COMS 4771
Introduction to Machine Learning

Nakul Verma
Machine learning: what?

Study of making machines **learn** a concept **without** having to explicitly **program** it.

- Constructing algorithms that can:
  - learn from input data, and be able to make predictions.
  - find interesting patterns in data.

- Analyzing these algorithms to understand the limits of ‘learning’
Machine learning: why?

We are smart programmers, why can’t we just write some code with a set of rules to solve a particular problem?

Write down a set of rules to code to distinguish these two faces:

What if we don’t even know the explicit task we want to solve?
Machine learning: problems in the real world

• Recommendation systems (Netflix, Amazon, Overstock)

• Stock prediction (Goldman Sachs, Morgan Stanley)

• Risk analysis (Credit card, Insurance)

• Face and object recognition (Cameras, Facebook, Microsoft)

• Speech recognition (Siri, Cortana, Alexa, Dragon)

• Search engines and content filtering (Google, Yahoo, Bing)
so.... how do we do it?

This is what we will focus on in this class!
This course

We will learn:

• Study a prediction problem in an **abstract manner** and come up with a solution which is **applicable to many problems** simultaneously.

• Different types of paradigms and **algorithms** that have been successful in prediction tasks.

• How to systematically **analyze** how good an algorithm is for a prediction task.
Prerequisites

Mathematical prerequisites
• Basics of probability and statistics
• Linear algebra
• Calculus

Computational prerequisites
• Basics of algorithms and datastructure design
• Ability to program in a high-level language.
Website:

http://www.cs.columbia.edu/~verma/classes/fal7/coms4771/

The team:

Instructor: Nakul Verma (me)
TAs: Ashu Nanda, Dheeraj Kalmekolan, Abhimanyu Yadav, Sheallika Singh
Students: you!

Evaluation:

• Homeworks (40%)
• Exam 1 (30%)
• Exam 2 (30%)
Homeworks:

- No late homework
- **Must** type your homework (no handwritten homework)
- Please include your name and UNI
- Submit a pdf copy of the assignment via gradescope (94NVV9)
- Email the tarball of the programming segment to the TA

- We encourage discussing the problems (piazza), but **please don’t copy**.
Announcement!

- Visit the course website
- Review the basics (prerequisites)
- HW0 is out!
- Sign up on Piazza & Gradescope
Let’s get started!
A closer look at some prediction problems...

• Handwritten character recognition:

\[
\{ 0, 1, 2, \ldots, 9 \}
\]

• Spam filtering:

\[
\{ \text{spam}, \text{not spam} \}
\]

• Object recognition:

\[
\{ \text{building, tree, car, road, sky,\ldots} \}
\]
Machine Learning: the basics...

Commonalities in a prediction problem:

Input: \( \mathbf{x} = \begin{pmatrix} x^{(1)} \\ \vdots \\ x^{(d)} \end{pmatrix} \in \mathcal{X} \)

Output: \( y = \begin{pmatrix} 5 \end{pmatrix} \in \mathcal{Y} \{0, 1, 2, \ldots, 9\} \)

To learn: \( f \)
Machine Learning: the basics...

Data: \((\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots \in \mathcal{X} \times \mathcal{Y}\)

Assumption: there is a (relatively simple) function \(f^* : \mathcal{X} \rightarrow \mathcal{Y}\) such that \(f^*(\vec{x}_i) = y_i\) for most \(i\)

Learning task: given \(n\) examples from the data, find an approximation \(\hat{f} \approx f^*\)

Goal: \(\hat{f}\) gives mostly correct prediction on unseen examples
Data: \( \bar{x}_1, \bar{x}_2, \ldots \in \mathcal{X} \)

Assumption: there is an underlying structure in \( \mathcal{X} \)

Learning task: discover the structure given \( n \) examples from the data

Goal: come up with the summary of the data using the discovered structure

More later in the course...
Supervised Machine Learning

Statistical modeling approach:

- Labeled training data $(n \text{ examples from data})$

- $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots (\vec{x}_n, y_n)$
  - drawn independently from a fixed underlying distribution
  - (also called the i.i.d. assumption)

How to select $\hat{f} \in \mathcal{F}$?

- Maximum likelihood (best fits the data)
- Maximum a posteriori (best fits the data but incorporates prior assumptions)
- Optimization of ‘loss’ criterion (best discriminates the labels)
- ...

select $\hat{f}$ from...

from a pool of models $\mathcal{F}$

that maximizes label agreement of the training data
Given some data \( \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathcal{X} \) i.i.d. (Let’s forget about the labels for now)

Say we have a model class \( \mathcal{P} = \{p_\theta \mid \theta \in \Theta\} \)

find the parameter settings \( \theta \) that **best fits** the data.

