18-661 Introduction to Machine Learning

Nearest Neighbors

Spring 2023

ECE – Carnegie Mellon University

Outline

1. Parametric and Nonparametric Models
2. Nearest Neighbor Classification
3. K Nearest Neighbors Classification
4. Practical Aspects of NN

Parametric and Nonparametric Models

Parametric vs. Nonparametric

- So far, we've discussed parametric machine learning models:
  - Linear regression
  - Naïve Bayes
  - Logistic regression
  - Linear SVMs
- This week, 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., $P(x|\theta, \mathcal{D}) = P(x|\theta)$.
  - E.g. logistic regression, SVM makes 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 prediction directly based on the data $\mathcal{D}$ (without explicitly learning a fixed parameter 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.

Outline

1. Parametric and Nonparametric Models
2. Nearest Neighbor Classification
3. K Nearest Neighbors Classification
4. Practical Aspects of NN

Nearest Neighbor Classification

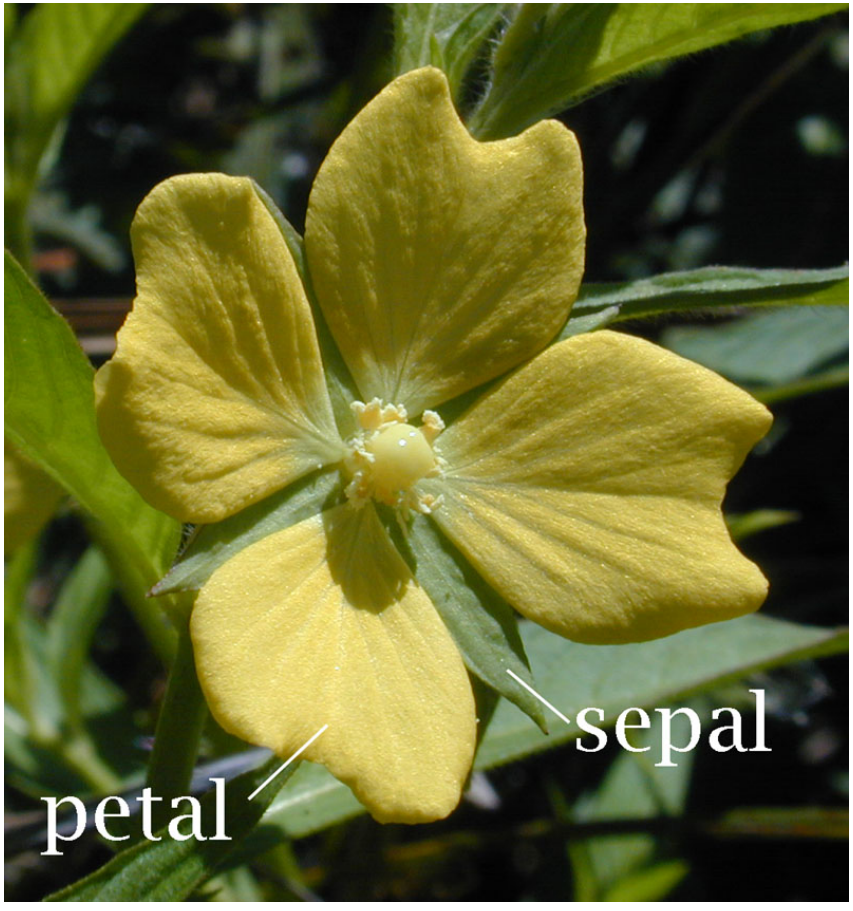
Recognizing Flowers

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

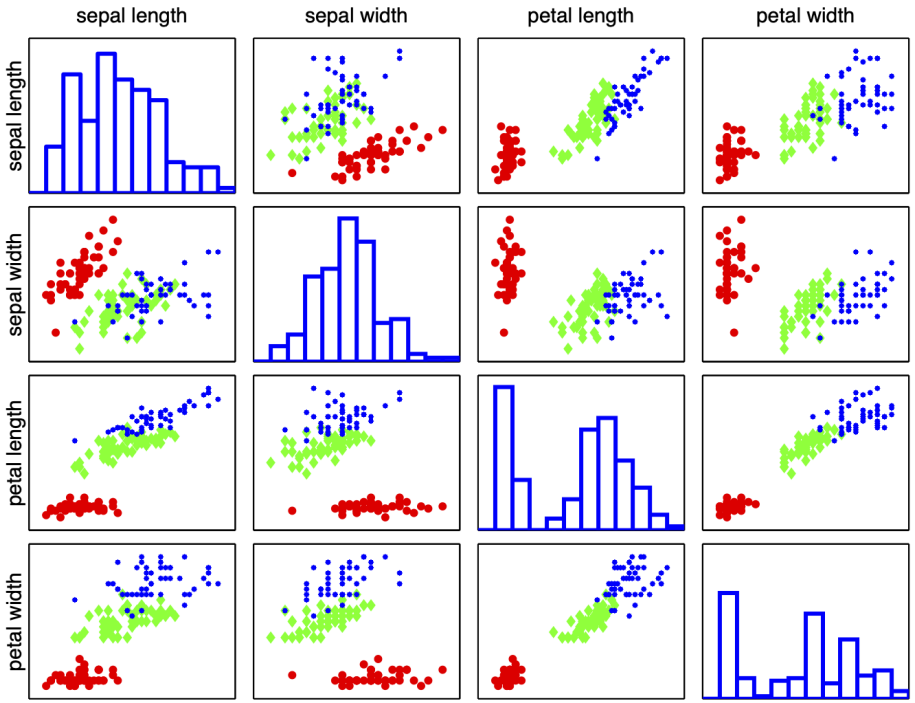
Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
4.6	3.1	1.5	0.2	*I. setosa*
5.0	3.6	1.4	0.2	*I. setosa*
5.4	3.9	1.7	0.4	*I. setosa*
4.6	3.4	1.4	0.3	*I. setosa*
5.0	3.4	1.5	0.2	*I. setosa*
4.4	2.9	1.4	0.2	*I.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

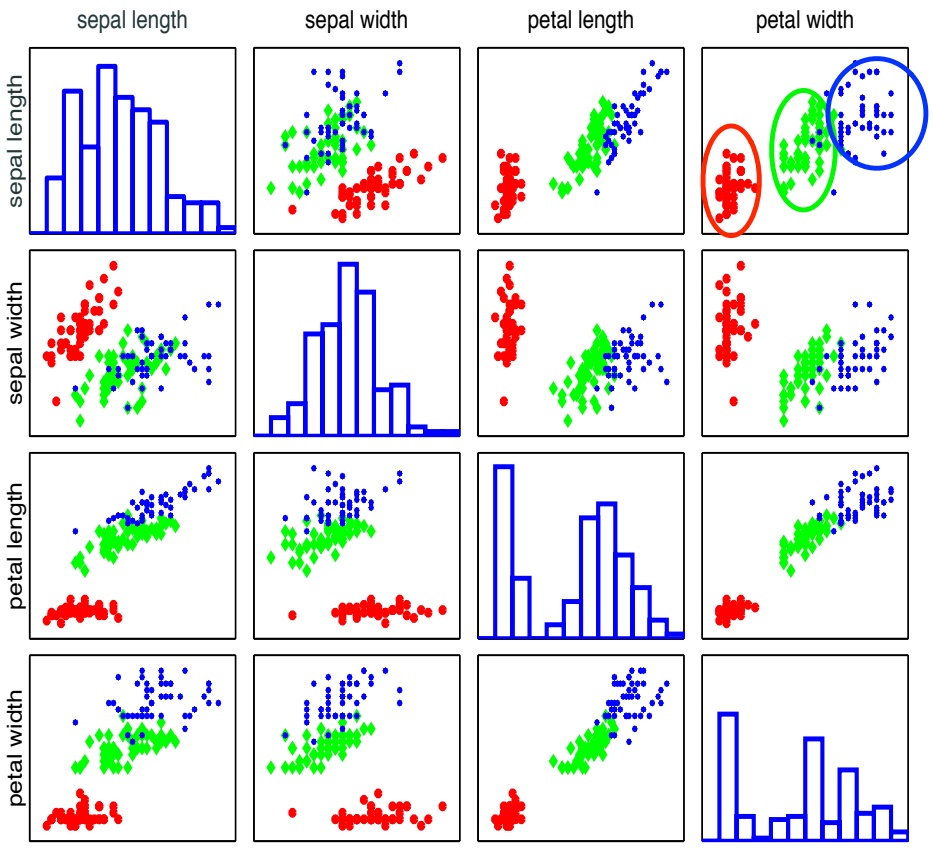
Which combination of features separates the three classes?

