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?
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)
Machine learning: how?

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.
Let’s get started!
A closer look at some prediction problems...

• Handwritten character recognition:

\[
\begin{array}{cccc}
5 & 6 & 7 & 9
\end{array}
\]

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

• Spam filtering:

\{ \text{spam,} \ \text{not spam} \}

• Object recognition:

\{ \text{building, tree, car, road, sky,} \ldots \}
Commonalities in a prediction problem:

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

To learn: \( f \)

Output: \( y = 5 \in \mathcal{Y} \)
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

---

**Supervised learning**

**Training Phase**

- Labeled training data (\(n\) examples from data)

**Learning Algorithm**

- 'classifier' \(\hat{f}\)

**Testing Phase**

- Unlabeled test data (unseen / future data)

- prediction
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
\((n \text{ examples from data})\)

Learning Algorithm

‘classifier’ \(\hat{f}\)

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

select \(\hat{f}\) from…?
from a pool of models \(\mathcal{F}\)
that maximizes label agreement of the training data

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)
• ...
ok, so given a joint input/output space $\mathcal{X} \times \mathcal{Y}$ where the data is distributed according to the distribution $D(\mathcal{X} \times \mathcal{Y})$ a classifier is simply a (measureable) function of the type $f : \mathcal{X} \rightarrow \mathcal{Y}$

there are plenty of functions of such kind, so which $f$ to pick?

Examples

- $f_1(\bar{x}) := y$ for some fixed $y \in \mathcal{Y}$ for all inputs $\bar{x} \in \mathcal{X}$ ‘Constant’ classifier

- $f_2(\bar{x}) := \begin{cases} 
0 & \text{if } x \geq 5 \\
1 & \text{otherwise} 
\end{cases}$ say $\mathcal{X} = \mathbb{R}$ $\mathcal{Y} = \{0, 1\}$ ‘threshold’ classifier

- $f_3(\bar{x}) := \arg\max_{y \in \mathcal{Y}} P[Y = y]$ ‘Majority class’ classifier
So... there are plenty of functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ to choose from.

What might be a reasonable/good function $f$?

After all, we want to do prediction. It better be the case that we select an $f$ which does well in terms of prediction.

Ideally we want: on any example pair $(x,y)$ from $D$ we want $f$ such that $f(x) = y$.

That is, define **accuracy of a classifier** $f$, ie $\text{acc}(f)$ as

$$\text{acc}(f) := P_{(\vec{x},y)}[f(\vec{x}) = y] = \mathbb{E}_{(\vec{x},y)}[1[f(\vec{x}) = y]]$$

we want a classifier $f$ which maximizes acc.
Let’s study the following classifier

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

aka ‘Bayes’ classifier.

\textit{this classifier is remarkably good!}
Why the particular $f = \text{argmax}_y P[Y|X]$?

Accuracy of a classifier $f$:

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

Assume binary classification (for simplicity):

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

Let:

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

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

Theorem:

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

!!! Bayes classifier is optimal !!!
Optimality of Bayes classifier

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

**Observation:** Fix any \( \bar{x} \in \mathcal{X} \), then for any classifier \( h \)

\[
P[h(\bar{x}) = y | X = \bar{x}] = P[h(\bar{x}) = 1, Y = 1 | X = \bar{x}] + P[h(\bar{x}) = 0, Y = 0 | X = \bar{x}]
= 1[h(\bar{x}) = 1] \cdot P[Y = 1 | X = \bar{x}] + 1[h(\bar{x}) = 0] \cdot P[Y = 0 | X = \bar{x}]
= 1[h(\bar{x}) = 1] \eta(\bar{x}) + 1[h(\bar{x}) = 0] \left(1 - \eta(\bar{x})\right)
\]

So:

\[
P[f(\bar{x}) = y | X = \bar{x}] - P[g(\bar{x}) = y | X = \bar{x}]
= \eta(\bar{x}) \left[1[f(\bar{x}) = 1] - 1[g(\bar{x}) = 1] \right] + (1 - \eta(\bar{x})) \left[1[f(\bar{x}) = 0] - 1[g(\bar{x}) = 0] \right]
= (2\eta(\bar{x}) - 1) \left[1[f(\bar{x}) = 1] - 1[g(\bar{x}) = 1] \right]
\geq 0 \quad \text{By the choice of } f
\]

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?

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

The true data distribution is unknown, so \( P[Y|X] \) is unknown

• Since \( P[Y|X] \) is unknown, need to estimate it from observed data samples
• What about the quality of this estimation?
• There are numerous \( x \in X \) (usually infinite), so how many \( P[Y|X=x] \) do we need to construct a practically viable approximation to the Bayes classifier?
We know there are numerous $P[Y|X]$ (one for each $X=x$), but observe

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

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

Bayes rule

indep. of $y$

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

Class conditional probability model  Class Prior

• We STILL need to estimate $P[X|Y]$ and $P[Y]$ from data, but the number of distributions that need to be estimated is small, since $|Y|$ is tiny (usually 2)
Estimating Densities

Ok so how do we estimate conditional densities like $P[X|Y]$ or densities (like $P[X]$ or $P[Y]$ in general?)

