Lecture 3:
- Kernel Regression
- Distance Metrics
- Curse of Dimensionality
- Linear Regression
Administrative

• **Assignment 1** will be out on Friday

• It is due **October 20** (i.e. in two weeks).

• It includes
  - Pencil-and-paper derivations
  - Implementing kNN classifier
  - numpy/Python code

• **Note:** Lecture slides are not enough, you should also read related book chapters!
Recall from last time… Nearest Neighbors

Example dataset: CIFAR-10
10 labels
50,000 training images
10,000 test images.

• Very simple method
• Retain all training data
  - It can be slow in testing
  - Finding NN in high dimensions is slow
• Metrics are very important
• Good baseline

For every test image (first column), examples of nearest neighbors in rows

adoption from Fei-Fei Li & Andrej Karpathy & Justin Johnson
Classification

• Input: $X$
  - Real valued, vectors over real.
  - Discrete values (0,1,2,...)
  - Other structures (e.g., strings, graphs, etc.)

• Output: $Y$
  - Discrete (0,1,2,...)

slide by Aarti Singh and Barnabas Poczos
Regression

• Input: $X$
  - Real valued, vectors over real.
  - Discrete values (0, 1, 2, ...)
  - Other structures (e.g., strings, graphs, etc.)

• Output: $Y$
  - Real valued, vectors over real.

Stock Market Prediction

Y = ?
X = Feb01
What should I watch tonight?

**The Martian (2015)**

- **Rating:** 8.1/10
- **Ratings:** 8.1/10 from 271,829 users
- **Metascore:** 80/100
- **Reviews:** 750 user, 499 critic, 46 from Metacritic.com

*During a manned mission to Mars, Astronaut Mark Watney is presumed dead after a fierce storm and left behind by his crew. But Watney has survived and finds himself stranded and alone on the hostile planet. With only meager supplies, he must draw upon his ingenuity, wit and spirit to subsist and find a way to signal to Earth that he is alive.*

- **Director:** Ridley Scott
- **Writers:** Drew Goddard (screenplay), Andy Weir (book)
- **Stars:** Matt Damon, Jessica Chastain, Kristen Wiig

[See More on IMDb Pro »](https://www.imdb.com/title/tt2726907/)
What should I watch tonight?

[Image of IMDb page for Point Break (2015)]

**Title:** Point Break (2015)

**Rating:** 5.4/10 from 7,322 users  Metascore: 34/100

**Reviews:** 60 user | 84 critic | 19 from Metacritic.com

**Summary:**
A young FBI agent infiltrates an extraordinary team of extreme sports athletes he suspects of masterminding a string of unprecedented, sophisticated corporate heists. "Point Break" is inspired by the classic 1991 hit.

**Director:** Ericson Core

**Writers:** Kurt Wimmer (screenplay), Rick King (story), 5 more credits »

**Stars:** Édgar Ramírez, Luke Bracey, Ray Winstone | See full cast and crew »

[Image of IMDb page for Point Break (2015)]
What should I watch tonight?

Predict this automatically!
Today

• Kernel regression
  – nonparametric

• Distance metrics

• Linear regression (more on Thursday)
  – parametric
    – simple model
Simple 1-D Regression

- Circles are data points (i.e., training examples) that are given to us.
- The data points are uniform in $x$, but may be displaced in $y$
  
  \[ t(x) = f(x) + \varepsilon \]
  
  with $\varepsilon$ some noise.
- In green is the “true” curve that we don’t know.
Kernel Regression
K-NN for Regression

• Given: Training data \{((x_1, y_1), ..., (x_n, y_n))\}
  – Attribute vectors: $x_i \in X$
  – Target attribute $y_i \in \mathcal{R}$

• Parameter:
  – Similarity function: $K : X \times X \rightarrow \mathcal{R}$
  – Number of nearest neighbors to consider: $k$

• Prediction rule
  – New example $x'$
  – K-nearest neighbors: $k$ train examples with largest $K(x_i, x')$

\[
h(x') = \frac{1}{k} \sum_{i \in \text{knn}(x')} y_i
\]
1-NN for Regression

Figure Credit: Carlos Guestrin
1-NN for Regression

- Often bumpy (overfits)

Figure Credit: Andrew Moore
9-NN for Regression

- Often bumpy (overfits)
Multivariate distance metrics

• Suppose the input vectors $x_1, x_2, \ldots x_N$ are two dimensional:

$$x_1 = (x_{11}, x_{12}), \quad x_2 = (x_{21}, x_{22}), \ldots x_N = (x_{N1}, x_{N2}).$$

• One can draw the nearest-neighbor regions in input space.

Dist$(x_i, x_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2$

Dist$(x_i, x_j) = (x_{i1} - x_{j1})^2 + (3x_{i2} - 3x_{j2})^2$

The relative scalings in the distance metric affect region shapes

Slide Credit: Carlos Guestrin
Example: Choosing a restaurant

• In everyday life we need to make decisions by taking into account lots of factors
• The question is what weight we put on each of these factors (how important are they with respect to the others).

