18-661 Introduction to Machine Learning

Nearest Neighbors

Spring 2025

ECE – Carnegie Mellon University

Announcements

- Homework 2 is due on Friday February 21. Homework 3 will be released on Friday as well.

Announcements

- Homework 2 is due on Friday February 21. Homework 3 will be released on Friday as well.
- Midterm exam is on Wednesday, February 26.
    - The end of this lecture has some slides on mini-exam 1 and review for the midterm.
    - Friday's recitation will go over practice problems and exam review.
    - If you have an excused absence, email us with the reason by Friday February 21. If something comes up after that (e.g., illness, family emergency), email us as soon as possible.

Outline

Recap: SVMs

Summary: Three SVM Formulations

Hard-margin (for separable data)

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1, \ \forall \ n$$

Soft-margin (add slack variables)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \ \text{ s.t. } y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \ \xi_n \geq 0, \ \forall \ n$$

Hinge loss (define a loss function for each data point)

$$\min_{\boldsymbol{w},b} \ \sum_n \max(0, 1 - y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Summary of Dual Formulation

Primal Max-Margin Formulation

$$\begin{aligned} \min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \boldsymbol{x}_n + b] \geq 1 - \xi_n, \quad \forall \ \ n \\ & \xi_n \geq 0, \quad \forall \ n \end{aligned}$$

Dual Formulation

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{x}_m^\top \boldsymbol{x}_n \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall \ n \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

- In dual formulation, the # of variables is independent of dimension.
- Only the support vectors have nonzero dual variables.
- Can easily recover the primal solution $\boldsymbol{w}, b$ from dual solution.

Primal and Dual SVM Formulations: Kernel Versions

Primal

$$\begin{aligned}\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n \\ \text{s.t.} \quad & y_n[\boldsymbol{w}^\top \phi(\boldsymbol{x}_n) + b] \geq 1 - \xi_n, \quad \forall \; n \\ & \xi_n \geq 0, \quad \forall \; n\end{aligned}$$

Dual

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n \phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$$

$$\begin{aligned}\text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall \; n \\ & \sum_n \alpha_n y_n = 0\end{aligned}$$

IMPORTANT POINT: In the dual problem, we only need $\phi(\boldsymbol{x}_m)^\top \phi(\boldsymbol{x}_n)$.

The Kernel Trick

In dual SVM, we can use any of the kernel functions discussed in the previous lecture.

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\
\text{s.t.} \quad & 0 \le \alpha_n \le C, \quad \forall\ n \\
& \sum_n \alpha_n y_n = 0
\end{aligned}$$

The Kernel Trick

In dual SVM, we can use any of the kernel functions discussed in the previous lecture.

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0\end{aligned}$$

Each choice of kernel function will correspond to doing SVM using the transformed data $\phi(\boldsymbol{x})$, but we do not need to know what exactly is $\phi(\boldsymbol{x})$.

The Kernel Trick

In dual SVM, we can use any of the kernel functions discussed in the previous lecture.

$$\begin{aligned}\max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad \forall\ n \\ & \sum_n \alpha_n y_n = 0\end{aligned}$$

Each choice of kernel function will correspond to doing SVM using the transformed data $\phi(\boldsymbol{x})$, but we do not need to know what exactly is $\phi(\boldsymbol{x})$.

This is allows us using more complicated $\phi(\boldsymbol{x})$ (like the $\phi(\boldsymbol{x})$ associated with radial basis function) to boost performance - without knowing what $\phi(\boldsymbol{x})$ is! This is known as the "kernel trick".

Test Prediction

Learning $\boldsymbol{w}$ and b:

$$\boldsymbol{w} = \sum_n \alpha_n y_n \phi(\boldsymbol{x}_n),$$

$$b = y_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) = y_n - \sum_m \alpha_m y_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

But for test prediction on a new point $\mathbf{x}$, do we need the form of $\phi(\mathbf{x})$ in order to find the sign of $\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$?

Test Prediction

Learning $\boldsymbol{w}$ and b:

$$\boldsymbol{w} = \sum_n \alpha_n y_n \phi(\boldsymbol{x}_n),$$

$$b = y_n - \boldsymbol{w}^\top \phi(\boldsymbol{x}_n) = y_n - \sum_m \alpha_m y_m k(\boldsymbol{x}_m, \boldsymbol{x}_n)$$

But for test prediction on a new point $\mathbf{x}$, do we need the form of $\phi(\mathbf{x})$ in order to find the sign of $\boldsymbol{w}^\top \phi(\boldsymbol{x}) + b$? Fortunately, no!

Test Prediction:

$$h(\boldsymbol{x}) = \textsc{sign}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

At test time it suffices to know the kernel function! So we really do not need to know ϕ.

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n) \text{ for } n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Training

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_m)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Summary of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Select a kernel. In general, you can just use one of the popular kernel functions (polynomial kernel or radial kernel).

Training

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n k(\mathbf{x}_m, \mathbf{x}_m)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

$$\sum_n \alpha_n y_n = 0$$

Prediction

$$h(\boldsymbol{x}) = \text{SIGN}(\sum_n y_n \alpha_n k(\boldsymbol{x}_n, \boldsymbol{x}) + b)$$

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

What if the data is not linearly separable?