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

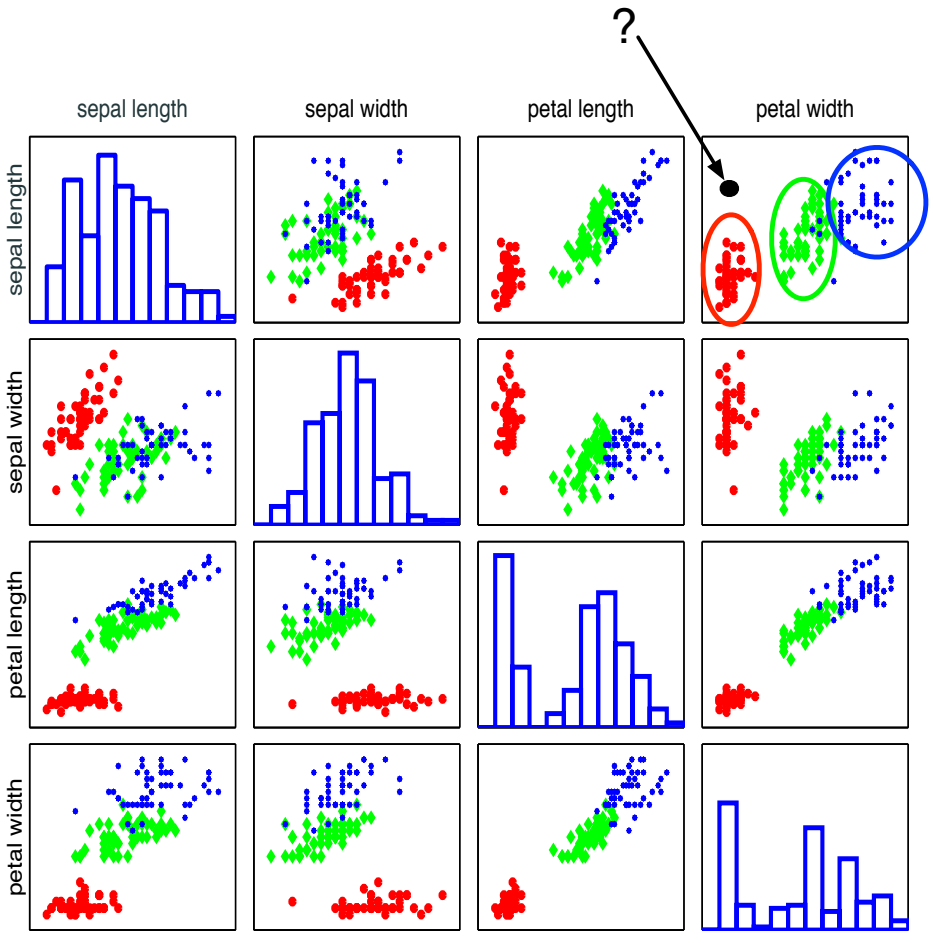


Different Types Seem Well-clustered and Separable

Using two features: petal width and sepal length



Labeling an Unknown Flower Type



Closer to red cluster: so labeling it as setosa

Nearest Neighbor Classification (NNC)

Training data (set)

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

Nearest neighbor of a test data point x

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

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

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

Classification rule

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

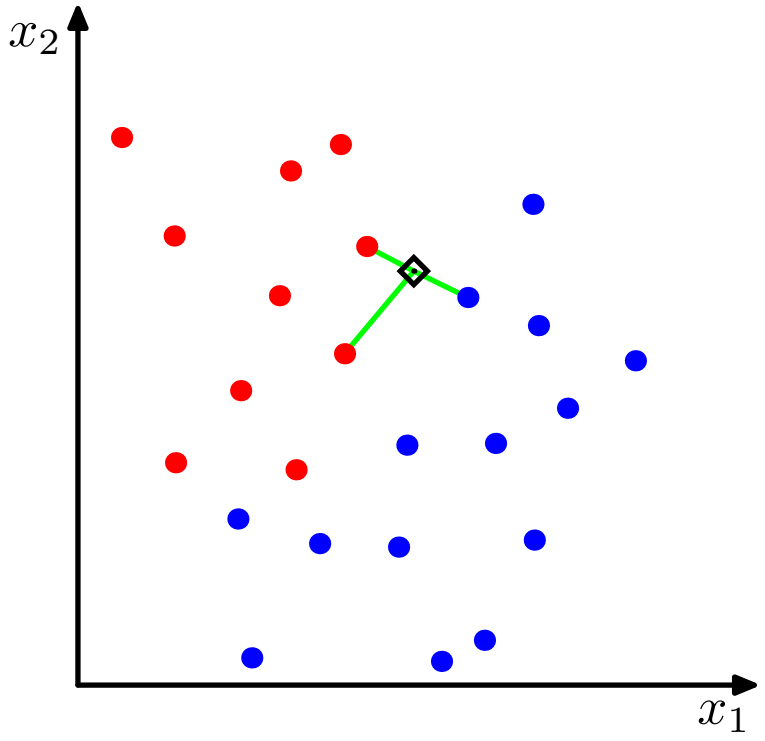
Example: if $\text{nn}(\boldsymbol{x}) = 2$, then

