Lecture 2: Memorizer and Nearest Neighbor

Introduction to Learning and Analysis of Big Data
The new waiter

- The coffee shop *GuessYou* prides itself in serving the customers their favorite drinks before the customers order them.
- New waiters are allowed to ask customers what is their favorite drink for **one week**.
- After this week they waiters need to decide **what to serve each customer**, without asking any more customers.
- The waiters’ tips depend on how well they do!
Supervised learning

- In **supervised learning**, the learner gets to learn from **examples** with their true **labels**.
- The learner then needs to devise a **prediction rule**.
- The prediction rule will be used to **predict** the labels of future examples.
- The success of the prediction rule is measured by how **accurately** it predicts the labels of future examples.
- In the coffee shop example:
  - Waiter = learner (learning algorithm)
  - Customers = examples
  - Favorite drinks = labels
  - Tips after first week = prediction success
Supervised learning

- There are two distinct phases:
  - **Training phase**: The learner receives labeled examples, and outputs a prediction rule.
    - The set of labeled examples that the learner receives is the **training sample**
  - **Test phase**: The prediction rule is used on new examples.
A simple learning algorithm for the waiter

- The waiter can **memorize** what each customer likes.
- Over the first week the waiter writes down what every customer liked.
- After the first week, every customer that is in the list gets the same drink that they asked for before.
- For customers not in the list: the waiter will just guess something.
- How good is this algorithm?
A formal description

- $\mathcal{X}$ - the set of all possible examples
- $\mathcal{Y}$ - the set of all possible labels
- A training sample: $S = ((x_1, y_1), \ldots, (x_m, y_m))$
- A **learning algorithm** is any algorithm that has:
  - **Input**: A training sample $S$
  - **Output**: A prediction rule $\hat{h}_S : \mathcal{X} \rightarrow \mathcal{Y}$. 
The **Memorize** algorithm

### Memorize algorithm

**input** A training sample $S$

**output** A function $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.

1. Set $\hat{h}_S = f^\text{mem}_S$ where $f^\text{mem}_S$ is defined as:

$$\forall x \in \mathcal{X}, \quad f^\text{mem}_S(x) = \begin{cases} y & (x, y) \in S \\ \text{a random label} & \text{otherwise} \end{cases}$$

In the second case, the label is drawn **uniformly at random** from $\mathcal{Y}$.

•
How good is the Memorize algorithm?

- It depends...
  - Perhaps the coffee shop has only a single customer;
  - Perhaps each customer visits the coffee shop only once;
  - Perhaps after the first week the coffee shop moved and the regular customers were replaced by new ones.

To analyze the algorithm, we need to make some assumptions. We will use these assumptions throughout most of this course.

The main assumptions:

- There is some distribution of examples and labels, and each labeled example is an independent random draw from this distribution.
- The algorithm does not know this distribution.
A distribution over labeled examples

The coffee shop example:

- $\mathcal{X} = \{"Monica", "Phoebe", "Ross", "Joey", "Chandler", "Rachel"\}$
- $\mathcal{Y} = \{"Juice", "Tea", "Coffee"\}$
- $\mathcal{D}$ is a distribution over $\mathcal{X} \times \mathcal{Y}$.
- $\mathcal{D}$ defines a probability for every pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$.
- This is denoted $\mathbb{P}_{(X, Y) \sim \mathcal{D}}[(X, Y) = (x, y)]$, or $\mathcal{D}((x, y))$.

A possible $\mathcal{D}$:

<table>
<thead>
<tr>
<th></th>
<th>Juice</th>
<th>Tea</th>
<th>Coffee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monica</td>
<td>0</td>
<td>10%</td>
<td>0</td>
</tr>
<tr>
<td>Phoebe</td>
<td>20%</td>
<td>10%</td>
<td>0</td>
</tr>
<tr>
<td>Ross</td>
<td>0</td>
<td>0</td>
<td>20%</td>
</tr>
<tr>
<td>Joey</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Chandler</td>
<td>20%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rachel</td>
<td>5%</td>
<td>5%</td>
<td>10%</td>
</tr>
</tbody>
</table>
A distribution over labeled examples

- According to our assumption, each pair of a customer and its desired drink is chosen randomly according to $\mathcal{D}$.

- Our training sample is $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where each $(x_i, y_i)$ is drawn independently from $\mathcal{D}$.
  - This is denoted $S \sim \mathcal{D}^m$.

