Nearest Neighbor Classification

Professor Ameet Talwalkar

Outline

1. Administration
2. First learning algorithm: Nearest neighbor classifier
3. Deeper understanding of NNC
4. Some practical aspects of NNC
5. What we have learned

Registration / PTEs

- Course is currently full
- **I am not giving out PTEs** until after HW1 submission date
- I expect several students will drop the course
  - ML is very popular, so many students are interested
  - Many students don't have mathematical maturity for graduate level material
  - Last year's attrition rate much higher than typical grad-level class
- If you're not registered, I'd encourage you to stay patient
  - I am confident that all qualified students will be able to enroll

Math Quiz

- Representative of math concepts you are excepted to know
- Math quiz grades available next week at office hours
  - Graded to assess your background (but not part of final grade)
  - We may contact students who perform poorly
- Math Quiz is a requirement
  - We will not grade your HW if you don't take it
  - You can take it in office hours if you haven't already
- Be honest / realistic with yourself about your background
  - It's better for you, me, and your classmates to drop the course now rather than a month from now

Homework 1

- Available online, and due next Wednesday at beginning of class
- Also representative of math concepts you are excepted to know
  - Basic math questions, MATLAB coding assignment, Academic Integrity Form
  - Look on course website for details about getting access to MATLAB
- Submission details are included in the assignment
- Join Piazza! (students are asking questions already)

Outline

1. Administration
2. First learning algorithm: Nearest neighbor classifier
   - Intuitive example
   - General setup for classification
   - Algorithm
3. Deeper understanding of NNC
4. Some practical aspects of NNC
5. What we have learned

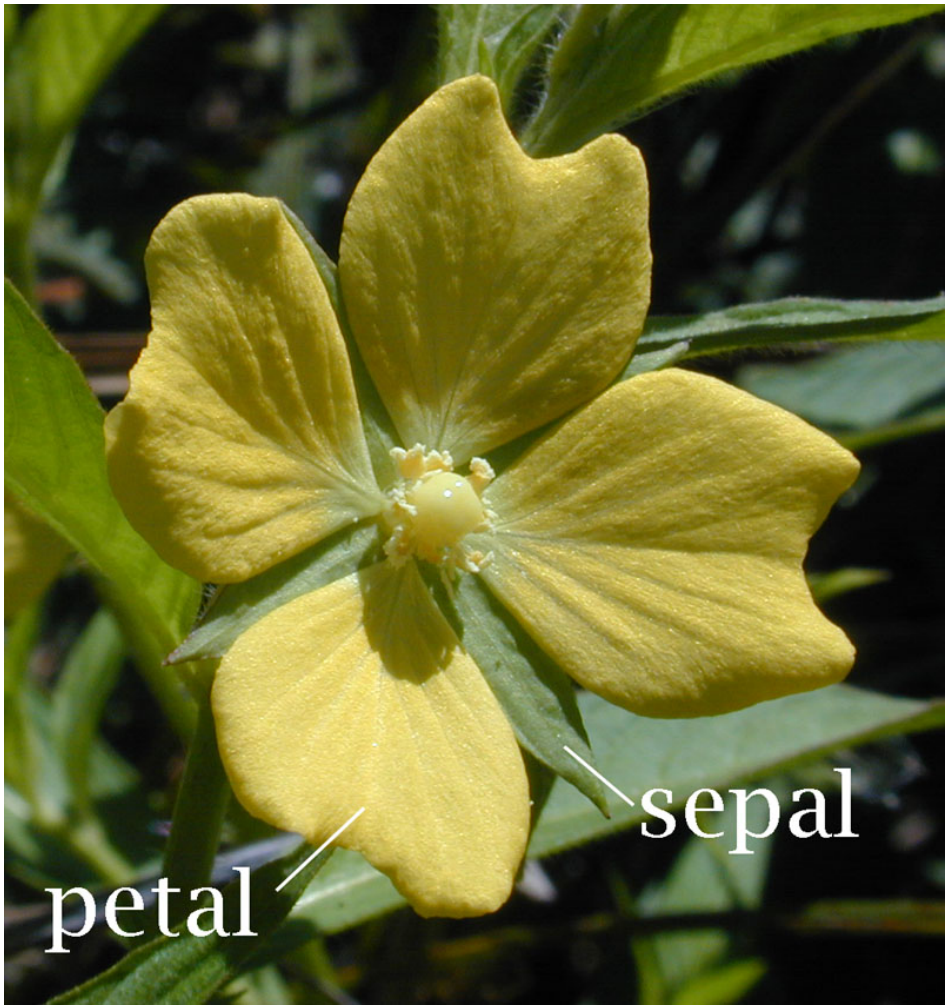
Recognizing flowers

Types of Iris: setosa, versicolor, and virginica



Measuring the properties of the flowers

Features: the widths and lengths of sepal and petal



Often, data is conveniently organized as a table

Ex: Iris data (click here for all data)