$$y_{\text{nn}(\boldsymbol{x})} = y_2,$$

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)

Example: Classify Iris with Two Features

Training data

ID (n)	petal width (x_1)	sepal length (x_2)	category (y)
1	0.2	5.1	setosa
2	1.4	7.0	versicolor
3	2.5	6.7	virginica

Flower with unknown category

petal width $= 1.8$ and sepal length $= 6.4$

Calculating distance from (x_1, x_2) to (x_{n1}, x_{n2}): $(x_1 - x_{n1})^2 + (x_2 - x_{n2})^2$

ID	distance
1	4.25
2	0.52
3	0.58

Thus, the predicted category is *versicolor*.

How to Measure "Nearness"?

Previously, we used Euclidean distance

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

We can also use alternative distances

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

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

- Or, the ℓ_∞ (supremum) distance:

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

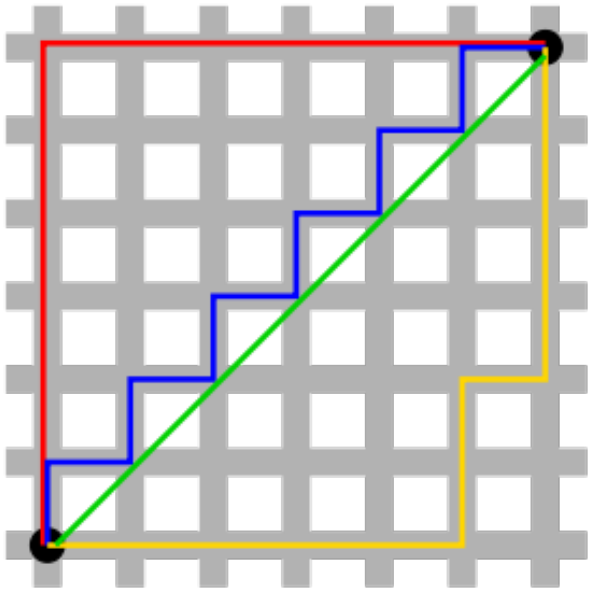


Figure 2: Green line is Euclidean distance. Red, Blue, and Yellow lines are L_1 distance

Nearest Neighbors for Regression

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

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

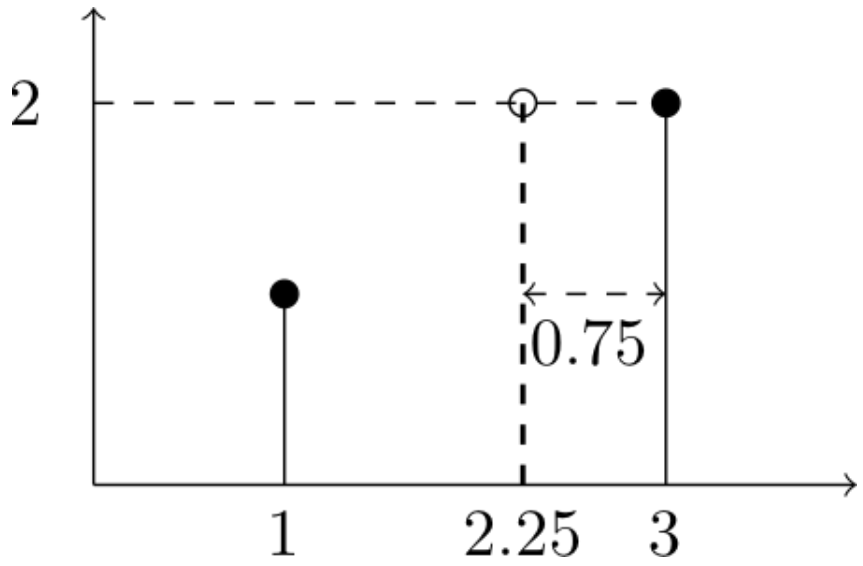
where $\mathrm{nn}(\boldsymbol{x}) \in [\mathrm{N}] = \{1, 2, \cdots, \mathrm{N}\}$ indexes a training instance

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

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.