- After the training time ends, random pairs $(x, y)$ (customers with desired drinks) continue to be drawn from $\mathcal{D}$, but this time the waiter only observes $x$, and uses $\hat{h}_S$ to predict $y$.

- How to measure the success of $\hat{h}_S$?
Measuring the success of the algorithm

- At test time, if the customers keep coming indefinitely, the fraction of cases in which the waiter gets the wrong drink is the **probability of error** of the waiter’s prediction rule:

\[
\text{err}(\hat{h}_S, \mathcal{D}) := \mathbb{P}_{(X,Y) \sim \mathcal{D}}[\hat{h}_S(X) \neq Y].
\]

- Suppose that each customer has only one drink that they order.

<table>
<thead>
<tr>
<th></th>
<th>Juice</th>
<th>Tea</th>
<th>Coffee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monica</td>
<td>0</td>
<td>10%</td>
<td>0</td>
</tr>
<tr>
<td>Phoebe</td>
<td>30%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ross</td>
<td>0</td>
<td>0</td>
<td>20%</td>
</tr>
<tr>
<td>Joey</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Chandler</td>
<td>20%</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Rachel</td>
<td>0</td>
<td>0</td>
<td>20%</td>
</tr>
</tbody>
</table>

- So \(\mathcal{D}\) has the property of **deterministic label for each example**:

\[
\forall x \in \mathcal{X}, \text{ there is only one } y \in \mathcal{Y} \text{ s.t. } \mathcal{D}((x, y)) > 0.
\]

- In this case, any customer that came in the first week will always receive the correct drink.
Probability of error for the prediction rule of Memorize

- Suppose $|\mathcal{Y}| = k$ (i.e., $k$ possible drinks)
- Suppose deterministic label for each example
- For each customer $x \in \mathcal{X}$, denote by $p_x$ the probability of being the one at the door:
  \[ p_x := \mathbb{P}(X, Y) \sim \mathcal{D} | X = x] = \sum_{y \in \mathcal{Y}} \mathcal{D}((x, y)) \]

Denote $\mathcal{X}_S = \{x | \exists y \text{ s.t. } (x, y) \in S\}$.

- Missing mass: $M_S := \sum_{x \in \mathcal{X} \setminus \mathcal{X}_S} p_x$
  - Probability of being surprised by an unfamiliar customer.
- Memorizer makes a mistake on $x$ when:
  - A new customer arrives: $x \notin \mathcal{X}_S$, and
  - The random label is wrong.
- Probability of error of $\hat{h}_S = f_{\text{mem}}^S$:
  \[
  \text{err}(\hat{h}_S, \mathcal{D}) = \mathbb{P}(X, Y) \sim \mathcal{D} | X \notin \mathcal{X}_S \text{ and wrong random label}
  = \mathbb{P}[X \notin \mathcal{X}_S] \cdot \mathbb{P}[\text{wrong random label} | X \notin \mathcal{X}_S]
  = M_S \cdot \frac{k - 1}{k}.
  \]
Probability of error for the prediction rule of **Memorize**

- Recall: \( k = |\mathcal{Y}| \) (# drinks), \( M_S := \sum_{x \in \mathcal{X} \setminus \mathcal{X}_S} p_x \) (missing mass).
- We saw
\[
\text{err}(\hat{h}_S, \mathcal{D}) = \frac{k - 1}{k} M_S.
\]
- \( M_S \) depends on the **random** training sample \( S \).
- What is the **expected value** of \( M_S \) over random training samples?
- Expected missing mass:
\[
\mathbb{E}_{S \sim D^m}[M_S] = \mathbb{E}_{S \sim D^m}[\sum_{x \in \mathcal{X} \setminus \mathcal{X}_S} p_x] = \mathbb{E}_{S \sim D^m}[\sum_{x \in \mathcal{X}} p_x \cdot I[x \notin \mathcal{X}_S]] \\
= \sum_{x \in \mathcal{X}} p_x \cdot \mathbb{E}[I[x \notin \mathcal{X}_S]] = \sum_{x \in \mathcal{X}} p_x \cdot P[x \notin \mathcal{X}_S] \\
= \sum_{x \in \mathcal{X}} p_x(1 - p_x)^m.
\]
How good is the result?

- \( \text{err}(\hat{h}_S, \mathcal{D}) = \frac{k-1}{k} M_S \), and \( \mathbb{E}[M_S] = \sum_{x \in \mathcal{X}} p_x (1 - p_x)^m \).