- 4 features
- 3 classes

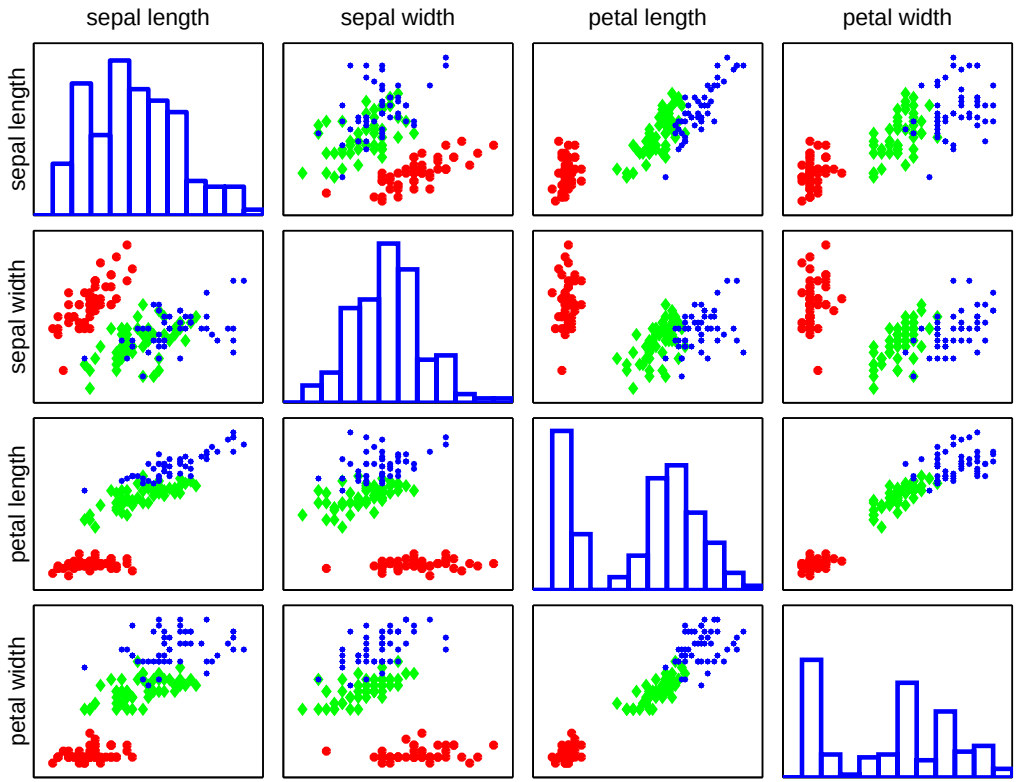
Fisher's *Iris* Data

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

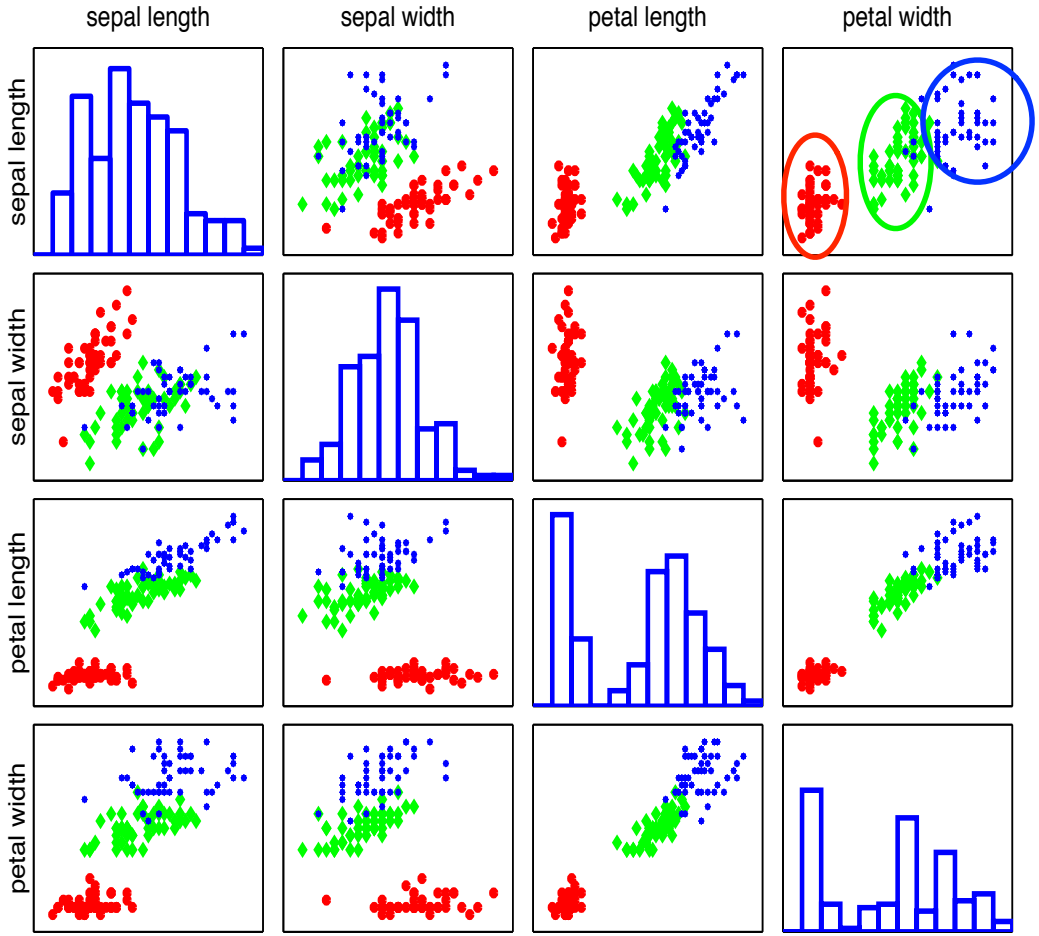
Visualization of data helps to identify the right learning model to use