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*.

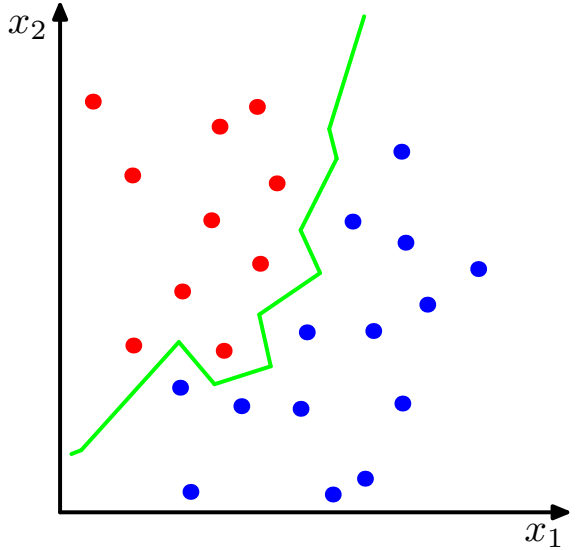
Outline

1. Parametric and Nonparametric Models
2. Nearest Neighbor Classification
3. K Nearest Neighbors Classification
4. Practical Aspects of NN

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

The set of K-nearest neighbors

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

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

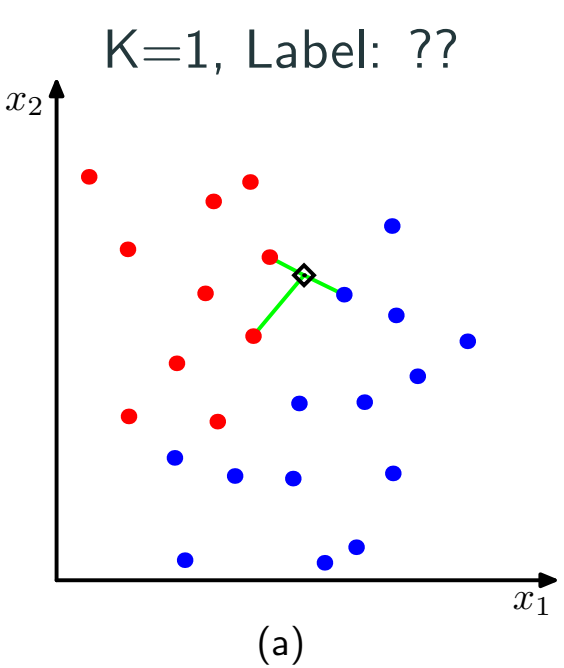
  We use the *indicator function* $\mathbb{I}(y_n == c)$ to represent the votes.
- Aggregate everyone's vote

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

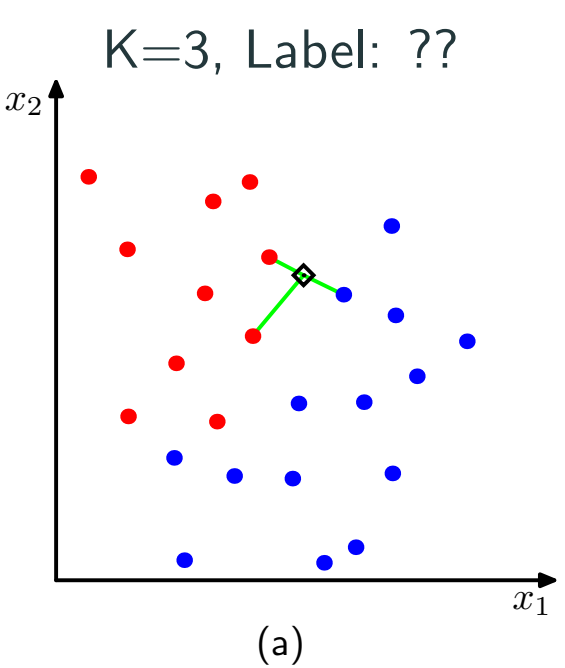
- Label with the majority, breaking ties arbitrarily