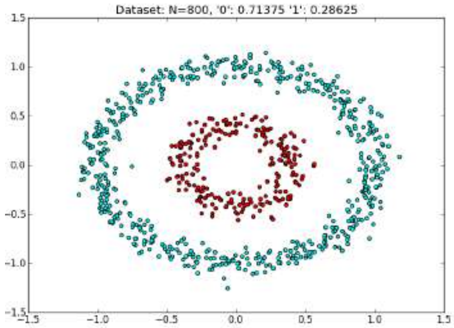
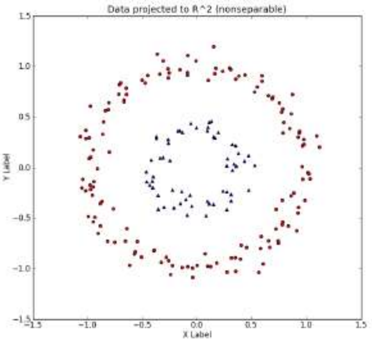


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Use feature $\phi(x) = [x_1, x_2, x_1^2 + x_2^2]$ to transform the data in a 3D space



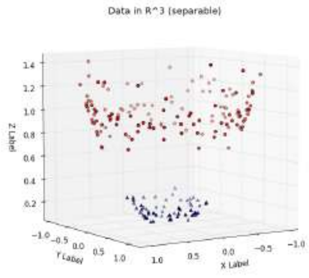


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Then find the decision boundary. How? Solve the dual problem!

$$\max_{\boldsymbol{\alpha}} \sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^\top \boldsymbol{\phi}(\boldsymbol{x}_n)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad \forall\ n$$

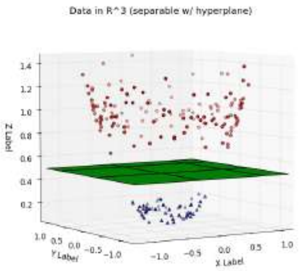
$$\sum_n \alpha_n y_n = 0$$

Then find $\mathbf{w}$ and b. Predict $y = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}) + b)$.

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Here is the resulting decision boundary



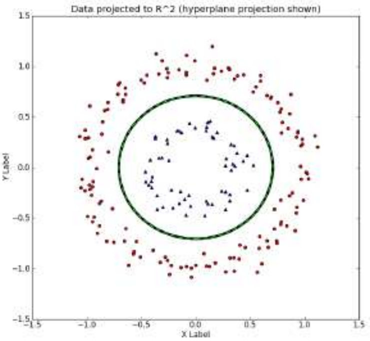


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Example of Kernel SVM

Given a dataset $\{(\boldsymbol{x}_n, y_n)$ for $n = 1, 2, \ldots, N\}$, how do you classify it using kernel SVM ?

Effect of the choice of kernel: Radial Basis Kernel

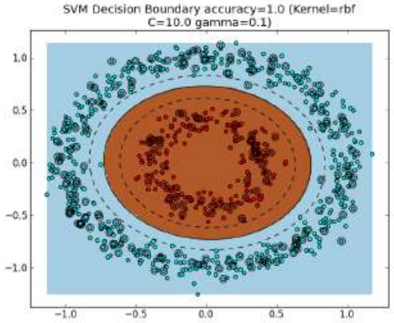


Image Source: `https://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

Advantages of SVM

Now we have shown all of the below.

1. Maximizes distance of training data from the boundary
2. Only requires a subset of the training points.
3. Is less sensitive to outliers.
4. Scales better with high-dimensional data.
5. Generalizes well to many nonlinear models.

Outline

1. Recap: SVMs
2. Parametric and Nonparametric Models
3. Nearest Neighbor Classification
4. K Nearest Neighbors Classification
5. Practical Aspects of NN
6. Midterm Review

Parametric and Nonparametric Models

Parametric vs. Nonparametric

- So far, we've discussed parametric machine learning models:
    - Linear regression
    - Naïve Bayes
    - Logistic regression
    - Linear SVMs

Parametric vs. Nonparametric

- So far, we've discussed parametric machine learning models:
  - Linear regression
  - Naïve Bayes
  - Logistic regression
  - Linear SVMs
- Next, we will discuss two *nonparametric* models:
  - Nearest neighbors
  - Decision trees

Parametric vs. Nonparametric

Key difference:

- Parametric models can be characterized via some *fixed* set of parameters θ. Given this set of parameters, our future predictions are independent of the data $\mathcal{D}$, i.e., $P(x|\theta, \mathcal{D}) = P(x|\theta)$.