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

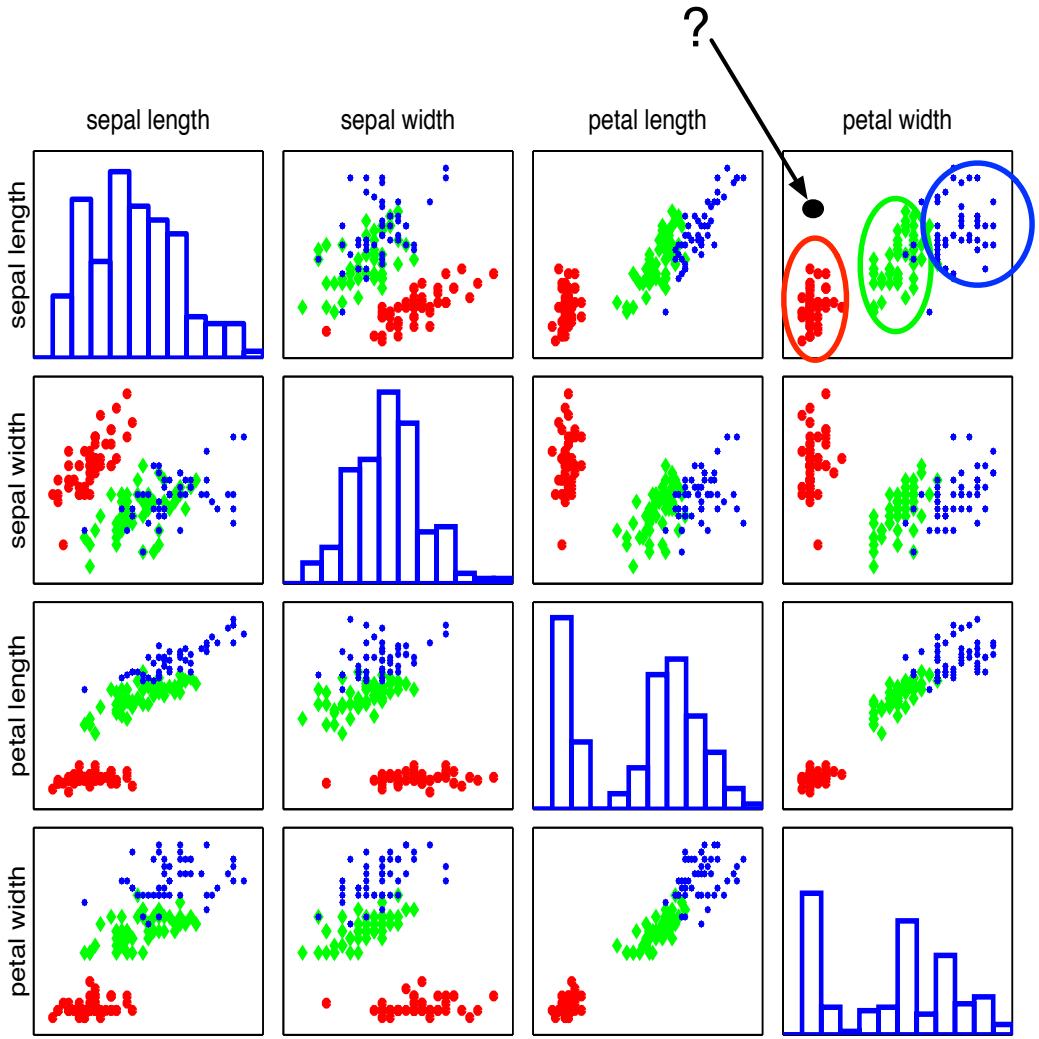


Different types seem well-clustered and separable

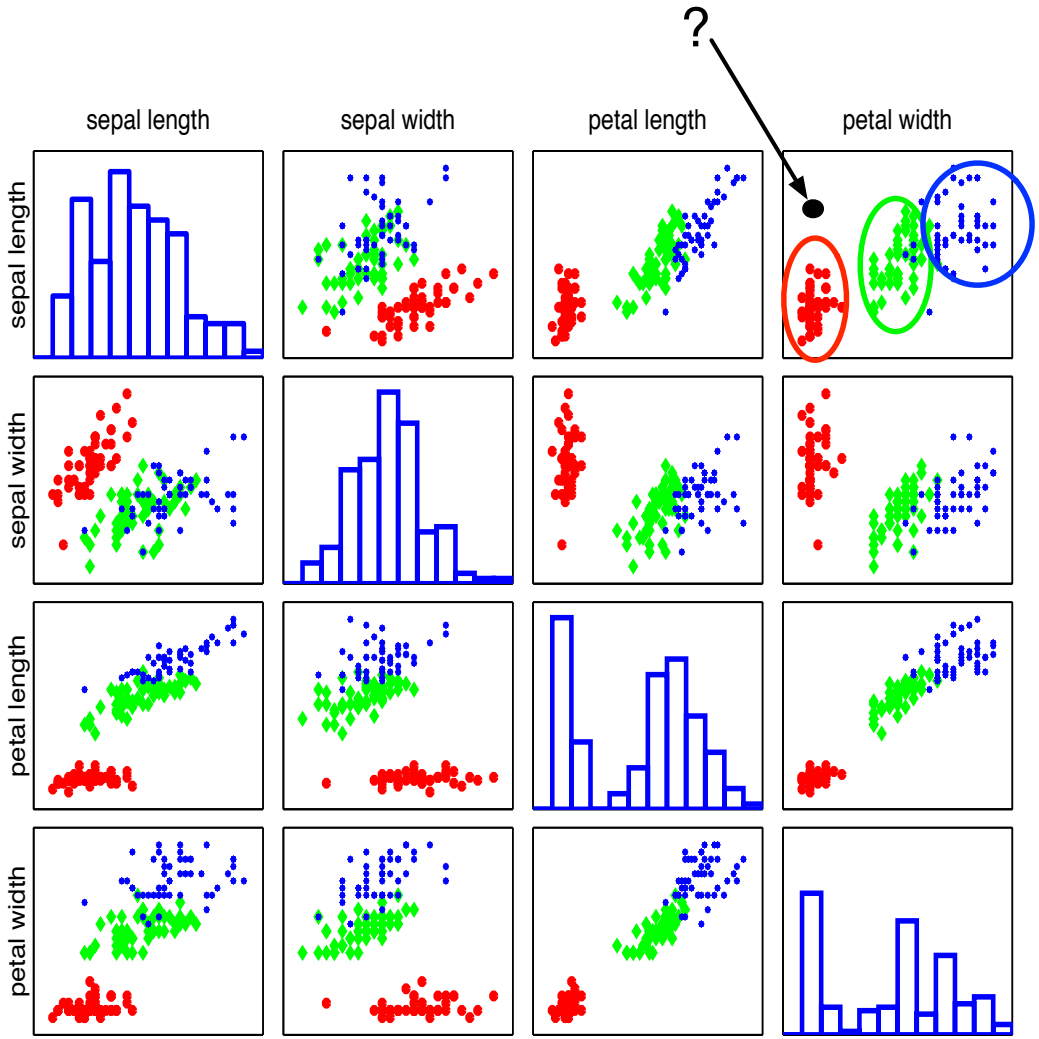
Using two features: petal width and sepal length



Labeling an unknown flower type



Labeling an unknown flower type



Closer to red cluster: so labeling it as setosa

Multi-class classification

Classify data into one of the multiple categories

- Input (feature vectors): $\boldsymbol{x} \in \mathbb{R}^{\mathsf{D}}$
- Output (label): $y \in [\mathsf{C}] = \{1, 2, \cdots, \mathsf{C}\}$
- Learning goal: $y = f(\boldsymbol{x})$

Special case: binary classification

- Number of classes: $\mathsf{C} = 2$
- Labels: $\{0, 1\}$ or $\{-1, +1\}$

More terminology

Training data (set)