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

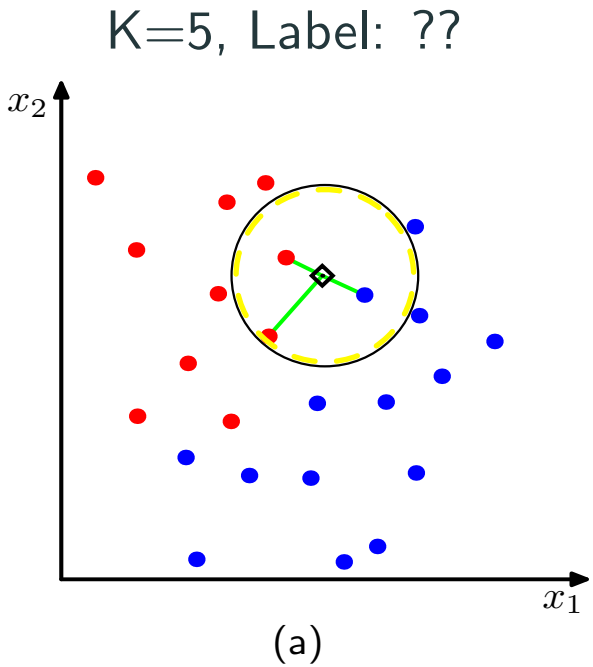
Example



(a)

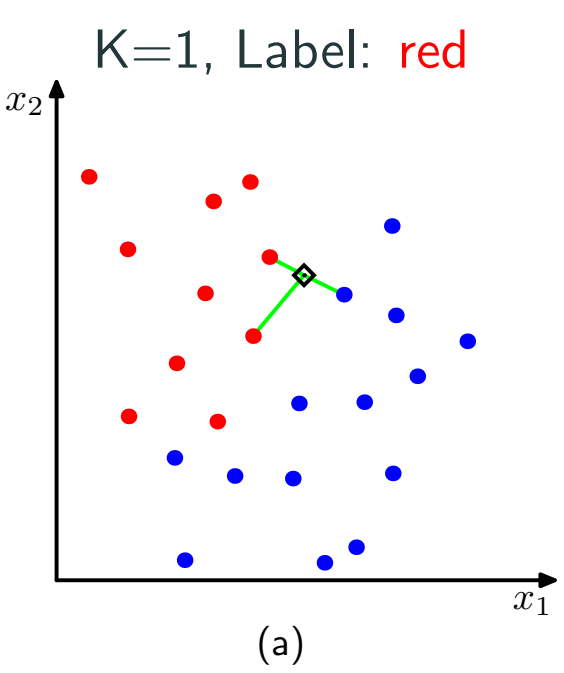


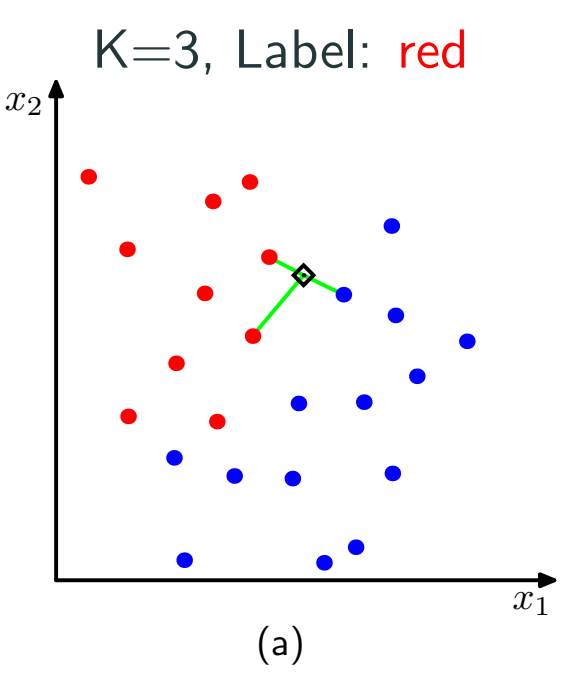
(a)