Multiple ways of estimating densities

• If the particular form of the data-density is known/assumed, we can do some sort of **parametric density estimation**
  
  [e.g. likelihood estimation, max a-postiori estimation, mixture models, etc.]

• If one doesn’t want to/ if it is inappropriate to assume a specific distribution, we can do **non-parametric density estimation**
  
  [e.g. kernel density estimation, parzen windows, histograms, etc.]

(each technique has its pros and cons)
Maximum Likelihood Estimation (MLE)

Given some data \( \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n \in X \) i.i.d.

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!

Likelihood \( \mathcal{L}(\theta|X) := P(X|\theta) = P(\tilde{x}_1, \ldots, \tilde{x}_n|\theta) \) i.i.d.

\[ = \prod_{i=1}^{n} p(\tilde{x}_i|\theta) = \prod_{i=1}^{n} p(\tilde{x}_i) \]

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

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

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

Fitting a statistical probability model to heights of females

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

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

Model class: Gaussian models in \( \mathbb{R} \)

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

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

$$\arg \max_{\theta} \mathcal{L}(\theta|X) = \arg \max_{\theta} \log \mathcal{L}(\theta|X) \quad \text{“Log” likelihood}$$

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

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

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

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

Maximizing $\mu$:

$$0 = \nabla_\mu \left( \sum_{i=1}^{n} g_i(\mu, \sigma^2) \right) \quad \Rightarrow \quad \mu_{ML} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Maximizing $\sigma^2$:

$$\sigma_{ML}^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2$$
MLE Example

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: \( \mu = \frac{1}{n} \sum_{i=1}^{n} x_i \) and \( \sigma^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2 \)

What about other model classes?
Other popular probability models

Bernoulli model (coin tosses)  
Multinomial model (dice rolls)  
Poisson model (rare counting events)  
Gaussian model (most common phenomenon)  

Most machine learning data is vector valued!

Multivariate Gaussian Model  

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** $\mathbb{R}^d$

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

$\vec{\mu}$ = mean vector  
$\Sigma$ = Covariance matrix (positive definite)
Back to Classification

So how can we construct a practical classifier inspired from Bayes classifier?

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

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

Class conditional probability model  Class Prior

Class prior:
Since \( Y \) is finite, we can model \( P[Y] \) as a generalized Bernoulli distribution and estimate the parameters from observed data using MLE!

Class conditional:
Use a separate probability model \( P[X | Y=y] \) for individual categories/class-type \( y \)
Again, we can find the appropriate parameters for the models using MLE!
**Task:** learn a classifier to identify an individual as **male** or **female** based on collecting, say, just the height and weight measurements

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

\[
\mathcal{Y} = \{\text{male}, \text{female}\}
\]

For a certain population of interest, the data may be distributed according to

\[
\mathcal{D} = \mathcal{D}(\mathcal{X} \times \mathcal{Y})
\]

which is unknown, but fixed

**Observation:** The following labelled data is collected/observed

\[
(x_1, y_1) \quad (x_2, y_2) \quad (x_3, y_3) \quad (x_4, y_4) \\
((110,5.2), f), ((185,5.11), m), ((220,6.2), m), (200,5.5), f), \ldots \sim_{\text{i.i.d.}} \mathcal{D}(\mathcal{X} \times \mathcal{Y})
\]
Classification via MLE Example

For this task, the Bayes classifier would be

\[ f(\vec{x}) = \arg \max_{y \in \{\text{male}, \text{female}\}} P_{D|X|Y=y}[X = \vec{x} | Y = y] \cdot P_{D|Y}[Y = y] \]

[Note: $D$ is unknown, hence each density needs to be estimated]

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

**Estimating the class priors:**

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

[using Binomial model; parameter estimation via MLE]

**Estimating the class conditionals:**

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

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

[by modelling each $P[X|Y]$ using, say, Bivariate Gaussian; parameter estimation via MLE]
What are we doing geometrically?

Data geometry:

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

- Male data
- 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)
Task: learn a classifier to distinguish \textbf{males} from \textbf{females} based on say height and weight measurements

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

Using \textbf{labelled} training data, learn all the parameters:

Learning \textbf{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 \textbf{class conditionals}:
\[ P[X|Y = \text{male}] = p_{\theta(\text{male})}(X) \]
\[ P[X|Y = \text{female}] = p_{\theta(\text{female})}(X) \]
\[ \theta(\text{male}) = \text{\textbf{MLE}} \text{ using only male data} \]
\[ \theta(\text{female}) = \text{\textbf{MLE}} \text{ using only female data} \]
We just made our first predictor $f$!
Classification via Prob. Models: Variation

Naïve Bayes classifier:

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

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

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:

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

Accuracy of $f$

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

We can use training data to estimate...

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

**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