- So the expected error of the Memorizer rule is:

\[
\mathbb{E}_{S \sim \mathcal{D}^m}[\text{err}(\hat{h}_S, \mathcal{D})] = \frac{k - 1}{k} \sum_{x \in \mathcal{X}} p_x (1 - p_x)^m.
\]

- Is that a good result?
Is that a good result?

- We got
  \[ \mathbb{E}_{S \sim D^m}[\text{err}(\hat{h}_S, D)] = \frac{k-1}{k} \sum_{x \in \mathcal{X}} p_x (1 - p_x)^m. \]

- For any fixed distribution \( D \), the error \( \to 0 \) when \( m \) grows.
- There is no sample size \( m \) which is good for all distributions.
  - If \( |\mathcal{X}| = N \) and all customers have probability \( p_x = 1/N \):
    \[ \mathbb{E}_{S \sim D^m}[\text{err}(\hat{h}_S, D)] = \frac{k-1}{k} (1 - 1/N)^m \leq \frac{k-1}{k} e^{-m/N}. \]
  - E.g. if \( k = 2 \) and the customers arrive uniformly
    \[ \mathbb{E}_{S \sim D^m}[\text{err}(\hat{h}_S, D)] \leq \frac{1}{2} e^{-m/N}. \]
    but if \( N > 2m \), the expected error is at least 30%.
  - We cannot tell in advance whether one week is enough for the waiter to train its prediction rule.
- We would like distribution-free guarantees: for some sample size \( m \), the learning algorithm will get low error for any distribution.
Does **Memorize** actually learn anything?

- In some sense, **Memorize** doesn’t actually learn anything, because it only “learns” what it already saw.
- It cannot **generalize** to unseen examples.
- What can we do to improve this?
Generalizing to unseen examples

- If the waiter sees a customer that it didn’t see in the training time, but this customer is similar to a customer that it did see, perhaps they have the same favorite drink.

- What do we mean by “similar”?
  - Same age?
  - Same gender?
  - Same height?
  - Same weight?
  - ...

- We need some distance function: $\rho : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$. 
The nearest neighbor algorithm

- $S = ((x_1, y_1), \ldots, (x_m, y_m))$
- For $x \in \mathcal{X}$, denote by $\text{nn}(x)$ the index of the most similar example to $x$ in $S$. So:

$$\rho(x_{\text{nn}(x)}, x) = \min_{i \leq m} \rho(x_i, x).$$

---

**Nearest Neighbor algorithm**

**input** A training sample $S$

**output** A function $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$.

1. Set $\hat{h}_S = f_{\text{nn}}^S$, where $f_{\text{nn}}^S$ is defined as:

$$\forall x \in \mathcal{X}, \quad f_{\text{nn}}^S (x) = y_{\text{nn}}(x).$$
How to choose the distance function?

- Represent each example in $\mathcal{X}$ by a vector in $\mathbb{R}^d$, where $d$ is the number of features for each example.
- **feature**: a single property of the example.
- Map all possible feature values to real numbers.
- For instance, possible features for customers:
  - age (already a number)
  - gender (represent one of the genders as 1, the other as 0)
  - eye color: can be mapped to several binary features:
    - Is eye color blue? (0/1)
    - Is eye color brown? (0/1)
    - Is eye color green? (0/1)
  - E.g. a 25 years-old female customer with blue eyes: (25, 1, 1, 0, 0)
- Define the distance to be the Euclidean distance in $\mathbb{R}^d$:
  $$\rho(x, x') = \|x - x'\| \equiv \sqrt{\sum_{i=1}^{d} (x(i) - x'(i))^2}$$

Kontorovich and Sabato (BGU) Lecture 2
Examples of feature mappings

- Task: Identify digits in scanned images.
  - The examples are grayscale photos of size $128 \times 128$ pixels.
  - The labels are the digits $0, \ldots, 9$.
  - Can represent examples in $\mathbb{R}^d$, with $d = 128^2$.
  - A feature for every pixel. Value of feature is intensity of pixel.

- Task: Identify whether a document is a scientific article.
  - The examples are text documents of varying length.
  - The labels are 0 or 1: 1 if scientific article.
  - Can represent examples in $\mathbb{R}^d$, where $d =$ number of words in the English language.
  - Use a lookup table to map each word to one of $1, \ldots, d$.
  - $x(i) = 1$ iff word $i$ appears in the document $x$.

