. Since every point is assumed to have a label, the probabilities are complementary, that is, for any point s it holds that $D_+(s) = 1 - D_-(s)$.

Given a new point (x, y) we would like to be be able to assign a label to it.
The probability that (x, y) has a '+' label is $\Pr[+|(x, y)]$. Using Bayesian rules of conditional probability:

$$\Pr[+|(x, y)] = \frac{\Pr[(x, y)|+] \cdot \Pr[+]}{\Pr[(x, y)]} = p$$

Let us define λ as the probability to draw a '+' label point from the domain, that is $\lambda = \Pr[+]$.
$\Pr[(x, y)]$ is the probability to draw this point from the whole domain. Since the distribution probabilities are known, this can be calculated as:

$$D[(x, y)] \equiv Pr[(x, y)] = \lambda \cdot D_+[(x, y)] + (1 - \lambda) \cdot D_-[(x, y)]$$

1.2.2 Prediction and loss model

We are now interested in classifying the point (x, y). Which classification shall we choose? The obvious choice would be $'+'$ if $D_+[(x, y)] > D_-[(x, y)]$. Is this always the best approach? Assume we are dealing with a medical test for cancer where $'+'$ means the patient has cancer, $'-'$ means he doesn't. Here we can allow ourselves to make the mistake of telling someone without cancer that he's probably ill (and sending him for further tests) but not vice-versa! In this case the "loss" of a false negative misclassification will incur a huge compensation lawsuit. On the other hand (false positive) the cost of some extra tests is negligible.
We wish to minimize the loss, but there can be many loss models, let's look at a few.

Boolean error (0-1 Loss)

If we commit an error, we lose 1, otherwise, we lose nothing. We compare
$\Pr[+|(x, y)] = p$ (the probability of receiving '+' label for (x, y)) versus
$\Pr[-|(x, y)] = 1 - p$ (The probability of receiving '-' label for point (x, y)), and choose '+' if $p > 1 - p$ that is if $p > \frac{1}{2}$.
0-1 loss is very insensitive to the probabilities; $p = 0.99$ or $p = 0.51$ would both result in choosing '+'.

Quadratic Loss

In this model, the machine outputs a real number q.
The loss is given by $(1 - q)^2$ for an outcome of '+', and q^2 for an outcome of '-'.
The expected loss is $l(q) = p \cdot (1 - q)^2 + (1 - p) \cdot q^2$. What would be the optimal q?

To find the minimal loss, we take the derivative:

$$\frac{dl(q)}{dq} = 2q(1-p) - 2p(1-q) = 2q - 2p = 0 \quad \Rightarrow \quad p = q$$

Taking the second derivative we verify that this is indeed a minimum:

$$\frac{d^2l(q)}{d^2q} = 2p + 2(1-p) = 2 > 0$$

Therefore, using the quadratic loss function we minimize the loss by setting $p = q$.
Notice that quadratic loss is sensitive to outliers.

Logarithmic Loss

Again the machine outputs a real number q. We'll be penalized as follows:

- $-\log q$ for an erroneous '+'
- $-\log(1-q)$ for an erroneous '-'

Our total expected loss will be $l(q) = -p \cdot \log q - (1-p) \cdot \log(1-q)$. What would be the optimal q?
To find the minimum loss, we take the derivative:

$$\frac{dl(q)}{dq} = \frac{-p}{-q} - \frac{1-p}{1-q} = 0 \quad \Rightarrow \quad p = q$$

Again, setting $q = p$ we will achieve minimal loss.
Note the difference between quadratic loss and logarithmic loss: Although both result in a choice of $q = p$, in logarithmic loss it is possible to experience infinite loss (when setting $q = 0$ or $q = 1$), whereas quadratic loss will result in the loss of at most 1 for any q.