- 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 (evaluation) 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 $\boldsymbol{x} \notin \mathcal{D}^{\text{TRAIN}}$

Training data and test data should *not* overlap: $\mathcal{D}^{\text{TRAIN}} \cap \mathcal{D}^{\text{TEST}} = \emptyset$

Nearest neighbor classification (NNC)

Nearest neighbor

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

where $\mathsf{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$, i.e., the index to one of the training instances

Nearest neighbor classification (NNC)

Nearest neighbor

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

where $\mathsf{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$, i.e., the index to one of the training instances

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

Nearest neighbor classification (NNC)

Nearest neighbor

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

where $\mathsf{nn}(\boldsymbol{x}) \in [\mathsf{N}] = \{1, 2, \cdots, \mathsf{N}\}$, i.e., the index to one of the training instances

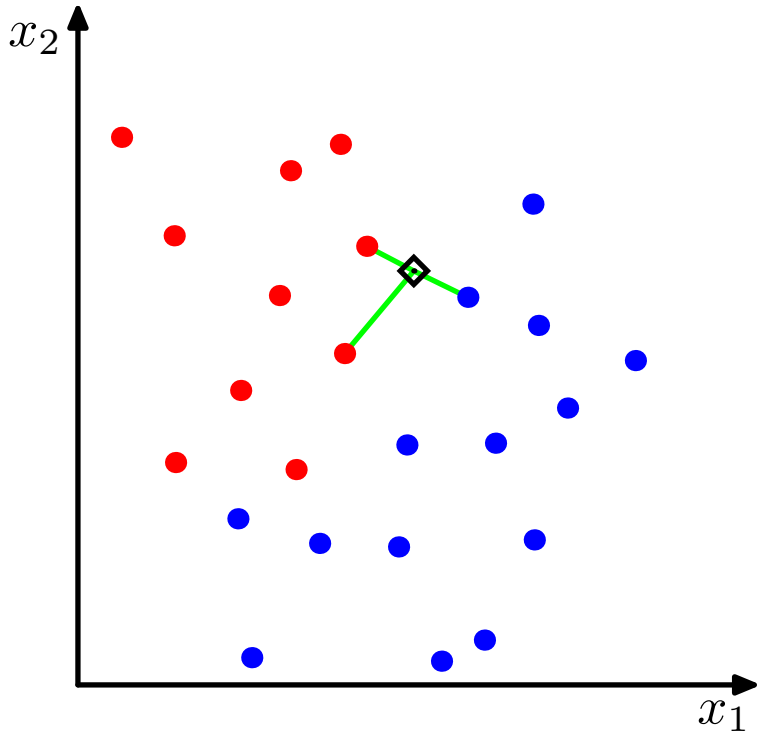
$$\mathsf{nn}(\boldsymbol{x}) = \arg\min_{n \in [\mathsf{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2 = \arg\min_{n \in [\mathsf{N}]} \sum_{d=1}^{\mathsf{D}} (x_d - x_{nd})^2$$

Classification rule

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

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

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 width $= 6.4$

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 width $= 6.4$

Calculating distance $= \sqrt{x_1 - x_{n1})^2 + (x_2 - x_{n2})^2}$

ID	distance
1	1.75
2	0.72
3	0.76

Thus, the predicted category is *versicolor* (the real category is *virginica*)

How to measure nearness with other distances?

Previously, we use the Euclidean distance

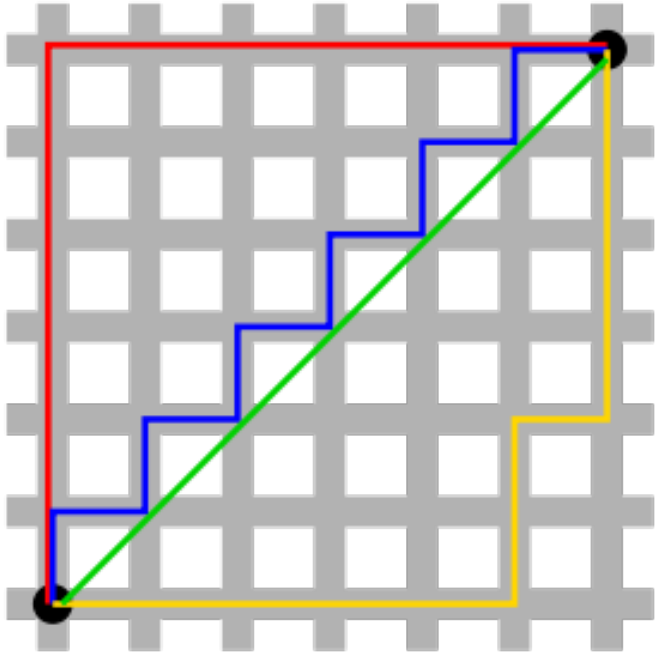
$$\text{nn}(\boldsymbol{x}) = \arg\min_{n \in [\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2$$

We can also use alternative distances

E.g., the following L_1 distance (i.e., city block distance, or Manhattan distance)

$$\text{nn}(\boldsymbol{x}) = \arg\min_{n \in [\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_1$$

$$= \arg\min_{n \in [\text{N}]} \sum_{d=1}^{\text{D}} |x_d - x_{nd}|$$



How to measure nearness with other distances?