If each model \( p \), is a **probability model** then we can find the best fitting probability model via the **likelihood estimation**!

\[
\text{Likelihood } \mathcal{L}(\theta|X) := P(X|\theta) = P(\vec{x}_1, \ldots, \vec{x}_n|\theta) = \prod_{i=1}^{n} P(\vec{x}_i|\theta) = \prod_{i=1}^{n} p_\theta(\vec{x}_i)
\]

Interpretation: How **probable** (or how likely) is the data given the model \( p_\theta \) ?

Parameter setting \( \theta \) that maximizes \( \mathcal{L}(\theta|X) \)

\[
\arg \max_{\theta} \mathcal{L}(\theta|X) = \arg \max_{\theta} \prod_{i=1}^{n} p_\theta(\vec{x}_i)
\]
MLE Example

Fitting a statistical probability model to heights of females

Height data (in inches): $60, 62, 53, 58, \ldots \in \mathbb{R}$

$x_1, x_2, \ldots x_n \in \mathcal{X}$

Model class: Gaussian models in $\mathbb{R}$

$$p_{\theta}(x) = p_{\{\mu, \sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \quad \mu = \text{mean parameter} \quad \sigma^2 = \text{variance parameter} > 0$$

So, what is the MLE for the given data $X$?
MLE Example (contd.)

Height data (in inches): \( x_1, x_2, \ldots, x_n \in \mathcal{X} = \mathbb{R} \)

Model class: Gaussian models in \( \mathbb{R} \)
\[
p_{\{\mu, \sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right)
\]

MLE:
\[
\arg \max_{\theta} \mathcal{L}(\theta|X) = \arg \max_{\mu, \sigma^2} \prod_{i=1}^{n} p_{\{\mu, \sigma^2\}}(x_i)
\]

Good luck!

Trick #1:
\[
\arg \max_{\theta} \mathcal{L}(\theta|X) = \arg \max_{\theta} \log \mathcal{L}(\theta|X)
\]

“Log” likelihood

Trick #2: finding max (or other extreme values) of a function is simply analyzing the ‘stationary points’ of a function. That is, values at which the derivative of the function is zero!
MLE Example (contd. 2)

Let’s calculate the best fitting $\theta = \{\mu, \sigma^2\}$

$$\arg\max_\theta \mathcal{L}(\theta|X) = \arg\max_\theta \log \mathcal{L}(\theta|X)$$

"Log" likelihood

$$= \arg\max_{\mu, \sigma^2} \log \left( \prod_{i=1}^{n} p_{\{\mu, \sigma^2\}}(x_i) \right)$$

i.i.d.

$$= \arg\max_{\mu, \sigma^2} \sum_{i=1}^{n} \log \left( p_{\{\mu, \sigma^2\}}(x_i) \right)$$

$$= \arg\max_{\mu, \sigma^2} \sum_{i=1}^{n} \left[ -\frac{1}{2} \log(2\pi\sigma^2) - \frac{(x_i - \mu)^2}{2\sigma^2} \right]$$

$g_i(\mu, \sigma^2)$

Maximizing $\mu$:

$$0 = \nabla_\mu \left( \sum_{i=1}^{n} g_i(\mu, \sigma^2) \right)$$

$$\Rightarrow \mu_{ML} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Maximizing $\sigma^2$:

$$\sigma^2_{ML} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$
So, the best fitting **Gaussian model** 

\[ p_{\{\mu, \sigma^2\}}(x) := \frac{1}{\sqrt{2\pi \sigma^2}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right) \]

Female height data: 60, 62, 53, 58, ... \( \in \mathbb{R} \)

\[ x_1, x_2, \ldots, x_n \in \mathcal{X} \]

Is the one with parameters:

\[ \mu = \frac{1}{n} \sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 \]

What about other model classes?
Other popular probability models

- Bernoulli model (coin tosses)   \textit{Scalar valued}
- Multinomial model (dice rolls) \textit{Scalar valued}
- Poisson model (rare counting events) \textit{Scalar valued}
- Gaussian model (most common phenomenon) \textit{Scalar valued}

Most machine learning data is vector valued!