1.3 Classification Assumptions

Given a set of points $X = \{x_1, ...x_n | x_i \in \mathbb{R}^d, d \in \mathbb{N}\}$ and corresponding labels $B = \{b_1, ...b_n | b_i \in \{0,1\}\}$ we would like to find a classification function $h : \mathbb{R}^d \mapsto \{0,1\}$ which receives a new point $x \in \mathbb{R}^d$ and returns the correct label as much as possible.
Since in our view, $\forall x \in \mathbb{R}^d$ there exists a (let's assume single) correct label $b \in \{0,1\}$, one can talk about the 'classification function', $f : \mathbb{R}^d \mapsto \{0,1\}$ which given a point returns the correct label. Obviously, we would like to have $h = f$ but f might not have a compact representation. So what can we do?

1.3.1 Explicit Assumption

We can assume that f has a (simple) explicit formulation, for example:

$$f(x) = \sum_{i=1}^{d} \alpha_i x_i$$

i.e., $f(x)$ is a linear function of the input. In this case, the algorithm will search for the α_i. If our hypothesis is correct and there are enough equations, we will be able to find all the α_i. If the hypothesis is incorrect we will either get an incorrect value for a given α_i or there will be no solution, which will contradict our assumption.
A preferable method would be to choose

$$h(x) = \sum_{i=1}^{d} \alpha_i x_i$$

And then, the closer f is to a linear function, the more precise the prediction. The difference is that we are restricting our hypothesis $h(x)$ and not the target function $f(x)$.

1.3.2 Bayesian Inference

In this model, our chosen function h is part of a (possibly infinite) class of functions (hypothesis) H, one of which *may* be f.
Like before, given $S =< X, B >$ and a new point $x \in \mathbb{R}^d$ we would like to find the correct label. Putting this in mathematical terms, we would like to calculate

$$\Pr[f(x) = 1 | x, S]$$

Choosing to compute for the '1' label is arbitrary, knowing the probability for one label automatically gives us the probability for the other (why?).

Now, assuming $f \in H$ we can write this as

$$\Pr[f(x) = 1 | x, S] = \sum_{h \in H} h(x) \Pr[f = h | S]$$

And using Bayesian inference we can further develop this

$$\Pr[f = h | S] = \frac{Pr[S | f = h] \cdot \Pr(f = h)}{\Pr[S]}$$

Let's look at the terms in the equation.
$\Pr[S|f = h]$ denotes the probability of seeing the set S (points and their labels) if h is indeed

the correct classifier. Since the functions H are computable, we can compute $\forall h \in H, \forall x \in X, h(x)$ and see how close the result is to B.
$\Pr[S]$ is independent of the classifier and therefore often acts as a normalizer and can generally be ignored.
$\Pr[f = h]$ is our assumption that a certain $h \in H$ is the correct classifier *before* we have seen the set S. This is called the *a-priori* distribution on H, and it should incorporate all our prior knowledge.
Sometimes we can assign probabilities to certain hypotheses even before we see the data, for instance, we will prefer simpler hypotheses to more complex ones. Many times we like to select a non-informative prior, such as a uniform distribution over H.
Two popular Bayesian methods are:

- Maximum Likelihood (ML)
  In this method we select the $h \in H$ which maximizes the probability of seeing the set S; i.e., $\text{argmax}_{h \in H} \Pr[S|h]$. The *a-priori* distribution is ignored here.

- Maximum *a-posteriori* (MAP)
  In this method we wish to find the most probable function h, given the set S (the h most likely to produce the labels B given the points X); $\text{argmax}_{h \in H} \Pr[h|S]$

These two methods are identical when the *a-priori* distribution is uniform.

1.3.3 The basic PAC Model

We have an unknown distribution D over domain X (this means that every $x \in X$ is seen with probability $D(x)$), set of samples i.i.d. $S \subset X$, an unknown target function $f(x)$ that classifies each point $x \in X$, and a set of hypotheses H.
The Goal:
We are looking for a hypothesis $h \in H$ that minimizes the **true error**:

$$\varepsilon(h) = \Pr_D[h(x) \neq f(x)] \stackrel{def}{=} \sum_{x \in X} D(x) \cdot I(h(x) \neq f(x))$$