Such feature mappings are used by many learning algorithms.

So we will usually assume in this course that $\mathcal{X} \subseteq \mathbb{R}^d$. 

-
Nearest Neighbor illustration

- Each example in the training sample creates a Voronoi cell.
- All the new examples that fall in this cell will get the same label.
- For $\mathcal{X} = \mathbb{R}^2$, we can draw the sample and the resulting Voronoi cells.
Nearest Neighbor and the distance function

- Will **Nearest Neighbor** work well?
- It depends on the distance function.
- If the distance function is the Euclidean distance, it depends on the representation of $\mathcal{X}$: the feature mapping.
- Good case: many of the features are relevant to the classification task.
- Bad case: many of the features are irrelevant.
  - Suppose only the customer’s age determines the favorite drink.
  - Two customers of same age might still be “far” if there are many other features.
- Same distance function can be good for one task, but bad for another!
- Can we get close to the **best possible prediction rule**?
The Bayes-optimal rule

- What is the best prediction rule?
- Assume \( \mathcal{Y} = \{0, 1\} \). (E.g. Tea and Coffee.)
- Define

\[
\eta(x) := \mathbb{P}_{(X,Y) \sim D}[Y = 1 | X = x] \equiv D((x, 1))/D(x).
\]

(E.g. the probability that Ross selects Coffee.)
- The best prediction rule is the **Bayes-optimal rule**:

\[
h^*(x) := \mathbb{I}[\eta(x) > 1/2].
\]

- For a general set of labels \( \mathcal{Y} \):

\[
h^*(x) := \arg\max_{y \in \mathcal{Y}} \mathbb{P}_{(X,Y) \sim D}[Y = y | X = x].
\]
The Bayes-optimal rule

- Proof for the Bayes-optimal rule:

\[
\text{err}(h, D) = \mathbb{P}_{(X, Y) \sim D}[h(X) \neq Y]
\]

\[
= \sum_{(x, y) \in \mathcal{X} \times \mathcal{Y}: h(x) \neq y} \mathbb{P}[X = x, Y = y]
\]

\[
= \sum_{(x, y) \in \mathcal{X} \times \mathcal{Y}: h(x) \neq y} \mathbb{P}[X = x] \cdot \mathbb{P}[Y = y | X = x]
\]

\[
= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}: h(x) \neq y} \mathbb{P}[X = x] \cdot \mathbb{P}[Y = y | X = x]
\]

\[
= \sum_{x \in \mathcal{X}} \mathbb{P}[X = x] \cdot \sum_{y \in \mathcal{Y}: h(x) \neq y} \mathbb{P}[Y = y | X = x]
\]

\[
= \sum_{x \in \mathcal{X}} \mathbb{P}[X = x] \cdot \mathbb{P}[Y \neq h(x) | X = x]
\]

\[
= \sum_{x \in \mathcal{X}} \mathbb{P}[X = x] (1 - \mathbb{P}[Y = h(x) | X = x])
\]

- err\((h, D)\) is smallest when for each \(x\), \(\mathbb{P}[Y = h(x) | X = x]\) is largest.
- Hence \(\text{argmax}_{y \in \mathcal{Y}} \mathbb{P}_{(X, Y) \sim D}[Y = y | X = x]\) is the optimal choice for \(h(x)\).
When does Nearest Neighbor approach the optimal error?

- When does NN approach the Bayes optimal as $m \to \infty$?
- Simple case: suppose each $x$ has a **deterministic label**
  - $\eta(x)$ is either 0 or 1 for each $x$
  - What is the smallest distance between $x$’s with a different label?
  - What is the Bayes-optimal error?
  - If the distribution is **discrete**:
    - If the sample includes all points in the **support of $D$**
    - Then NN gets zero error on the distribution.
- But can we get there without seeing all points in the sample?
- What about a non-discrete distribution?
- What if the label is not deterministic?
- What property of the distribution allows NN to do well?
Approaching the Bayes-optimal error

- NN will do well if examples that are close together usually have the same label.
- A “nice” distribution is one in which there is some $c > 0$ such that

$$\forall x, x' \in \mathcal{X}, \quad |\eta(x) - \eta(x')| \leq c\|x - x'\|$$

(recall $\eta(x) := \mathbb{P}_{(X,Y) \sim \mathcal{D}}[Y = 1 \mid X = x]$)

- We say in this case that $\eta$ is $c$-Lipschitz.
- Suppose that in $\mathcal{D}$:
  - every $x$ has a deterministic label, and
  - $\eta$ of $\mathcal{D}$ is $c$-Lischitz.