- Multivariate Gaussian Model \textit{Vector valued}

Multivariate version available of other scalar valued models
Multivariate Gaussian

**Univariate** $\mathbb{R}$

\[
p_{\{\mu, \sigma^2\}}(x) := \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right)
\]

$\mu$ = mean parameter
$\sigma^2$ = variance parameter $> 0$

**Multivariate** $\mathbb{R}^d$

\[
p_{\{\mu, \Sigma\}}(\vec{x}) := \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp \left( -\frac{1}{2} (\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu}) \right)
\]

$\vec{\mu}$ = mean vector
$\Sigma$ = Covariance matrix (positive definite)
MLE sounds great, how do we use it to do classification using labelled data?

\[
\hat{f}(\vec{x}) = \arg \max_{y \in \mathcal{Y}} P[Y = y | X = \vec{x}]
\]

\[
= \arg \max_{y \in \mathcal{Y}} \frac{P[X = \vec{x} | Y = y] \cdot P[Y = y]}{P[X = \vec{x}]}
\]

Bayes rule

\[
= \arg \max_{y \in \mathcal{Y}} P[X = \vec{x} | Y = y] \cdot P[Y = y]
\]

Class prior:

Simply the probability of data sample occurring from a category

Class conditional:

Use a separate probability model individual categories/class-type
We can find the appropriate parameters for the model using MLE!
Task: learn a classifier to distinguish **males** from **females** based on say height and weight measurements

Classifier: \[ \hat{f}(\vec{x}) = \arg\max_{y \in \{\text{male}, \text{female}\}} P[X = \vec{x}|Y = y] \cdot P[Y = y] \]

Using **labelled** training data, learn all the parameters:

**Learning class priors:**

- \( P[Y = \text{male}] = \text{fraction of training data labelled as male} \)
- \( P[Y = \text{female}] = \text{fraction of training data labelled as female} \)

**Learning class conditionals:**

- \( P[X|Y = \text{male}] = p_{\theta(\text{male})}(X) \)
- \( P[X|Y = \text{female}] = p_{\theta(\text{female})}(X) \)

\( \theta(\text{male}) = \text{MLE using only male data} \)

\( \theta(\text{female}) = \text{MLE using only female data} \)
What are we doing geometrically?

Data geometry:

\[ x = \mathbb{R}^2 = \text{Weight} \times \text{Height} \]

- [Blue] male data
- [Red] female data
What are we doing geometrically?

Data geometry:

\[ X = \mathbb{R}^2 = \text{Weight} \times \text{Height} \]

- Male data
- Female data
- MLE Gaussian (male)
- MLE Gaussian (female)
Classification via MLE Example

Task: learn a classifier to distinguish males from females based on say height and weight measurements

Classifier: \[ \hat{f}(\vec{x}) = \arg \max_{y \in \{\text{male, female}\}} P[X = \vec{x} | Y = y] \cdot P[Y = y] \]

Using labelled training data, learn all the parameters:

Learning class priors:

\[ P[Y = \text{male}] = \text{fraction of training data labelled as male} \]
\[ P[Y = \text{female}] = \text{fraction of training data labelled as male} \]

Learning class conditionals:

\[ P[X | Y = \text{male}] = p_{\theta(\text{male})}(X) \]
\[ P[X | Y = \text{female}] = p_{\theta(\text{female})}(X) \]

\[ \theta(\text{male}) = \text{MLE using only male data} \]
\[ \theta(\text{female}) = \text{MLE using only female data} \]
Classification via MLE Example

We just made our first predictor \( \hat{f} \)!

But why:
\[
\hat{f}(\vec{x}) = \arg \max_{y \in \mathcal{Y}} \quad P[Y = y | X = \vec{x}]
\]
Why the particular $f = \arg\max_{y} P[Y|X]$?

Accuracy of a classifier $f$:

$$P(\bar{x},y)\left[ f(\bar{x}) = y \right] = \mathbb{E}(\bar{x},y)\left[ \mathbf{1}\left[ f(\bar{x}) = y \right] \right]$$

Assume binary classification (for simplicity):

$\mathcal{Y} = \{0, 1\}$

Let:

$$f(\bar{x}) = \arg \max_{y\in\{0,1\}} P[Y = y|X = \bar{x}] \quad \text{Bayes classifier}$$

$$g(\bar{x}) = \mathcal{X} \rightarrow \{0, 1\} \quad \text{any classifier}$$

Theorem:

$$P(\bar{x},y)\left[ g(\bar{x}) = y \right] \leq P(\bar{x},y)\left[ f(\bar{x}) = y \right]$$

