Artificial Intelligence (AI)

- Building “intelligent systems”
- Lots of parts to intelligent behavior

RoboCup

Darpa GC (Stanley)

Chess (Deep Blue v. Kasparov)
Machine learning (ML)

- One (important) part of AI
- Making predictions (or decisions)
- Getting better with experience (data)
- Problems whose solutions are “hard to describe”
Machine Learning

Math

Programming

Statistics
Probability
Linear Algebra
Optimization

Data Structures
Algorithms
Computational Complexity
Data Management
Types of prediction problems

- Supervised learning
  - “Labeled” training data
  - Every example has a desired target value (a “best answer”)
  - Reward prediction being close to target

- Classification: a discrete-valued prediction (often: action / decision)
- Regression: a continuous-valued prediction
Types of prediction problems

- **Supervised learning**
- **Unsupervised learning**
  - No known target values
  - No targets = nothing to predict?
  - Reward “patterns” or “explaining features”
  - Often, data mining

---

The Princess Diaries
The Color Purple
Sense and Sensibility
Amadeus
Lethal Weapon
Ocean’s 11
The Lion King
Dumb and Dumber
Independence Day
The Lion King
Braveheart
Amadeus
Sense and Sensibility
romance?
serious?
escapist?
action?
Types of prediction problems

- Supervised learning
- Unsupervised learning
- **Semi-supervised learning**
  - Similar to supervised
  - some data have unknown target values

- Ex: medical data
  - Lots of patient data, few known outcomes
- Ex: image tagging
  - Lots of images on Flikr, but only some of them tagged
Types of prediction problems

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

- "Indirect" feedback on quality
  - No answers, just "better" or "worse"
  - Feedback may be delayed
Summary

What is machine learning?
- Computer science + Math (Optimization & Statistics)
- How do we learn from data to improve performance

Types of machine learning
- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
Data exploration

- Machine learning is a data science
  - Look at the data; get a “feel” for what might work

- What types of data do we have?
  - Binary values? (spam; gender; …)
  - Categories? (home state; labels; …)
  - Integer values? (1..5 stars; age brackets; …)
  - (nearly) real values? (pixel intensity; prices; …)

- Are there missing data?

- “Shape” of the data? Outliers?
Scientific software

- **Python**
  - Numpy, MatPlotLib, SciPy…

- **Matlab**
  - Octave (free)

- **R**
  - Used mainly in statistics

- **C++**
  - For performance, not prototyping

- And other, more specialized languages for modeling…
Representing data

• Example: Fisher’s “Iris” data

• Three different types of iris
  – “Class”, $y$

• Four “features”, $x_1, \ldots, x_4$
  – Length & width of
    sepals & petals

• 150 examples (data points)
# Intro to Basic Terminology and Notations

Samples (instances, observations)

<table>
<thead>
<tr>
<th></th>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>Setosa</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>Setosa</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>50</td>
<td>6.4</td>
<td>3.5</td>
<td>4.5</td>
<td>1.2</td>
<td>Versicolor</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>150</td>
<td>5.9</td>
<td>3.0</td>
<td>5.0</td>
<td>1.8</td>
<td>Virginica</td>
</tr>
</tbody>
</table>

Features (attributes, measurements, dimensions)

Class labels (targets)

Petal

Sepal
Representing the data in Python

- Have $m$ observations (data points)
  \[ \{ x^{(1)}, \ldots, x^{(m)} \} \]

- Each observation is a vector consisting of $n$ features
  \[ x^{(j)} = [x_1^{(j)}, x_2^{(j)}, \ldots, x_n^{(j)}] \]

- Often, represent this as a “data matrix”
  \[
  X = \begin{bmatrix}
  x^{(1)}_1 & \ldots & x^{(1)}_n \\
  \vdots & \ddots & \vdots \\
  x^{(m)}_1 & \ldots & x^{(m)}_n 
  \end{bmatrix}
  \]

```python
import numpy as np  #import numpy
iris = np.genfromtxt("data/iris.txt",delimiter=None)
X = iris[:,0:4]    # load data and split into features, targets
Y = iris[:,4]
print(X.shape)     # 150 data points; 4 features each
(150, 4)
```
Basic statistics