Parametric vs. Nonparametric

Key difference:

- Parametric models can be characterized via some *fixed* set of parameters θ. Given this set of parameters, our future predictions are independent of the data $\mathcal{D}$, i.e., $P(x|\theta, \mathcal{D}) = P(x|\theta)$.
    - E.g. logistic regression, SVM make a prediction based on the decision boundary $\boldsymbol{w}^\top \boldsymbol{x} + b$, where $(\boldsymbol{w}, b)$ can be viewed as the parameter θ.
    - Often simpler and faster to learn, but can sometimes be a poor fit

Parametric vs. Nonparametric

Key difference:

- Parametric models can be characterized via some *fixed* set of parameters θ. Given this set of parameters, our future predictions are independent of the data $\mathcal{D}$, i.e., $P(x|\theta, \mathcal{D}) = P(x|\theta)$.
  - E.g. logistic regression, SVM make a prediction based on the decision boundary $\boldsymbol{w}^\top \boldsymbol{x} + b$, where $(\boldsymbol{w}, b)$ can be viewed as the parameter θ.
  - Often simpler and faster to learn, but can sometimes be a poor fit
- Nonparametric models make a prediction directly based on the data $\mathcal{D}$ (without explicitly learning a fixed parameter set θ).
  - More complex and computationally expensive, but can learn more flexible patterns

Parametric vs. Nonparametric

Key difference:

- Parametric models can be characterized via some *fixed* set of parameters θ. Given this set of parameters, our future predictions are independent of the data $\mathcal{D}$, i.e., $P(x|\theta, \mathcal{D}) = P(x|\theta)$.
  - E.g. logistic regression, SVM make a prediction based on the decision boundary $\boldsymbol{w}^\top \boldsymbol{x} + b$, where $(\boldsymbol{w}, b)$ can be viewed as the parameter θ.
  - Often simpler and faster to learn, but can sometimes be a poor fit
- Nonparametric models make a prediction directly based on the data $\mathcal{D}$ (without explicitly learning a fixed parameter set θ).
  - More complex and computationally expensive, but can learn more flexible patterns
- Both parametric and nonparametric methods can be used for either regression or classification. Both require training data with input features and labels (i.e., supervised learning).

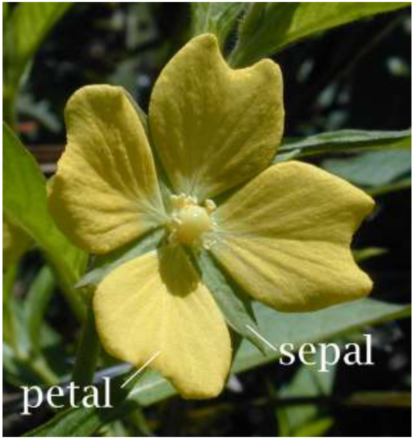
Nearest Neighbor Classification

Recognizing Flowers

Types of Iris: setosa, versicolor, and virginica

Measuring Flower Properties

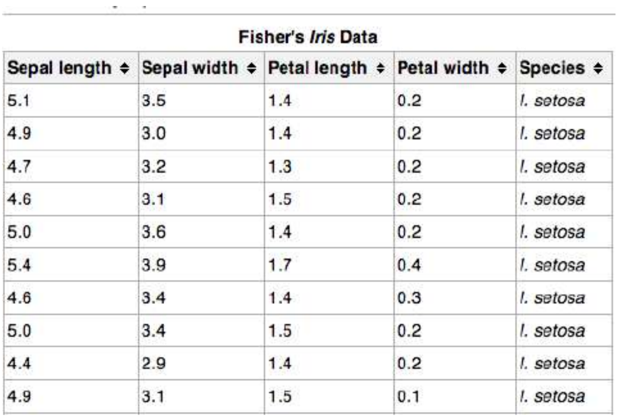
Features: the widths and lengths of sepal and petal



Data Often Fits into a Table

Ex: Iris data (click here for all data)

- 4 features
- 3 classes



Fisher's *Iris* Data

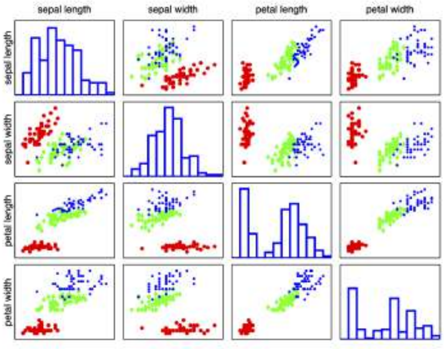
Sepal length ⬍	Sepal width ⬍	Petal length ⬍	Petal width ⬍	Species ⬍
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I. setosa*
4.9	3.1	1.5	0.1	*I. setosa*

Pairwise Scatter Plots of 131 Flower Specimens

Visualization of data helps to identify the right learning model

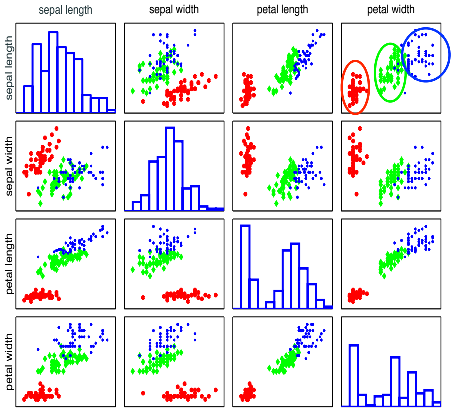
Which combination of features separates the three classes?