<table>
<thead>
<tr>
<th>Reviews (out of 5 stars)</th>
<th>$</th>
<th>Distance</th>
<th>Cuisine (out of 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>30</td>
<td>21</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>15</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td>53</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>
Euclidean distance metric

\[ D(x, x') = \sqrt{\sum_i \sigma_i^2 (x_i - x'_i)^2} \]

Or equivalently,

\[ D(x, x') = \sqrt{(x_i - x'_i)^T A (x_i - x'_i)} \]

where

\[ A = \begin{bmatrix}
\sigma_1^2 & 0 & \cdots & 0 \\
0 & \sigma_2^2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sigma_N^2 \\
\end{bmatrix} \]
Notable distance metrics (and their level sets)

Scaled Euclidian ($L_2$)

Mahalanobis (non-diagonal $A$)
Minkowski distance

\[ D = \left( \sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \]
Notable distance metrics (and their level sets)

- Scaled Euclidian ($L_2$)
- $L_1$ norm (absolute)
- $L_{\infty}$ (max) norm

Slide Credit: Carlos Guestrin
Parametric vs Non-parametric Models

• Does the capacity (size of hypothesis class) grow with size of training data?
  – Yes = Non-parametric Models
  – No = Parametric Models
Weighted K-NN for Regression

- Given: Training data \{ (x_1,y_1), \ldots, (x_n,y_n) \}
  - Attribute vectors: \( x_i \in X \)
  - Target attribute \( y_i \in \mathcal{R} \)

- Parameter:
  - Similarity function: \( K : X \times X \rightarrow \mathcal{R} \)
  - Number of nearest neighbors to consider: \( k \)

- Prediction rule
  - New example \( x' \)
  - K-nearest neighbors: \( k \) train examples with largest \( K(x_i,x') \)

\[
h(\bar{x}') = \frac{\sum_{i \in \text{knn}(\bar{x}')} y_i K(\bar{x}_i, \bar{x}')}{\sum_{i \in \text{knn}(\bar{x}')} K(\bar{x}_i, \bar{x}')}\]
Kernel Regression/Classification

Four things make a memory based learner:

- **A distance metric**
  - Euclidean (and others)

- **How many nearby neighbors to look at?**
  - All of them

- **A weighting function (optional)**
  - \( w_i = \exp\left(-\frac{d(x_i, \text{query})^2}{\sigma^2}\right) \)
  - Nearby points to the query are weighted strongly, far points weakly. The \( \sigma \) parameter is the Kernel Width. Very important.

- **How to fit with the local points?**
  - Predict the weighted average of the outputs
    \( \text{predict} = \frac{\sum w_i y_i}{\sum w_i} \)
Weighting/Kernel functions

\[ w_i = \exp(-d(x_i, \text{query})^2 / \sigma^2) \]

(Our examples use Gaussian)
Effect of Kernel Width

• What happens as $\sigma \to \infty$?

• What happens as $\sigma \to 0$?
Problems with Instance-Based Learning

• Expensive
  - No Learning: most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - Must use tricks like approximate nearest neighbour search
Problems with Instance-Based Learning

- Expensive
  - No Learning: most real work done during testing
  - For every test sample, must search through all dataset – very slow!
  - Must use tricks like approximate nearest neighbour search

- Doesn’t work well when large number of irrelevant features
  - Distances overwhelmed by noisy features
Problems with Instance-Based Learning

• Expensive
  – No Learning: most real work done during testing
  – For every test sample, must search through all dataset – very slow!
  – Must use tricks like approximate nearest neighbour search

• Doesn’t work well when large number of irrelevant features
  – Distances overwhelmed by noisy features

• Curse of Dimensionality
  • Distances become meaningless in high dimensions
Curse of Dimensionality

- Consider applying a KNN classifier/regressor to data where the inputs are uniformly distributed in the $D$-dimensional unit cube.

- Suppose we estimate the density of class labels around a test point $x$ by “growing” a hyper-cube around $x$ until it contains a desired fraction $f$ of the data points.

- The expected edge length of this cube will be $e_D(f) = f^{1/D}$.

- If $D = 10$, and we want to base our estimate on 10% of the data, we have $e_{10}(0.1) = 0.8$, so we need to extend the cube 80% along each dimension around $x$.

