Today’s quiz

Q1. Consider the following dataset:
   a) What is the entropy of this set of examples?
   b) What is the information gain of feature “Weather”?
   c) What is the information gain of feature “Day”?

| Day | Weather | GoHiking?
|-----|---------|-----------
| Mon | Sunny  | No        |
| Tues| Cloudy | No        |
| Wed | Rain    | No        |
| Thurs| Rain  | No        |
| Fri  | Sunny  | No        |
| Sat  | Sunny  | No        |
| Sun  | Sunny  | Yes       |

Q2. Give a decision tree that correctly represents the following Boolean function: \( Y = [X_1 \text{ AND } X_2] \text{ OR } [X_2 \text{ AND } X_3] \)
News

• Don’t forget to do your Peer-review 1 report
  – Go to https://cmt.research.microsoft.com/COMP598_2015
  – Select Your Role: Reviewer
  – Fill-out review form by Oct. 9th.  **Currently 15 / 80 reviews submitted.**
  – Comments should be detailed, constructive, insightful, fair.

• Starting working on Mini-project #2.
  – Go to https://inclass.kaggle.com/c/dialogue-classification
  • 6 submissions so far.
  – **Final submissions due by Oct.21, 7:59pm.**
  – Report due on CMT by Oct.21, 11:59pm.

Tips for analyzing text

• Natural Language toolkit: http://www.nltk.org/

• Common features:
  – Bag of words
  – Term frequency – inverse document frequency (TF-IDF)
  – Hashing
  – Word embeddings

• Dimensionality reduction (don’t consider all words, limit size of hash table / embedding dimension).

• Evaluate your feature choice on a validation set.
A complete (artificial) example

• An artificial binary classification problem with two real-valued input features:

![Decision Tree Graph](image)

A complete (artificial) example

• The decision tree, graphically:

![Decision Tree Graph](image)
A complete (artificial) example

• A decision tree or a linear classifier?

What label should we predict for this example?
Parametric supervised learning

- So far, we have assumed that we have a data set of labeled examples.
- From this, we learn a parameter vector of a fixed size such that some error measure based on the training data is minimized.
- These methods are called parametric, and main goal is to summarize the data using the parameters.
  - Parametric methods are typically global = one set of parameters for the entire data space.
- What if we just remembered the data? When new instances arrive, compare them with the data so far, find the most similar examples, and use them to find the answer.

Non-parametric learning methods

- Key idea: just store all training examples $< x_i, y_i >$.
- When a query is made, compute the value of the new instance based on the values of the closest (most similar) points.
- Requirements:
  - A distance function.
  - How many closest points (neighbors) to look at?
  - How do we compute the value of the new point based on the existing values?
What kind of distance metric?

- Euclidean distance.
- Maximum minimum difference along any axis.
- Weighted Euclidean distance (with weights based on domain knowledge): 
  \[ d(x, x') = \sum_{j=1}^{m} w_j (x_j - x'_j)^2 \]
- An arbitrary distance or similarity function \( d \), specific for the application at hand (works best, if you have one.)

Simple idea: Connect the dots!
Simple idea: Connect the dots!

One-nearest neighbor

- **Given**: Training data $X$, distance metric $d$ on $X$.

- **Learning**: Nothing to do! (Just store the data).

- **Prediction**: For $x \in X$
  
  Find nearest training sample $x$. 

  $$i^* = \arg\min_i d(x, x)$$

  Predict $y = y_i$
What does the approximator look like?

- Nearest-neighbor does not explicitly compute decision boundaries.
- But the effective decision boundaries are a subset of the Voronoi diagram for the training data.
- Each decision boundary is a line segment that is equidistant between two points of opposite classes.

Choice of distance metric is important!

Left: both attributes weighted equally; Right: second attributes weighted more
Distance metric tricks

• You may need to do preprocessing:
  – Scale the input dimensions (or normalize them).
  – Remove noisy inputs.
  – Determine weights for attributes based on cross-validation (or information-theoretic methods).

• Distance metric is often domain-specific.
  – E.g. string edit distance in bioinformatics.
  – E.g. trajectory distance in time series models for walking data.

• Distance can be learned sometimes.

---

k-nearest neighbor (kNN)

• **Given**: Training data $X$, distance metric $d$ on $X$.

• **Learning**: Nothing to do! (Just store the data).

• **Prediction**:
  – For $x \in X$, find the $k$ nearest training samples to $x$.
  – Let their indices be $i_1, i_2, \ldots, i_k$.
  – Predict: $y = \text{mean/median of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for regression
    $y = \text{majority of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for classification, or
    empirical probability of each class.
Classification, 2-nearest neighbor

Classification, 3-nearest neighbor
Classification, 5-nearest neighbor

![Graph showing 5-nearest neighbor classification]

Classification, 10-nearest neighbor

![Graph showing 10-nearest neighbor classification]
Classification, 15-nearest neighbor

Classification, 20-nearest neighbor
Regression, 2-nearest neighbor

Regression, 3-nearest neighbor
Regression, 5-nearest neighbor

Regression, 10-nearest neighbor
Bias-variance trade-off

• What happens if $k$ is low?
  Very non-linear functions can be approximated, but we also capture the noise in the data. Bias is low, variance is high.