!!! Bayes classifier is optimal !!!
**Theorem:** \( P_{(\mathbf{x},y)}[g(\mathbf{x}) = y] \leq P_{(\mathbf{x},y)}[f(\mathbf{x}) = y] \)

**Observation:** For any classifier \( h \)

\[
P[h(\mathbf{x}) = y | X = \mathbf{x}] = P[h(\mathbf{x}) = 0, Y = 0 | X = \mathbf{x}] + P[h(\mathbf{x}) = 1, Y = 1 | X = \mathbf{x}]
\]

\[
= \mathbf{1}[h(\mathbf{x}) = 1] \cdot P[Y = 1 | X = \mathbf{x}] + \mathbf{1}[h(\mathbf{x}) = 0] \cdot P[Y = 0 | X = \mathbf{x}]
\]

\[
= \mathbf{1}[h(\mathbf{x}) = 1] \eta(\mathbf{x}) + \mathbf{1}[h(\mathbf{x}) = 0] (1 - \eta(\mathbf{x}))
\]

So:

\[
P[f(\mathbf{x}) = y | X = \mathbf{x}] - P[g(\mathbf{x}) = y | X = \mathbf{x}]
\]

\[
= \eta(\mathbf{x}) \left[ \mathbf{1}[f(\mathbf{x}) = 1] - \mathbf{1}[g(\mathbf{x}) = 1] \right] + (1 - \eta(\mathbf{x})) \left[ \mathbf{1}[f(\mathbf{x}) = 0] - \mathbf{1}[g(\mathbf{x}) = 0] \right]
\]

\[
= (2 \eta(\mathbf{x}) - 1) \left[ \mathbf{1}[f(\mathbf{x}) = 1] - \mathbf{1}[g(\mathbf{x}) = 1] \right]
\]

\[
\geq 0 \quad \text{By the choice of } f
\]

Integrate over \( X \) to remove the conditional
So... is classification a solved problem?

We know that Bayes classifier is optimal.

So have we solved all classification problems?

Not even close!

Why?

\[
f(\bar{x}) = \arg\max_{y \in \mathcal{Y}} P[Y = y | X = \bar{x}] \]

\[
= \arg\max_{y \in \mathcal{Y}} P[X = \bar{x} | Y = y] \cdot P[Y = y]
\]

How to estimate \( P[Y | X] \) ?

How to estimate \( P[X | Y] \) ?

• How good is the model class?
• Quality of estimation degrades with increase in the dimension of \( X \! \! \)!
• Active area of research!
Classification via Prob. Models: Variation

Naïve Bayes classifier:

\[
\hat{f}(\vec{x}) = \arg \max_{y \in Y} \quad P[X = \vec{x}|Y = y] \cdot P[Y = y]
\]

\[
= \arg \max_{y \in Y} \prod_{j=1}^{d} P[X^{(j)} = x^{(j)}|Y = y] \cdot P[Y = y]
\]

Naïve Bayes assumption: The individual features/measurements are **independent** given the class label.

Advantages:

- Computationally very simple model. Quick to code.

Disadvantages:

- Does not properly capture the interdependence between features, giving bad estimates.
How to evaluate the quality of a classifier?

Your friend claims: “My classifier is better than yours”
How can you evaluate this statement?

Given a classifier $f$, we essentially need to compute:

$$P_{(x,y)}[f(x) = y] = \mathbb{E}_{(x,y)}\left[1[f(x) = y]\right]$$

Accuracy of $f$

But... we don’t know the underlying distribution

We can use training data to estimate...

$$\frac{1}{n} \sum_{i=1}^{n} 1[f(x_i) = y_i]$$

Severely overestimates the accuracy!

Why? Training data is already used to construct $f$, so it is NOT an unbiased estimator.
How to evaluate the quality of a classifier?

General strategy:
- Divide the labelled data into **training** and **test** FIRST
- Only use the training data for learning \( f \)
- Then the test data can be used as an **unbiased estimator** for gauging the predictive accuracy of \( f \)
What we learned...

• Why machine learning
• Basics of Supervised Learning
• Maximum Likelihood Estimation
• Learning a classifier via probabilistic modelling
• Optimality of Bayes classifier
• Naïve Bayes classifier
• How to evaluate the quality of a classifier
Questions?
Next time...

Direct ways of finding the discrimination boundary
Remember

• Visit the course website
  http://www.cs.columbia.edu/~verma/classes/fa17/coms4771/

• Review the basics (prerequisites)

• HW0 is out

• Sign up on Piazza & Gradescope