Figure 1: Each colored point is a flower specimen: setosa, versicolor, virginica

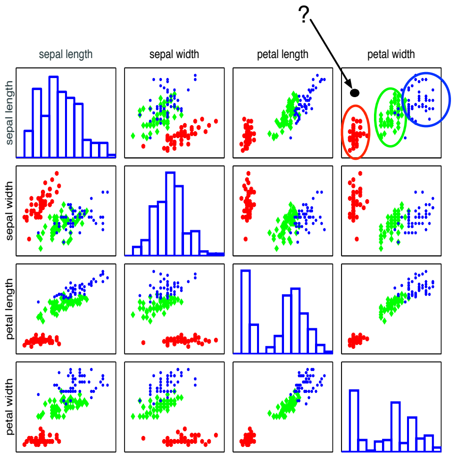


Different Types Seem Well-clustered and Separable

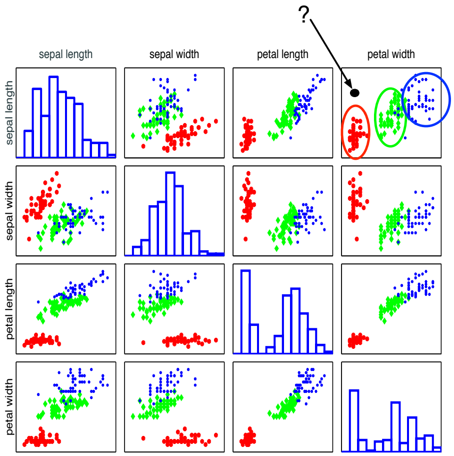
Using two features: petal width and sepal length



Labeling an Unknown Flower Type



Labeling an Unknown Flower Type



Closer to red cluster: so labeling it as setosa

Nearest Neighbor Classification (NNC)

Training data (set)

- N samples: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_N, y_N)\}$
- Input $\boldsymbol{x}_n \in \mathbb{R}^D$, output (label): $y_n \in [C] = \{1, 2, \cdots, C\}$
- Learning goal: given a test point $\boldsymbol{x}$, predict its label y.

Nearest Neighbor Classification (NNC)

Training data (set)

- N samples: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathsf{N}, y_\mathsf{N})\}$
- Input $\boldsymbol{x}_n \in \mathbb{R}^\mathsf{D}$, output (label): $y_n \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$
- Learning goal: given a test point $\boldsymbol{x}$, predict its label y.

Nearest neighbor of a test data point $\boldsymbol{x}$

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\text{nn}(\boldsymbol{x})}$$

where $\text{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$ is index of the closest training sample

Nearest Neighbor Classification (NNC)

Training data (set)

- N samples: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\mathrm{N}}, y_{\mathrm{N}})\}$
- Input $\boldsymbol{x}_n \in \mathbb{R}^{\mathrm{D}}$, output (label): $y_n \in [\mathrm{C}] = \{1, 2, \cdots, \mathrm{C}\}$
- Learning goal: given a test point $\boldsymbol{x}$, predict its label y.

Nearest neighbor of a test data point $\boldsymbol{x}$

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$ is index of the closest training sample

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \mathrm{argmin}_{n \in [\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2$$

Nearest Neighbor Classification (NNC)

Training data (set)

- N samples: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathrm{N}, y_\mathrm{N})\}$
- Input $\boldsymbol{x}_n \in \mathbb{R}^\mathrm{D}$, output (label): $y_n \in [\mathrm{C}] = \{1, 2, \cdots, \mathrm{C}\}$
- Learning goal: given a test point $\boldsymbol{x}$, predict its label y.

Nearest neighbor of a test data point $\boldsymbol{x}$

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$ is index of the closest training sample

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n\in[\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \mathrm{argmin}_{n\in[\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2$$

Classification rule

$$y = f(\boldsymbol{x}) = y_{\mathrm{nn}(\boldsymbol{x})}$$

Nearest Neighbor Classification (NNC)

Training data (set)

- N samples: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_\mathrm{N}, y_\mathrm{N})\}$
- Input $\boldsymbol{x}_n \in \mathbb{R}^\mathrm{D}$, output (label): $y_n \in [\mathrm{C}] = \{1, 2, \cdots, \mathrm{C}\}$
- Learning goal: given a test point $\boldsymbol{x}$, predict its label y.

Nearest neighbor of a test data point x

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$ is index of the closest training sample

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \mathrm{argmin}_{n \in [\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2$$

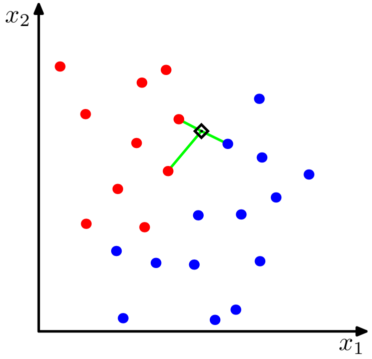
Classification rule

$$y = f(\boldsymbol{x}) = y_{\mathrm{nn}(\boldsymbol{x})}$$

Example: if $\mathrm{nn}(\boldsymbol{x}) = 2$, then $y_{\mathrm{nn}(\boldsymbol{x})} = y_2$, which is the label of the 2nd data point.