• What happens if $k$ is high?
  The output is much smoother, less sensitive to data variation. High bias, low variance.

• A validation set can be used to pick the best $k$.

Limitations of $k$-nearest neighbor (kNN)

• A lot of discontinuities!

• Sensitive to small variations in the input data.

• Can we fix this but still keep it (fairly) local?
**k-nearest neighbor (kNN)**

- **Given**: Training data $X$, distance metric $d$ on $X$.

- **Learning**: Nothing to do! (Just store the data).

- **Prediction**:
  - For $x \in X$, find the $k$ nearest training samples to $x$.
  - Let their indices be $i_1, i_2, \ldots, i_k$.
  - Predict: $y = \text{mean/median of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for regression

  $y = \text{majority of } \{y_{i_1}, y_{i_2}, \ldots, y_{i_k}\}$ for classification, or

  empirical probability of each class.

**Distance-weighted (kernel-based) NN**

- **Given**: Training data $X$, distance metric $d$ on $X$, weighting function $w : R \to R$.

- **Learning**: Nothing to do! (Just store the data).

- **Prediction**:
  - Given input $x$.
  - For each $x_i$, compute $w_i = w(d(x, x_i))$.
  - Predict: $y = \sum_i w_i y_i / \sum_i w_i$.

- **How should we weigh the distances?**
Some weighting functions

\[ \frac{1}{d(x_i, x)} \quad \frac{1}{d(x_i, x)^2} \quad \frac{1}{c + d(x_i, x)^2} \quad e^{\frac{d(x_i, x)^2}{\sigma^2}} \]

Gaussian weighting, small \( \sigma \)

Gaussian-weighted nearest neighbor with \( \sigma=0.25 \)
Gaussian weighting, medium $\sigma$

Gaussian weighting, large $\sigma$

All examples get to vote! Curve is smoother, but perhaps too smooth?
Locally-weighted linear regression

- Weighted linear regression: different weights in the error function for different points.

- **Locally** weighted linear regression: weights depend on the distance to the query point.
  - Use a local linear fit (rather than an average) around the query point.
  - *If the distance metric is well tuned*, it can lead to very good results (can represent non-linear functions easily and faithfully).

**Note:** This is not the same as distance-weighted NN. Here we re-weight the points based on distance to query point, then perform linear regression.

http://www.autonlab.org/autonweb/14691.html

Locally-weighted linear regression with Gaussian kernels

- Width of the kernel is crucial for good results.
  - Kernel is too wide -> the number of points if too big we we “linearize” the function too much.
  - Kernel is too narrow -> too few points are included and the regression line is subject to high variance.
  - Cross-validation can be used to tune the kernel width.
Lazy vs eager learning

- **Lazy learning**: Wait for query before generalization.
  - E.g. Nearest neighbour.

- **Eager learning**: Generalize before seeing query.
  - E.g. Logistic regression, LDA, decision trees, neural networks.

Pros and cons of lazy and eager learning

- Eager learners must create global approximation.
- Lazy learners create many local approximations.
- If they use the same hypothesis space, a lazy learner can represent more complex functions (e.g., consider $H = \text{linear function}$).

- Lazy learning has much faster training time.
- Eager learner does the work off-line, summarizes lots of data with few parameters.

- Lazy learner typically has slower query answering time (depends on number of instances and number of features) and requires more memory (must store all the data).
Scaling up

- **kNN in high-dimensional feature spaces?**
  - In high dim spaces, the distance between near and far points appears similar.
  - A few points ("hubs") show up repeatedly in the top kNN [Radovanovic et al., 2009].

- **kNN with larger number of datapoints?**
  - Can be implemented efficiently, $O(\log n)$ at retrieval time, if we use smart data structures:
    - Condensation of the dataset.
    - Hash tables in which the hashing function is based on the distance metric.
    - KD-trees (Tutorial: http://www.autonlab.org/autonweb/14665)

When to use instance-based learning

- Instances map to points in $\mathbb{R}^n$. Or else a given distance metric.
- Not too many attributes per instance (e.g. <20), otherwise all points start to look at a similar distance, and noise becomes a big issue.
- **Easily fooled by irrelevant attributes** (for most distance metrics.)

- Can produce confidence intervals in addition to the prediction.

- Provides a variable resolution approximation (based on density of points).
Application

Hays & Efros, Scene Completion Using Millions of Photographs, CACM, 2008.


What you should know

• Difference between **eager** vs **lazy** learning.

• Key idea of **non-parametric** learning.

• The **k-nearest neighbor** algorithm for classification and regression, and its properties.

• The distance-weighted NN algorithm and locally-weighted linear regression.