### Lecture 2: Supervised Learning

- k-Nearest Neighbor
- Linear regression
- Optimization perspective of regression

#### Data: (row = 1 data point)

<table>
<thead>
<tr>
<th>Person number</th>
<th>HW1</th>
<th>Final Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>83</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>94</td>
<td>87</td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>82</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>77</td>
<td>89</td>
</tr>
</tbody>
</table>

- number of points: \( n = 80 \)

**Input:** 80

**Output:** 80
Idea: Find the point with closest HW1 score (break ties randomly), predict same final score.

Pros
- Simple
- Sometimes all you need!
- Efficient
  - $O(\log(n))$ average
  - $O(n)$ worst case.
- KD-Tree.

Cons
- Not always accurate, even with tons of data.

Improvements
1) Average values for all points tied for "closest"
   - What if inputs are continuous?
2) Average the values for the k nearest points.
   - k is called a hyperparameter. - Called k-nearest neighbor k-NN.
Hyperparameter: a parameter whose value is used to control the learning process.

3) Assign a weight to the nearest neighbors based on their distance, and use a weighted average.
   - weighted k-NN.
Notation: Training data is an input-label pairs:
\[(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\]

Input label

Shorthand: \[(x_i, y_i)\] \(i = 1\) to \(n\)

Each input is a vector (array) of \(m\) features:
\[x_i = (x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,m})\]
\(x_{i,j} \in \mathbb{R}\) is real numbers.

\[x_i \in \mathbb{R}^m \times \mathbb{R} \times \ldots \times \mathbb{R}\]

indual measurable property or characteristic of a phenomenon being observed.
Each label is a real number.

\( \tilde{y}_i \in \mathbb{R} \), \( y_i \in \mathbb{R} \)

let \( \hat{y}_i \) be the agent's prediction of \( y_i \) from \( x_i \).

"\( \hat{\cdot} \)" → estimate or approximation of some "Regression problem"
Weighted k-NN

Input: \((x_i, y_i)_{i=1}^n, x_{n+1}, k \in \mathbb{N}_0, \sigma \in \mathbb{R}_0, \) distance measure \(\text{dist}\)

Output: \(\hat{y}_{n+1}\)

1) for \(i = 1\) to \(n\)
    \[d_i = \text{dist}(x_i, x_{n+1})\]
    \[w_i = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{d_i^2}{\sigma^2}\right)\]

2) Sort \((x_i, y_i, d_i, w_i)_{i=1}^n\) by \(d_i\), smallest to largest.

3) \(k' = \min(n, k)\)

4) \(c = \sum_{i=1}^{k'} w_i\)

5) \(\hat{y}_{n+1} = \frac{\sum_{i=1}^{k'} w_i y_i}{c}\)

6) Return \(\hat{y}_{n+1}\).

\(\text{Non-weighted:}\)

\[w_i = \begin{cases} 1 - \delta_i & \text{if } d_i \leq 1 \\ 0 & \text{otherwise.} \end{cases}\]
How pick hyper-params
- Art!

k, i, dist?

\[ \downarrow \]

how compute weights from distances?
let \( f \) be the function mapping inputs to label predictions.
\[
\hat{y}_i = f(x_i)
\]

We call \( f \) a model in **supervised learning**.

- 1-NN
- Linear model: \( \hat{y} = mx + b \)
- Polynomial model: \( \hat{y} = ax^2 + bx + c \)

\( k \)-NN is nonparametric. - it doesn't assume a particular parametric form for the model.

**Parametric model forms:**

- \( \hat{y} = mx + b \)

  parameters: \( m, b \)

  is often called "weights", \( w = (w_1, w_2, ..., w_{1*w}) \)

\( |w| \)

  absolute value

\( |W| \)

  cardinality of the set

\( ||w|| \)

  length of vector
Linear Regression

- Assume linear parametric form of the model.

\[ f_w(x_i) = w_1 x_{i1} + w_2 x_{i2} + \ldots + w_m x_{im} \]

\[ f(x_i, w) = \sum_{j=1}^{m} w_j x_{ij} \]

= \text{(dot product)}

= \mathbf{w}^T \mathbf{x}_i