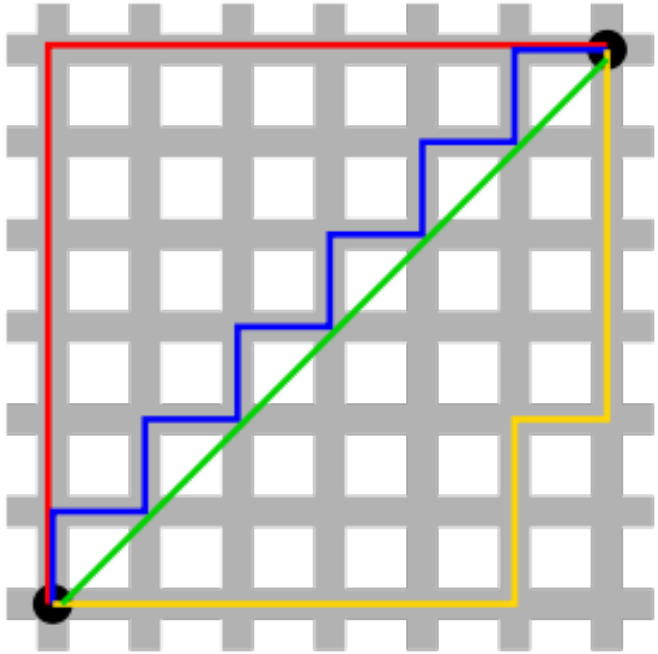
Previously, we use the Euclidean distance

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

We can also use alternative distances

E.g., the following L_1 distance (i.e., city block distance, or Manhattan distance)

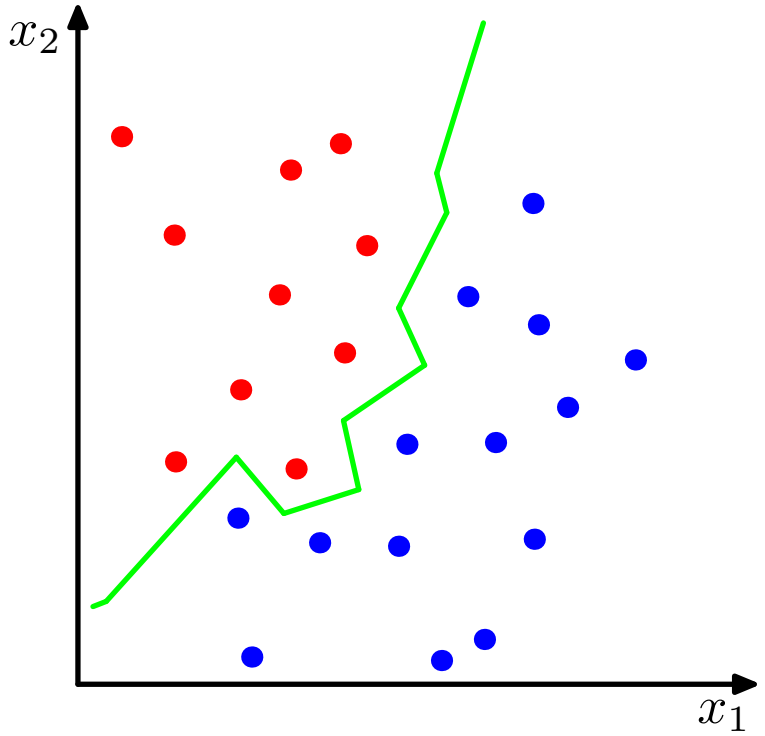
$$\begin{aligned}\text{nn}(\boldsymbol{x}) &= \arg\min_{n \in [\text{N}]} \|\boldsymbol{x} - \boldsymbol{x}_n\|_1 \\ &= \arg\min_{n \in [\text{N}]} \sum_{d=1}^{\text{D}} |x_d - x_{nd}|\end{aligned}$$



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

Decision boundary

For every point in the space, we can determine its label using the NNC rule. This gives rise to a *decision boundary* that partitions the space into different regions.



(b)

K-nearest neighbor (KNN) classification

Increase the number of nearest neighbors to use?

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

K-nearest neighbor (KNN) classification

Increase the number of nearest neighbors to use?