- Even if we only use 1% of the data, we find $e_{10}(0.01) = 0.63$. — no longer very local
Linear Regression
Simple 1-D Regression

- Circles are data points (i.e., training examples) that are given to us
- The data points are uniform in $x$, but may be displaced in $y$
  \[ t(x) = f(x) + \varepsilon \]
  with $\varepsilon$ some noise
- In green is the “true” curve that we don’t know
- Goal: We want to fit a curve to these points
Simple 1-D Regression

- **Key Questions:**
  - How do we parametrize the model (the curve)?
  - What **loss (objective) function** should we use to judge fit?
  - How do we optimize fit to unseen test data (generalization)?
Example: Boston House Prizes

- Estimate median house price in a neighborhood based on neighborhood statistics
- Look at first (of 13) attributes: per capita crime rate
- Use this to predict house prices in other neighborhoods
- Is this a good input (attribute) to predict house prices?

https://archive.ics.uci.edu/ml/datasets/Housing
Represent the data

- Data described as pairs \( D = \{(x_1, t_1), (x_2, t_2), \ldots, (x_N, t_N)\} \)
  - \( x \) is the **input feature** (per capita crime rate)
  - \( t \) is the **target output** (median house price)
  - \((i)\) simply indicates the training examples (we have \( N \) in this case)

- Here \( t \) is continuous, so this is a **regression problem**

- Model outputs \( y \), an estimate of \( t \)

  \[
  y(x) = w_0 + w_1 x
  \]

- What type of **model** did we choose?

- Divide the dataset into training and testing examples
  - Use the training examples to construct hypothesis, or function approximator, that maps \( x \) to predicted \( y \)
  - Evaluate hypothesis on test set
A simple model typically does not exactly fit the data — lack of fit can be considered noise

**Sources of noise:**
- Imprecision in data attributes (input noise, e.g. noise in per-capita crime)
- Errors in data targets (mislabling, e.g. noise in house prices)
- Additional attributes not taken into account by data attributes, affect target values (latent variables). In the example, what else could affect house prices?
- Model may be too simple to account for data targets
Least-Squares Regression

Define a model

\[ y(x) = w_0 + w_1 x \]

Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

Linear model:

\[ \mathbf{w} = \mathbf{X}^{-1} \mathbf{t} \]

For a particular hypothesis (\( y(x) \) defined by a choice of \( \mathbf{w} \), drawn), what does the loss represent geometrically?

How do we obtain weights \( \mathbf{w} = (w_0, w_1) \)?

For the linear model, what kind of a function is \( \mathbf{w} \)?

\[
y(x) = \text{function}(x, \mathbf{w})
\]
Least-Squares Regression

• Define a model

  Linear: \( y(x) = \text{function}(x, w) \)

  Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

  For a particular hypothesis (\( y(x) \) defined by a choice of \( w \), drawn in red), what does the loss represent geometrically?

  How do we obtain weights \( w = (w_0, w_1) \)?
• Define a model

Linear: \[ y(x) = w_0 + w_1 x \]
Least-Squares Regression

- Define a model
  
  Linear: $y(x) = w_0 + w_1 x$

- Standard loss/cost/objective function measures the squared error between $y$ and the true value $t$

  $$\ell(w) = \sum_{n=1}^{N} \left[ t^{(n)} - y(x^{(n)}) \right]^2$$
Least-Squares Regression

- Define a model
  Linear: \( y(x) = w_0 + w_1 x \)

- Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

Linear model: \( \ell(w) = \sum_{n=1}^{N} \left[ t^{(n)} - (w_0 + w_1 x^{(n)}) \right]^2 \)
Least-Squares Regression

- Define a model
  
  **Linear:** \( y(x) = w_0 + w_1 x \)

- Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

  Linear model: 
  
  \[
  \ell(w) = \sum_{n=1}^{N} \left[ t^{(n)} - (w_0 + w_1 x^{(n)}) \right]^2
  \]

- For a particular hypothesis \( y(x) \) defined by a choice of \( w \), drawn in red), what does the loss represent geometrically?
Least-Squares Regression

• Define a model
  
  Linear: \( y(x) = w_0 + w_1 x \)

• Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

  Linear model: \( \ell(w) = \sum_{n=1}^{N} \left[ t^{(n)} - (w_0 + w_1 x^{(n)}) \right]^2 \)

• The loss for the red hypothesis is the **sum of the squared vertical errors** (squared lengths of green vertical lines)
Least-Squares Regression

• Define a model

  Linear: \[ y(x) = w_0 + w_1 x \]

• Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

  Linear model:

  \[
  \ell(w) = \sum_{n=1}^{N} \left[ t^{(n)} - (w_0 + w_1 x^{(n)}) \right]^2
  \]

• How do we obtain weights \( w = (w_0, w_1) \)?
Least-Squares Regression

- Define a model
  
  **Linear:**  
  \[ y(x) = w_0 + w_1 x \]

- Standard loss/cost/objective function measures the squared error between \( y \) and the true value \( t \)

  **Linear model:**  
  \[ \ell(w) = \sum_{n=1}^{N} \left[ t^{(n)} - (w_0 + w_1 x^{(n)}) \right]^2 \]

- How do we obtain weights \( w = (w_0, w_1) \)? Find \( w \) that minimizes loss \( \ell(w) \)
Next Lecture:
Least Squares Optimization,
Model complexity, and
Regularization