Visual Example

In this 2-dimensional example, the nearest point to $\boldsymbol{x}$ is a red training instance, thus, $\boldsymbol{x}$ will be labeled as red.



(a)

How to Measure "Nearness"?

Previously, we used Euclidean distance

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n\in[\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$$

We can also use alternative distances

- E.g., the ℓ_1 distance (i.e., city block distance, or Manhattan distance):

$$\begin{aligned}\mathrm{nn}(\boldsymbol{x}) &= \mathrm{argmin}_{n\in[\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_1 \\ &= \mathrm{argmin}_{n\in[\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} |x_d - x_{nd}|\end{aligned}$$

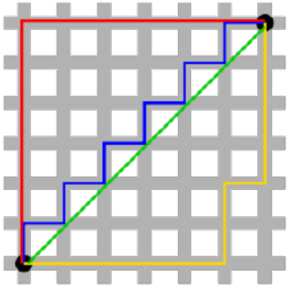


Figure 2: Green line is Euclidean distance. Red, Blue, and Yellow lines are L_1 distance

How to Measure "Nearness"?

Previously, we used Euclidean distance

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n\in[\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$$

We can also use alternative distances

- E.g., the ℓ_1 distance (i.e., city block distance, or Manhattan distance):

$$\begin{aligned}\mathrm{nn}(\boldsymbol{x}) &= \mathrm{argmin}_{n\in[\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_1 \\ &= \mathrm{argmin}_{n\in[\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} |x_d - x_{nd}|\end{aligned}$$

- Or, the ℓ_∞ (supremum) distance:

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n\in[\mathrm{N}]} \max_d |x_d - x_{nd}|$$

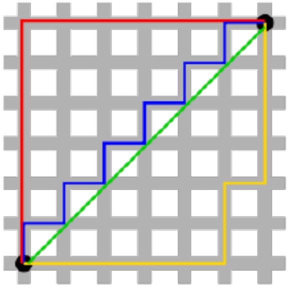


Figure 2: Green line is Euclidean distance. Red, Blue, and Yellow lines are L_1 distance

Nearest Neighbors for Regression

Recall the nearest neighbor of a (training or test) data point:

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$ indexes a training instance

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \mathrm{argmin}_{n \in [\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2$$

Nearest Neighbors for Regression

Recall the nearest neighbor of a (training or test) data point:

$$\boldsymbol{x}(1) = \boldsymbol{x}_{\mathrm{nn}(\boldsymbol{x})}$$

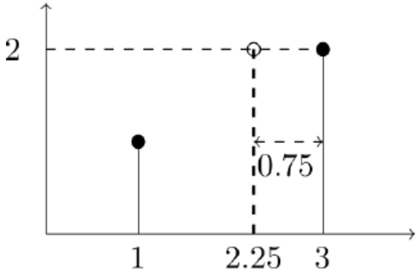
where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$ indexes a training instance

$$\mathrm{nn}(\boldsymbol{x}) = \mathrm{argmin}_{n\in[\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \mathrm{argmin}_{n\in[\mathrm{N}]} \sum_{d=1}^{\mathrm{D}} (x_d - x_{nd})^2$$

Regression rule

$$y = f(\boldsymbol{x}) = y_{\mathrm{nn}(\boldsymbol{x})}$$

Label **x** with the label of its nearest neighbor!



Parametric vs. Nonparametric, Revisited

Nonparametric models make predictions directly based on the data $\mathcal{D}$ (without learning any parametric model like the linear decision boundary in SVM/logistic regression).

- Parametric models are often simpler and faster to learn, but can sometimes be a poor fit.
- Nonparametric models are more complex and computationally expensive, but can learn more flexible patterns.

Parametric vs. Nonparametric, Revisited

Nonparametric models make predictions directly based on the data $\mathcal{D}$ (without learning any parametric model like the linear decision boundary in SVM/logistic regression).

- Parametric models are often simpler and faster to learn, but can sometimes be a poor fit.
- Nonparametric models are more complex and computationally expensive, but can learn more flexible patterns.

How does this manifest for nearest neighbors?

- Nearest neighbors often learns a *highly nonlinear* decision boundary.

Parametric vs. Nonparametric, Revisited

Nonparametric models make predictions directly based on the data $\mathcal{D}$ (without learning any parametric model like the linear decision boundary in SVM/logistic regression).

- Parametric models are often simpler and faster to learn, but can sometimes be a poor fit.
- Nonparametric models are more complex and computationally expensive, but can learn more flexible patterns.

How does this manifest for nearest neighbors?

- Nearest neighbors often learns a *highly nonlinear* decision boundary.
- But, we need to compare the test data point to *every sample in the training dataset*, which is *computationally expensive*.