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

The set of K-nearest neighbor

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

Let $\boldsymbol{x}(k) = \boldsymbol{x}_{\mathsf{nn}_k(\boldsymbol{x})}$, 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?

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

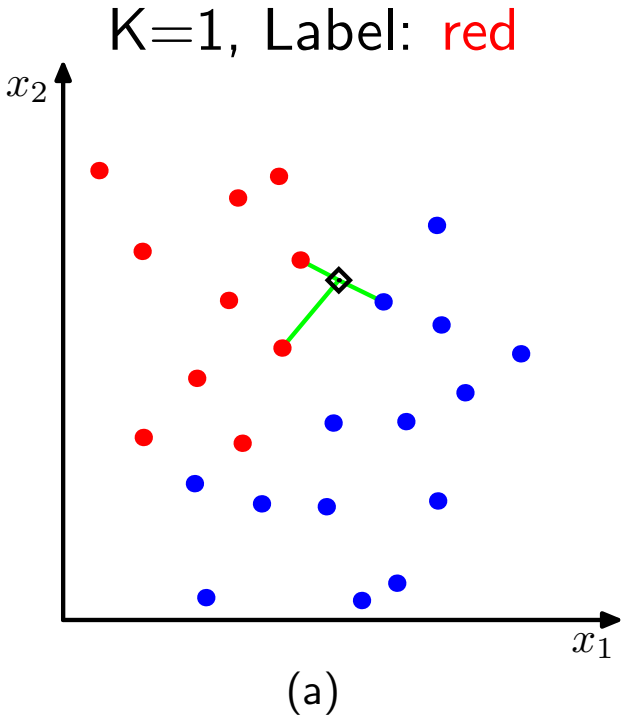
  We use the *indicator function* $\mathbb{I}(y_n == c)$ to represent.
- Aggregate everyone's vote

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

- Label with the majority

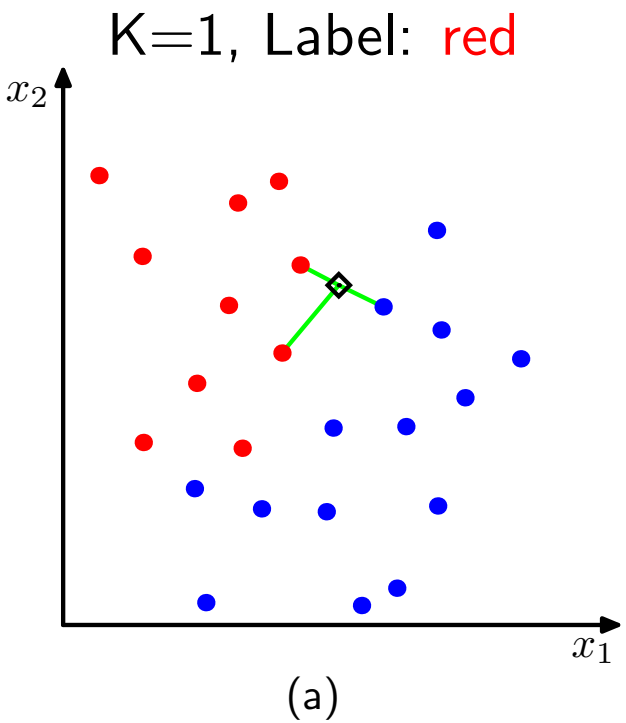
$$y = f(\boldsymbol{x}) = \arg\max_{c \in [\text{C}]} v_c$$