Where I is the indicator function, giving '1' when $h(x) \neq f(x)$ and '0' otherwise.
Note: $\varepsilon(h)$ may be non-zero even for the optimal h, since we are not guaranteed that $f \in H$.
Problem:
Had we known D, we could simply go over all $h \in H$ and find the hypothesis that minimizes $\varepsilon(h)$. However, we do not have this information, but only a *sample* S drawn i.i.d from X according to D.
Thus, we can minimize only the **observed error**:

$$\hat{\varepsilon}(h) = \sum_{x \in S} D(x) \cdot I(h(x) \neq f(x))$$

But since S was i.i.d. from X then $D(x)$ is the same for all $x \in S$. We can also approximate $D(x) = \frac{1}{|S|}$ for all $x \in S$ and so we get

$$\hat{\varepsilon}(h) = \frac{1}{|S|} \sum_{x \in S} I(h(x) \neq f(x))$$

(the error on the samples).

This means we may find the best hypothesis that matches $f(x)$ over the sample S, but not necessary over X. This raises two questions:

- How close is $\varepsilon(h)$ to $\hat{\varepsilon}(h)$?
- Is the best hypothesis on the sample also the best on the entire domain?

Another issue is a computational one - finding h that will minimize the observed error in an efficient manner.

1.4 Hypothesis Classes

These are the set of possible hypotheses H we were discussing before. From each class, we will wish to compile the best possible h.

1.4.1 Hyperplane

We are looking for a hyperplane that would separate the problem space into two different classification regions.
A hyperplane is a very simple model, and also a limited one:

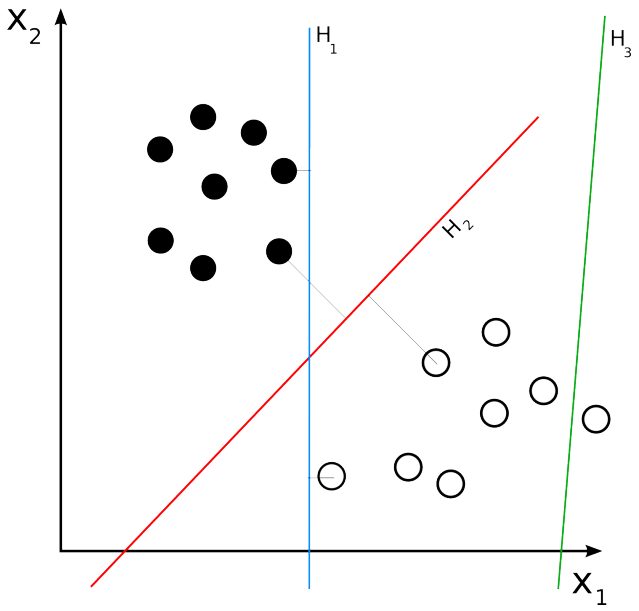

Figure 1.2: Hyperplanes separating data points

1.4.2 Decision Trees

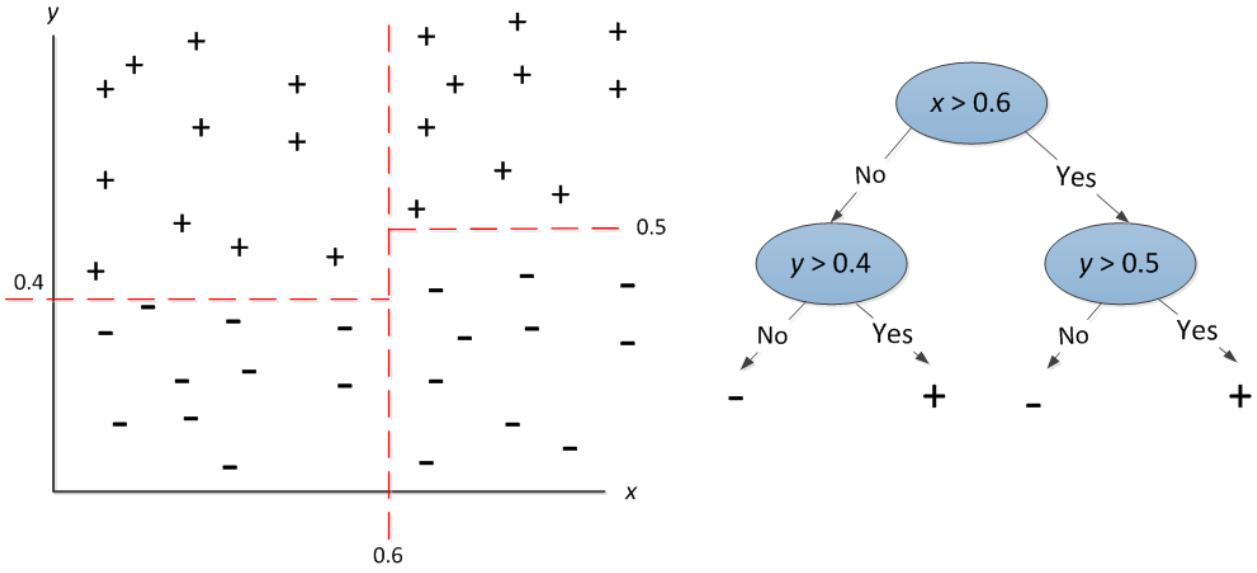

Figure 1.3: Decision Trees

1.4.3 Bad example

What happens if we do not have enough samples? Let's take a look at the following case: $S \subset X = \{0,1\}^d$, $f(x)$ is a random function $\{0,1\}^d \mapsto \{0,1\}$ and D is uniform over X.
In the hypothesis class $H = \{h_1, ..., h_d\}$, each function classifies a point according to a single input attribute, i.e., $\forall x \in \{0,1\}^d, x = a_1 a_2 ... a_d, h_i(x) = a_i$.
What will happen if $|S| = m < \log d$? For any $x \in S$, half the h's would classify it correctly, and half wouldn't.
The probability that some h classifies all m points correctly is $\frac{1}{2^m}$. The probability that there exists an h which classifies *all* the points correctly is $1-(1-\frac{1}{2^m})^d > 1-\frac{1}{e}$. This means that with constant probability we will find an h that works perfectly on the sample S but will (most likely) fail miserably on X.
This is the overfitting phenomenon.

1.4.4 PAC (Probably Approximately Correct) Methodology

- Choosing $h \in H$ that will minimize the observed error.
- Proving the generalization error.

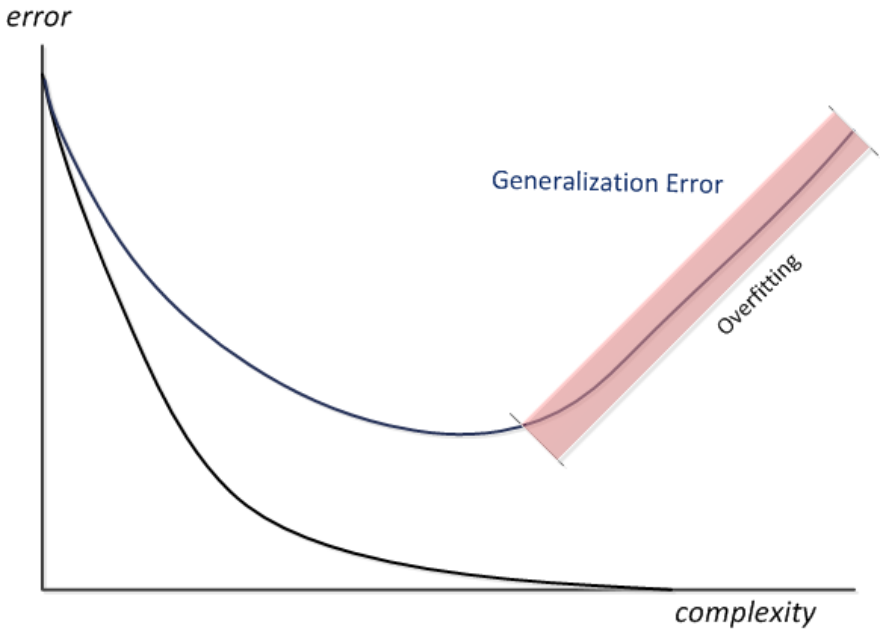

Figure 1.4: Generalization Error

1.5 Complexity versus Generalization

When choosing a model we have a tradeoff between hypothesis complexity and observed error. More complex hypotheses have lower observed errors, but may have higher true errors. A complex hypothesis that has many degrees of freedom would use all of them in order to reduce the observed error. By doing this, it will adapt to noise or to the specific sample structure and not to the structure of the true underlying distribution.
In order to avoid the "over-fitting" problem we would like to limit the complexity of the model and try to find a simple model even if it has a slightly higher observed error.