Outline

1. Recap: SVMs
2. Parametric and Nonparametric Models
3. Nearest Neighbor Classification
4. K Nearest Neighbors Classification
5. Practical Aspects of NN
6. Midterm Review

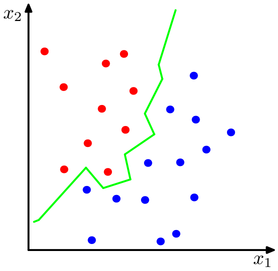
K Nearest Neighbors Classification

Drawbacks of Using One Nearest Neighbor

- What if the nearest neighbor of the test point is an outlier?
- Relying on one nearest neighbor makes the decision boundary susceptible to outliers

Drawbacks of Using One Nearest Neighbor

- What if the nearest neighbor of the test point is an outlier?
- Relying on one nearest neighbor makes the decision boundary susceptible to outliers
- Even without outliers, it results in a complex, jagged decision boundary



(b)

Solution: Use K nearest neighbors and combine their labels

K-Nearest Neighbor (KNN) Classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\text{nn}_1(\boldsymbol{x}) = \text{argmin}_{n\in[\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$

K-Nearest Neighbor (KNN) Classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\mathrm{nn}_1(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 2nd-nearest neighbor: $\mathrm{nn}_2(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$

K-Nearest Neighbor (KNN) Classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\text{nn}_1(\boldsymbol{x}) = \text{argmin}_{n\in[\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 2nd-nearest neighbor: $\text{nn}_2(\boldsymbol{x}) = \text{argmin}_{n\in[\text{N}]-\text{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 3rd-nearest neighbor: $\text{nn}_3(\boldsymbol{x}) = \text{argmin}_{n\in[\text{N}]-\text{nn}_1(\boldsymbol{x})-\text{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$

K-Nearest Neighbor (KNN) Classification

Increase the number of nearest neighbors to use?

- 1-nearest neighbor: $\mathrm{nn}_1(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 2nd-nearest neighbor: $\mathrm{nn}_2(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$
- 3rd-nearest neighbor: $\mathrm{nn}_3(\boldsymbol{x}) = \mathrm{argmin}_{n \in [\mathrm{N}] - \mathrm{nn}_1(\boldsymbol{x}) - \mathrm{nn}_2(\boldsymbol{x})} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$

The set of K-nearest neighbors

$$\mathrm{knn}(\boldsymbol{x}) = \{\mathrm{nn}_1(\boldsymbol{x}), \mathrm{nn}_2(\boldsymbol{x}), \cdots, \mathrm{nn}_K(\boldsymbol{x})\}$$

Let $\boldsymbol{x}(i) = \boldsymbol{x}_{\mathrm{nn}_i(\boldsymbol{x})}$, then

$$\|\boldsymbol{x} - \boldsymbol{x}(1)\|_2^2 \leq \|\boldsymbol{x} - \boldsymbol{x}(2)\|_2^2 \cdots \leq \|\boldsymbol{x} - \boldsymbol{x}(K)\|_2^2$$

How to Classify with K Neighbors?

Classification rule

- Every neighbor votes: suppose y_n (the true label) for $\boldsymbol{x}_n$ is c, then
  - vote for c is 1
  - vote for $c' \neq c$ is 0

How to Classify with K Neighbors?

Classification rule

- Every neighbor votes: suppose y_n (the true label) for $\boldsymbol{x}_n$ is c, then
  - vote for c is 1
  - vote for $c' \neq c$ is 0

  We use the *indicator function* $\mathbb{I}(y_n == c)$ to represent the votes.
- Aggregate everyone's vote

$$v_c = \sum_{n \in \mathrm{knn}(\boldsymbol{x})} \mathbb{I}(y_n == c), \quad \forall \quad c \in [\mathrm{C}]$$

How to Classify with K Neighbors?

Classification rule

- Every neighbor votes: suppose y_n (the true label) for $\boldsymbol{x}_n$ is c, then
    - vote for c is 1
    - vote for $c' \neq c$ is 0

  We use the *indicator function* $\mathbb{I}(y_n == c)$ to represent the votes.
- Aggregate everyone's vote

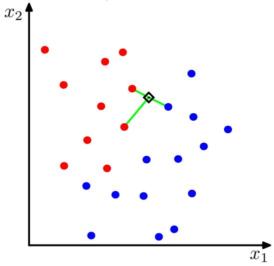
$$v_c = \sum_{n \in \text{knn}(\boldsymbol{x})} \mathbb{I}(y_n == c), \quad \forall \quad c \in [\text{C}]$$

- Label with the majority, breaking ties arbitrarily

$$y = f(\boldsymbol{x}) = \arg\max_{c \in [\text{C}]} v_c$$

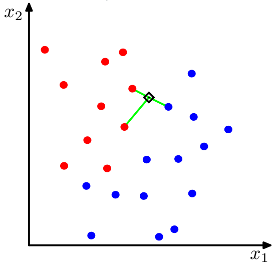
Example

K=1, Label: ??



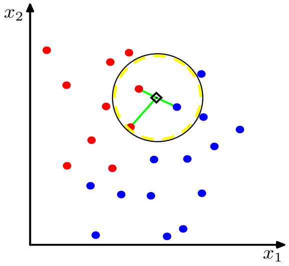
(a)

K=3, Label: ??