- In this case, what can we say about the support of $\mathcal{D}$?
- What does it mean for NN?
Approaching the Bayes-optimal error

- Simple case: **deterministic labels for each** $x$
  - What sample is needed so that NN finds the optimal prediction rule?
  - It suffices to have a sample point at each ball of radius $1/2c$.
  - Worst case: a grid with alternating labels.
  - Example: distribution in $\mathbb{R}^2$ (black: label 0, white: label 1)
    
    $\begin{array}{ccc}
    \bullet & \circ & \circ \\
    \circ & \bullet & \circ \\
    \bullet & \circ & \circ \\
    \circ & \bullet & \circ \\
    \end{array}$
  
  - In worst case, requires sample size exponential in $d$.

- General case: any value of $\eta(x)$ is possible.
  - Will NN find the optimal prediction rule?
  - Not necessarily! Even with $m \to \infty$.
  - The sample may have some points with the minority label.
  - If the distribution is *continuous*, there are no repeated example.
  - The NN will make a mistake around the points with a minority label in the sample.
Approaching the Bayes-optimal error

**Theorem**

If $\mathcal{X} \subseteq [0, 1]^d$, $\mathcal{Y} = \{0, 1\}$, and $\eta$ for the distribution $\mathcal{D}$ is $c$-Lipschitz, then

$$
\mathbb{E}_{S \sim \mathcal{D}^m}[\text{err}(f_{S}^{\text{nn}}, \mathcal{D})] \leq 2\text{err}(h^*, \mathcal{D}) + 4c\sqrt{d} m^{-1/(d+1)}.
$$

- The nearest neighbor rule approaches twice the error of the Bayes-optimal.
- In some cases, the sample size $m$ needs to be exponential in $d$.
- If $d$ is large, this could be a serious problem.
- This is termed **The curse of dimensionality**:
  - More dimensions (features) $\Rightarrow$ many more examples are needed.
**k-Nearest Neighbor**

- We showed that we can approach $2 \times$ The Bayes-optimal error.
- Can we approach the Bayes-optimal error exactly?
- $S = ((x_1, y_1), \ldots, (x_m, y_m))$
- For $x \in \mathcal{X}$, denote by $\pi_i(x)$ the index of the $i$'th closest example to $x$ in $S$. So:
  \[
  i \leq j \implies \rho(x_{\pi_i(x)}, x) \leq \rho(x_{\pi_j(x)}, x).
  \]

---

### k-Nearest Neighbors algorithm

**input** A training sample $S$, integer $k \geq 1$.

**output** A function $\hat{h}_S : \mathcal{X} \rightarrow \mathcal{Y}$.

1. Set $\hat{h}_S = f^k_{S-nn}$, where $f^k_{S-nn}$ is defined as:

   $$\forall x \in \mathcal{X}, \quad f^k_{S-nn}(x) = \text{The majority label among } y_{\pi_1(x)}, \ldots, y_{\pi_k(x)}.$$
Theorem

Suppose that $k_1, k_2, \ldots$ is a sequence such that $\lim_{m \to \infty} k_m = \infty$, and $\lim_{m \to \infty} k_m/m = 0$. Then

$$\lim_{m \to \infty} \mathbb{E}_{S \sim D^m}[\text{err}(f_{S}^{k_m-\text{nn}}, D)] = \text{err}(h^*, D).$$

- By increasing $k$ slowly with the sample size, the correct majority label can be approached in every point in the space of $\mathcal{X}$.
- **Curse of dimensionality** still holds for $k$-nearest neighbors.
- In practice, $k$-nearest neighbors is often used with some small fixed $k$. 
Efficient implementation

- Computing distances in high dimensions is expensive.
- Also: storing the entire sample is memory-intensive.
- How to speed up calculations and save memory?
  - Idea: project onto lower-dimensional subspace
    - **Principal Components Analysis (PCA)** preserves the general “cloud” shape of the data.
    - **Johnson-Lindenstrauss transform (JL)** approximately preserves pairwise distances.
  - Idea: sample compression/condensing
    - store only a few “representative” examples and discard the rest
    - memory efficient
    - also results in prediction speedup
    - turns out that **also** can improve generalization!
  - We will see some of these methods later in the course.