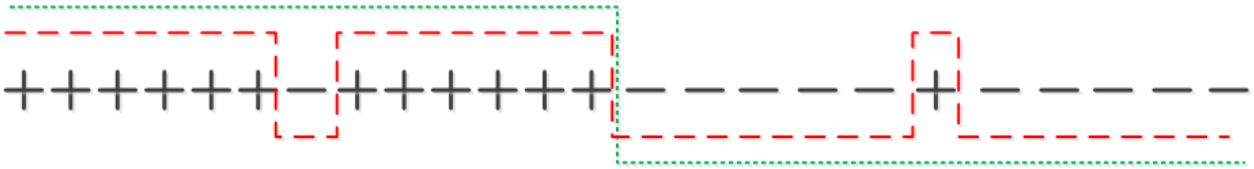

Figure 1.5: Two hypotheses for a given sample set. The complexity is the number of transitions from plus to minus. The dotted hypotheses is much better (and simpler) then the dashed one, though its observed error on the data is higher

To compensate a simple model for its (possibly) higher observed error, we can place a

penalty on complexity. We can do this by using one of the following criteria:

- **Minimum Description Length**

$$\hat{\varepsilon} + \frac{|\text{code length of h}|}{m}$$

- **Structural Risk Minimization**

$$\hat{\varepsilon} + \sqrt{\frac{\log |h|}{m}}$$

  (Where m is the number of items in the sample)

Using this formula we can calculate generalization errors.

1.6 Online Models

In time t, we receive a point x_t. We guess its classification y_t, and see the right classification $f(x_t)$.
Our goal is to minimize the number of errors, but this poses a problem - we can err each and every time, since our opponent (the one who reveals the correct classification) can always choose the opposite label.
The solution to this problem is to assume there exists a function $h^* \in H$ that always classifies correctly, prior to the choices we make.
Our goal now is to approximate this function h^*, given $x_1, x_2, ..., x_T$.
The advantages of online models:

- We don't need a big sample to start classifying (though we'll probably make a lot of mistakes in the beginning).
- The hypothesis can be changed during the whole process.
- We make no assumptions about the distribution, and it can also change over time.

1.7 Course Structure

1. Basic models - PAC, Bayes, and maybe Regression
2. Online models - Perceptron, Regret, Boosting
3. Sampling complexity - VC-Dimension
4. Efficient algorithms - Decision trees, Kernel + SVM, Fourier transform

1.8 Basic Concepts in Probability

During the course we'll often use probabilistic arguments. We recall here a few basic tools from probabilistic theory:

Markov Inequality

Let x be a non negative random variable (r.v.) and $E[x]$ the *expected value* (mean) of x, then:

$$Pr[x \geq \alpha] \quad \leq \quad \frac{E[x]}{\alpha}$$

Chebyshev Inequality

Suppose that x is arbitrary with variance $Var(x)$, then:

$$Pr[|x - \eta| \geq \beta] \ \leq \ \frac{Var(x)}{\beta^2} \quad where\ \eta = E[x].$$

Chernoff Inequality

Let the sequence of random variables $x_1, \ldots, x_n$ where $x_i \ \in \ \{0, 1\}$ be independent identically distributed (i.i.d.), and $Pr[x_i = 1] = p$, then:

$$Pr[|\sum_{i=1}^{n} \frac{x_i}{n} - p| \ \geq \ \lambda] \ \leq \ 2e^{-2\lambda^2 n}$$

The last inequality is especially interesting, since it gives us a tool to argue "how fast" the "observed mean" would converge to the "true mean".