(a)

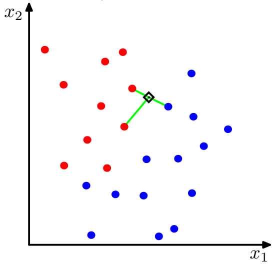
K=5, Label: ??



(a)

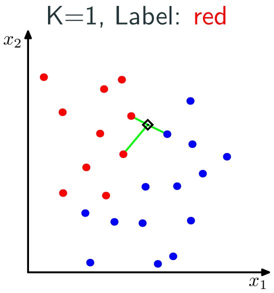
Example

K=1, Label: red

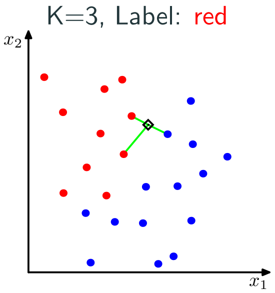


(a)

Example



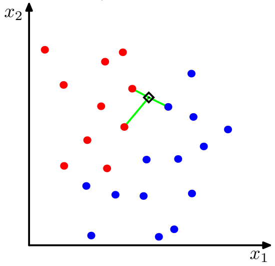
(a)



(a)

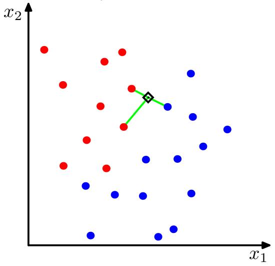
Example

K=1, Label: red



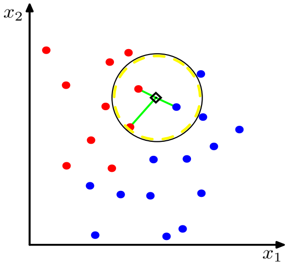
(a)

K=3, Label: red



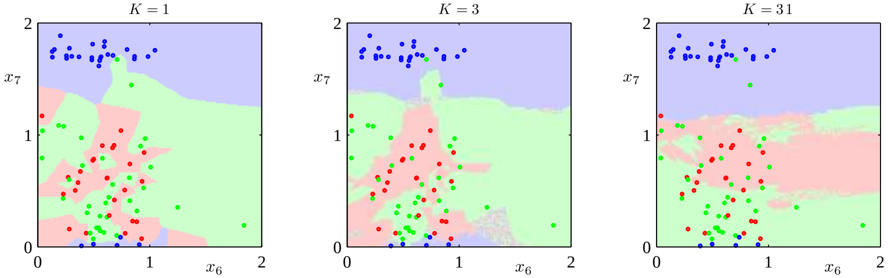
(a)

K=5, Label: blue



(a)

How to Choose an Optimal K?



- When K increases, the decision boundary becomes smoother and less susceptible to outliers.
- However, if K is too large, it can also lead to misclassification as we are taking votes from faraway training points.

How to Do Regression with K Neighbors?

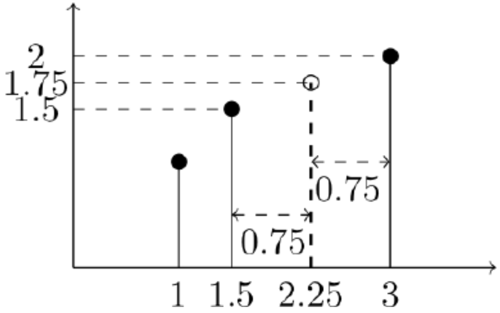
- We need a way to aggregate labels from each of the neighbors.
- Average the labels associated with the K points.

$$\hat{y} = \frac{1}{K} \sum_{n \in knn(\mathbf{x})} y_n$$

How to Do Regression with K Neighbors?

- We need a way to aggregate labels from each of the neighbors.
- Average the labels associated with the K points.

$$\hat{y} = \frac{1}{K} \sum_{n \in knn(\mathbf{x})} y_n$$



Pros and Cons of Nearest Neighbors

Advantages of NNC

- Simple and easy to implement – just compute distances, no optimization required
- Can learn complex decision boundaries

Pros and Cons of Nearest Neighbors

Advantages of NNC

- Simple and easy to implement – just compute distances, no optimization required
- Can learn complex decision boundaries

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(\mathrm{N}D)$ for labeling a data point.
- We need to “carry” the training data around. Without it, we cannot do classification.
- Choosing the right distance measure and K can be difficult.

Pros and Cons of Nearest Neighbors

Advantages of NNC

- Simple and easy to implement – just compute distances, no optimization required
- Can learn complex decision boundaries

Disadvantages of NNC

- Computationally intensive for large-scale problems: $O(\mathsf{N}D)$ for labeling a data point.
- We need to "carry" the training data around. Without it, we cannot do classification.
- Choosing the right distance measure and K can be difficult.
- Relies on the existence of training data points "close" to test points.
- Can break down if the classes are unbalanced.