- Look at basic information about features
  - Average value? (mean, median, etc.)
  - “Spread”? (standard deviation, etc.)
  - Maximum / Minimum values?

```python
print(np.mean(X, axis=0))  # compute mean of each feature
[ 5.8433  3.0573  3.7580  1.1993 ]
print(np.std(X, axis=0))  # compute standard deviation of each feature
[ 0.8281  0.4359  1.7653  0.7622 ]
print(np.max(X, axis=0))  # largest value per feature
[ 7.9411  4.3632  6.8606  2.5236 ]
print(np.min(X, axis=0))  # smallest value per feature
[ 4.2985  1.9708  1.0331  0.0536 ]
```
**Count the data falling in each of \( K \) bins**

- “Summarize” data as a length-\( K \) vector of counts (& plot)
- Value of \( K \) determines “summarization”; depends on \# of data
  - \( K \) too big: every data point falls in its own bin; just “memorizes”
  - \( K \) too small: all data in one or two bins; oversimplifies

```python
% Histograms in MatPlotLib
import matplotlib.pyplot as plt
X1 = X[:,0]  # extract first feature
Bins = np.linspace(4,8,17)  # use explicit bin locations
plt.hist(X1, bins=Bins)  # generate the plot
```
Scatterplots

- Illustrate the relationship between two features

```matlab
% Plotting in MatPlotLib
plt.plot(X[:,0], X[:,1], 'b.);  % plot data points as blue dots
```
Scatterplots

- For more than two features we can use a pair plot:
Supervised learning and targets

- Supervised learning: predict target values
- For discrete targets, often visualize with color

```python
plt.hist( [X[Y==c,1] for c in np.unique(Y)], bins=20, histtype='barstacked')
ml.histy(X[:,1], Y, bins=20)

colors = ['b','g','r']
for c in np.unique(Y):
    plt.plot( X[Y==c,0], X[Y==c,1], 'o', color=colors[int(c)] )
```

```python
ml.histy(X[:,1], Y, bins=20)
```
How does machine learning work?

- “Meta-programming”
  - Predict – apply rules to examples
  - Score – get feedback on performance
  - Learn – change predictor to do better

Program (“Learner”)
Characterized by some “parameters” $\theta$
Procedure (using $\theta$) that outputs a prediction

Training data (examples)
Features (x)
Feedback / Target values (y)

Learning algorithm
Change $\theta$
Improve performance

“predict”
Score performance (“cost function”)

“train”
How does machine learning work?

- **Notation**
  - Features \( x \)
  - Targets \( y \)
  - Predictions \( \hat{y} = f(x ; \theta) \)
  - Parameters \( \theta \)

**Program (“Learner”)**
Characterized by some “parameters” \( \theta \)
Procedure (using \( \theta \)) that outputs a prediction

**Training data (examples)**
- Features \( x \)
- Feedback / Target values \( y \)

**Learning algorithm**
Change \( \theta \)
Improve performance

“train”

“predict”

Score performance (“cost function”)
Suggests a relationship between \( x \) and \( y \)

**Prediction**: new \( x \), what is \( y \)?
Nearest neighbor regression

- Find training datum \( x^{(i)} \) closest to \( x^{(new)} \)
- Predict \( y^{(i)} \)
Nearest neighbor regression

- Defines a function \( f(x) \) implicitly
- "Form" is piecewise constant

"Predictor":
Given new features:
Find nearest example
Return its value
Linear regression

- Define form of function $f(x)$ explicitly
- Find a good $f(x)$ within that family

“Predictor”:
Evaluate line:
$$r = \theta_0 + \theta_1 x_1$$
return $r$
Measuring error

\[ \varepsilon^{(i)} = y^{(i)} - \hat{y}(x^{(i)}) \]

\[ \text{MSE} = \frac{1}{m} \sum_{i} \left( y^{(i)} - \hat{y}(x^{(i)}) \right)^2 \]
Regression vs. Classification

**Regression**
- Features $x$
- Real-valued target $y$
- Predict continuous function $\hat{y}(x)$

**Classification**
- Features $x$
- Discrete class $c$ (usually 0/1 or +1/-1)
- Predict discrete function $\hat{y}(x)$

![Graphs showing regression and classification examples](image)
Classification
Classification

$$ERR = \frac{1}{m} \sum_i \left[ y^{(i)} \neq \hat{y}(x^{(i)}) \right]$$