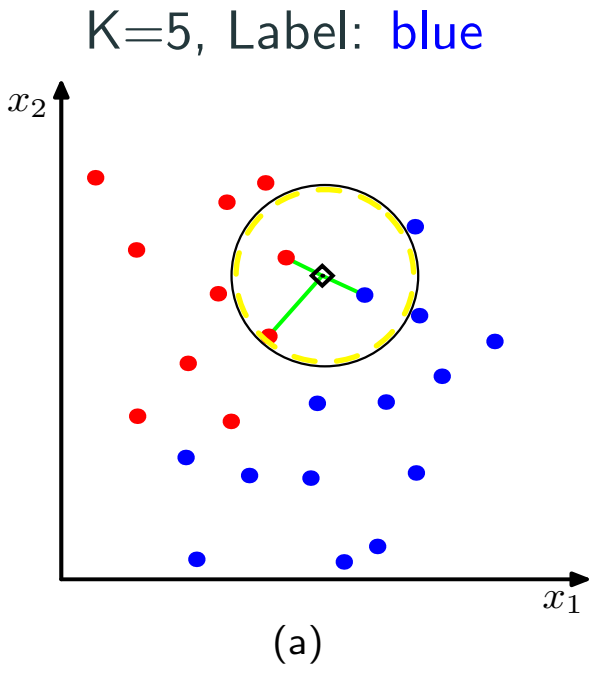


(a)

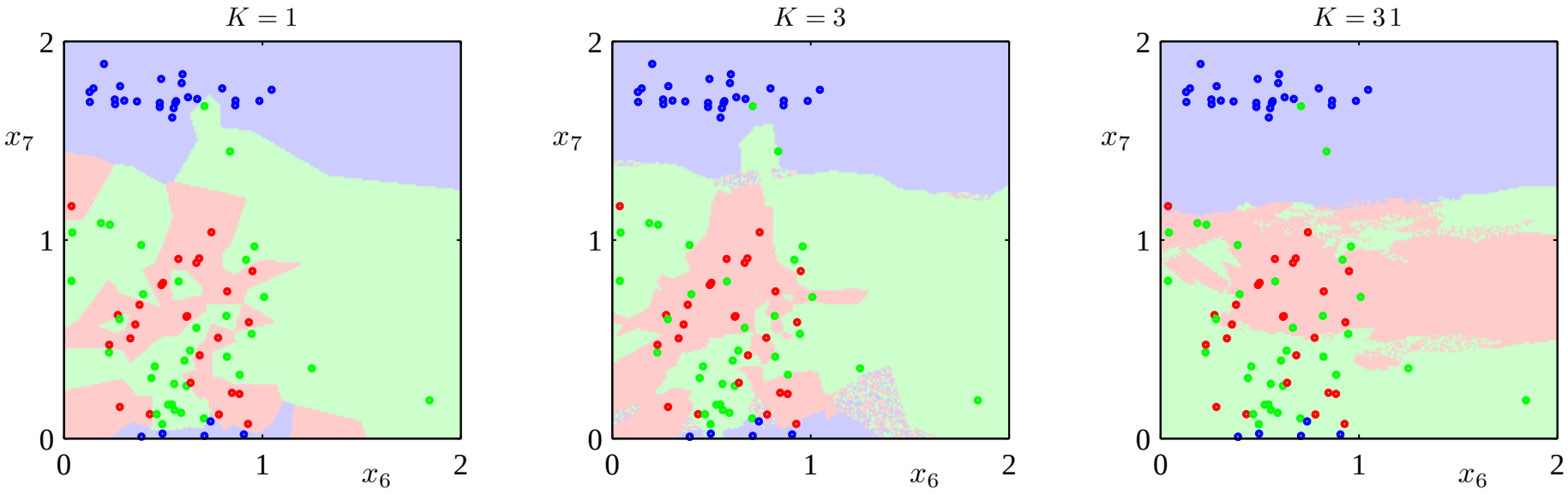
Example







How to Choose an Optimal K?