Practical Aspects of NN

Hyperparameters in NN

Two crucial choices for NN

- Choosing K, i.e., the number of nearest neighbors (default is 1)

Hyperparameters in NN

Two crucial choices for NN

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance). Alternatively, can use L_1 distance, or more generally, the following L_p distance measure

$$\|\boldsymbol{x} - \boldsymbol{x}_n\|_p = \left(\sum_d |x_d - x_{nd}|^p \right)^{1/p}$$

for $p \geq 1$.

Hyperparameters in NN

Two crucial choices for NN

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance). Alternatively, can use L_1 distance, or more generally, the following L_p distance measure

$$\|\boldsymbol{x} - \boldsymbol{x}_n\|_p = \left(\sum_d |x_d - x_{nd}|^p \right)^{1/p}$$

  for $p \geq 1$.

These are not specified by the algorithm itself — use (cross-)validation!

Preprocessing Data

Normalize data to have zero mean and unit standard deviation in each dimension

- Compute the means and standard deviations in each feature

$$\bar{x}_d = \frac{1}{\mathrm{N}} \sum_n x_{nd}, \qquad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Preprocessing Data

Normalize data to have zero mean and unit standard deviation in each dimension

- Compute the means and standard deviations in each feature

$$\bar{x}_d = \frac{1}{\mathrm{N}} \sum_n x_{nd}, \qquad s_d^2 = \frac{1}{N-1} \sum_n (x_{nd} - \bar{x}_d)^2$$

- Scale the feature accordingly

$$x_{nd} \leftarrow \frac{x_{nd} - \bar{x}_d}{s_d}$$

Many other ways of normalizing data — you would need/want to try different ones and pick among them using (cross) validation

You Should Know

- The differences between parametric and non-parametric learning models
- How to implement K-nearest neighbors for regression and classification
- Practical aspects of NNC, such as tuning hyperparameters (K, distance metric) and feature scaling.

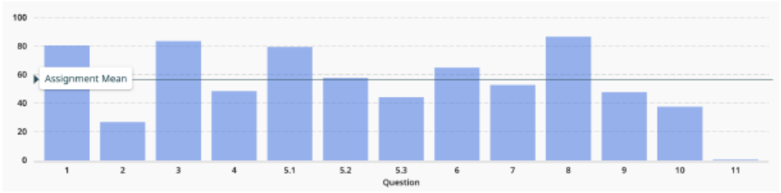
Midterm Review

Concepts You Should Know

This is a quick overview of important concepts/methods/models.

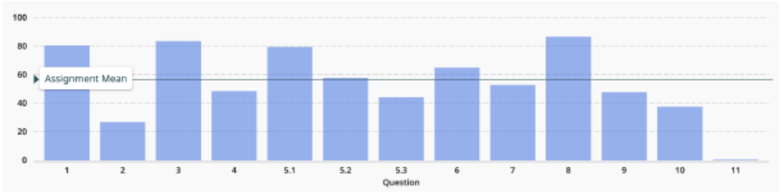
- MLE/MAP: how to find the likelihood of one or more observations, how to use a prior distribution, how to optimize the likelihood
- Linear regression: formulation, how it relates to MLE/MAP, feature scaling, ridge regression, gradient descent, nonlinear basis functions
- Bias-variance trade-off: bias and variance, overfitting, cross-validation
- Naive Bayes: naive classification rule, why it is naive, Laplacian smoothing
- Logistic regression: generative vs. discriminative models, formulation, how it relates to MLE, comparison to naive Bayes, sigmoid function, cross-entropy function, nonlinear boundaries
- Multi-class: one-vs-all and one-vs-one approaches, softmax function/multinomial logistic regression
- SVMs: hinge loss formulation, max-margin formulation, dual of the SVM problem, kernel functions
- Nearest neighbors: parametric vs. nonparametric models, k-nearest neighbors, tuning hyperparameters, feature scaling

Mini-Exam 1



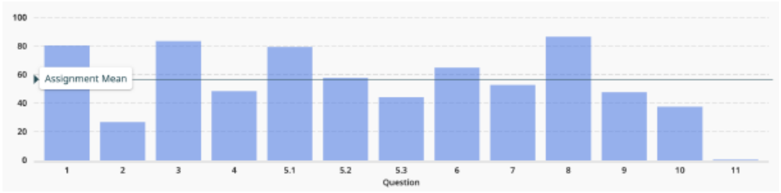
- Q2: MLE/MAP. *MAP does not always give a better parameter estimate than MLE.*

Mini-Exam 1



- Q2: MLE/MAP. *MAP does not always give a better parameter estimate than MLE.*
- Q4: Linear regression. *The linear regression solution is not, in general, a closed form solution to* $\mathbf{X}\mathbf{w} = \mathbf{y}$.

Mini-Exam 1



- Q2: MLE/MAP. *MAP does not always give a better parameter estimate than MLE.*
- Q4: Linear regression. *The linear regression solution is not, in general, a closed form solution to* $\mathbf{X}\mathbf{w} = \mathbf{y}$.
- Q9: Linear and ridge regression. *Regularization will generally increase training loss.*