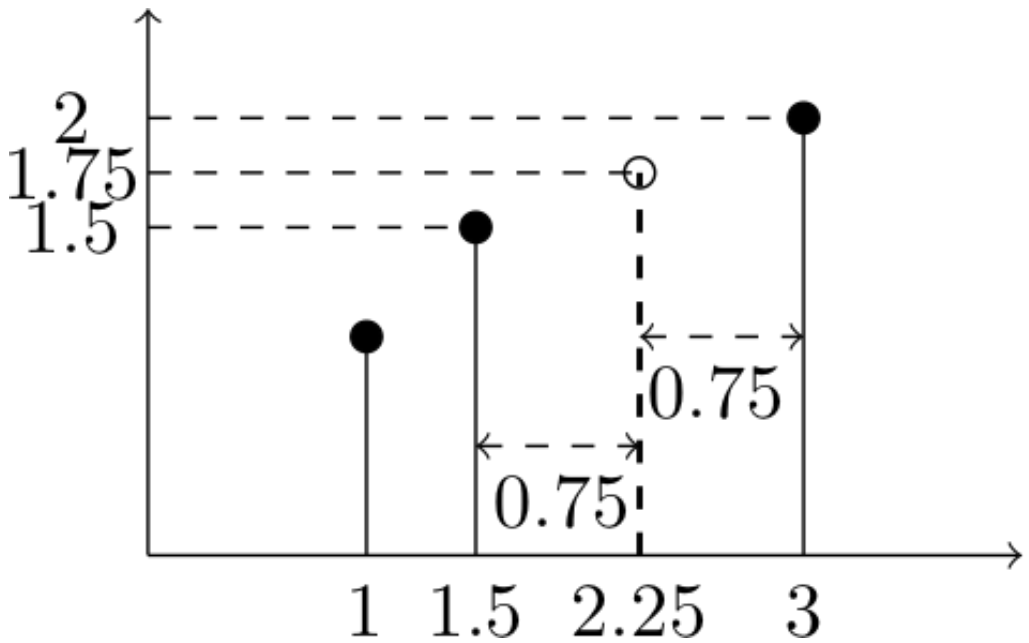


- When K increases, the decision boundary becomes smoother and less susceptible to outliers.
- However, if K is too large, it can also lead to misclassification as we are taking votes from faraway training points.

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.

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



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

Disadvantages of NNC continued

- Relies on the existence of training data points "close" to test points.
- Can break down if the classes are unbalanced.

Outline

1. Parametric and Nonparametric Models
2. Nearest Neighbor Classification
3. K Nearest Neighbors Classification
4. Practical Aspects of NN

Practical Aspects of NN

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 — selecting them requires empirical studies and are task/dataset-specific.

Hyperparameter Tuning on a Validation Dataset

Training data

- N samples/instances: $\mathcal{D}^{\text{TRAIN}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{N}}, y_{\text{N}})\}$
- They are used for learning $f(\cdot)$

Test data

- M samples/instances: $\mathcal{D}^{\text{TEST}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{M}}, y_{\text{M}})\}$
- They are used for assessing how well $f(\cdot)$ will do in predicting an unseen $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Validation data

- L samples/instances: $\mathcal{D}^{\text{VAL}} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_{\text{L}}, y_{\text{L}})\}$
- They are used to optimize hyperparameter(s).

Training data, validation and test data should *not* overlap!

Recipe

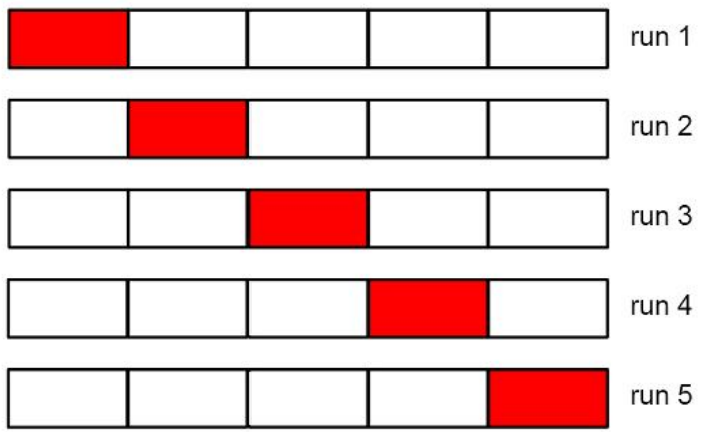
- For each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$ (we don't need this step for NNC)
  - Evaluate the performance of the model on $\mathcal{D}^{\text{VAL}}$
- Choose the model with the best performance on $\mathcal{D}^{\text{VAL}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

Cross-Validation

What if we do not have validation data?

- We split the training data into S equal parts.
- We use each part *in turn* as a validation dataset and use the others as a training dataset.
- We choose the hyperparameter such that the model performs the best across all runs (based on e.g. average.)

Figure 3: S = 5: 5-fold cross validation



Tip: Preprocess Data

Distances depend on units of the features!

Tip: Preprocess 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.