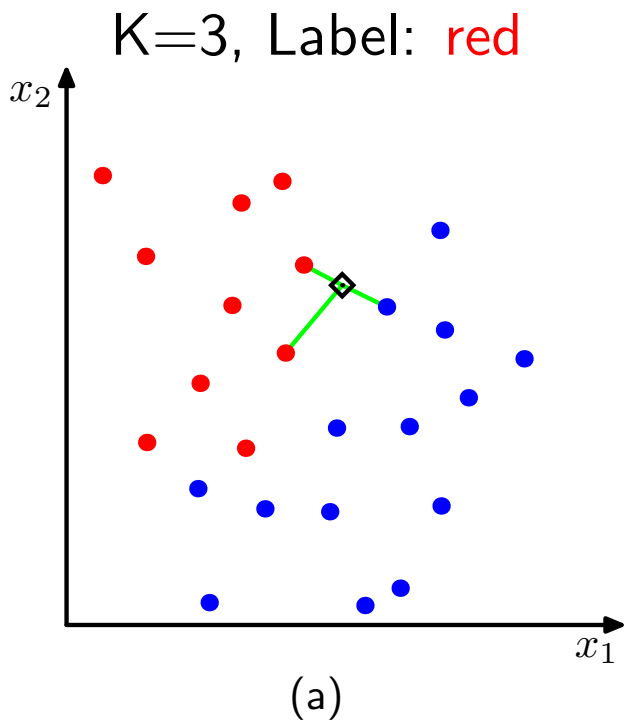
Example



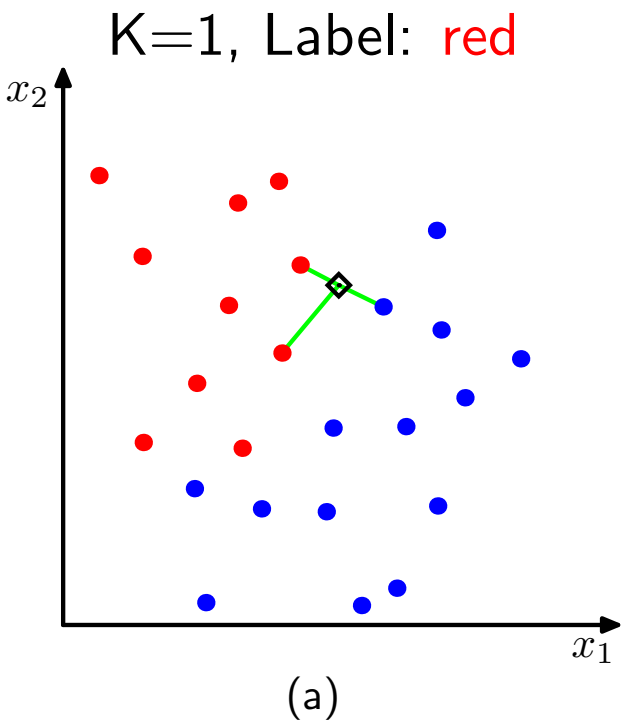
(a)

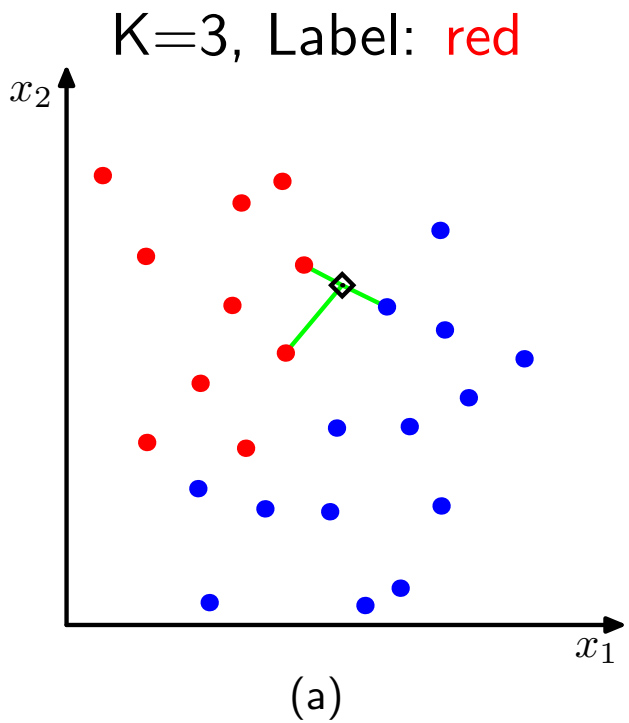
Example

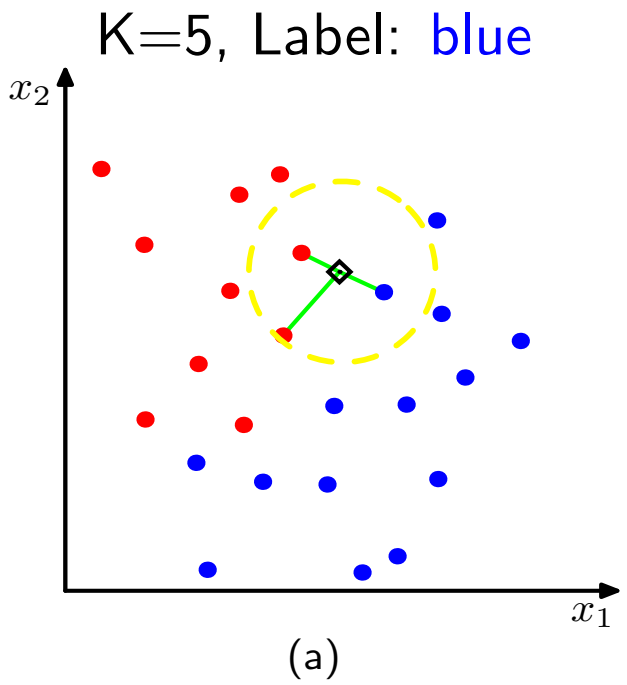




Example

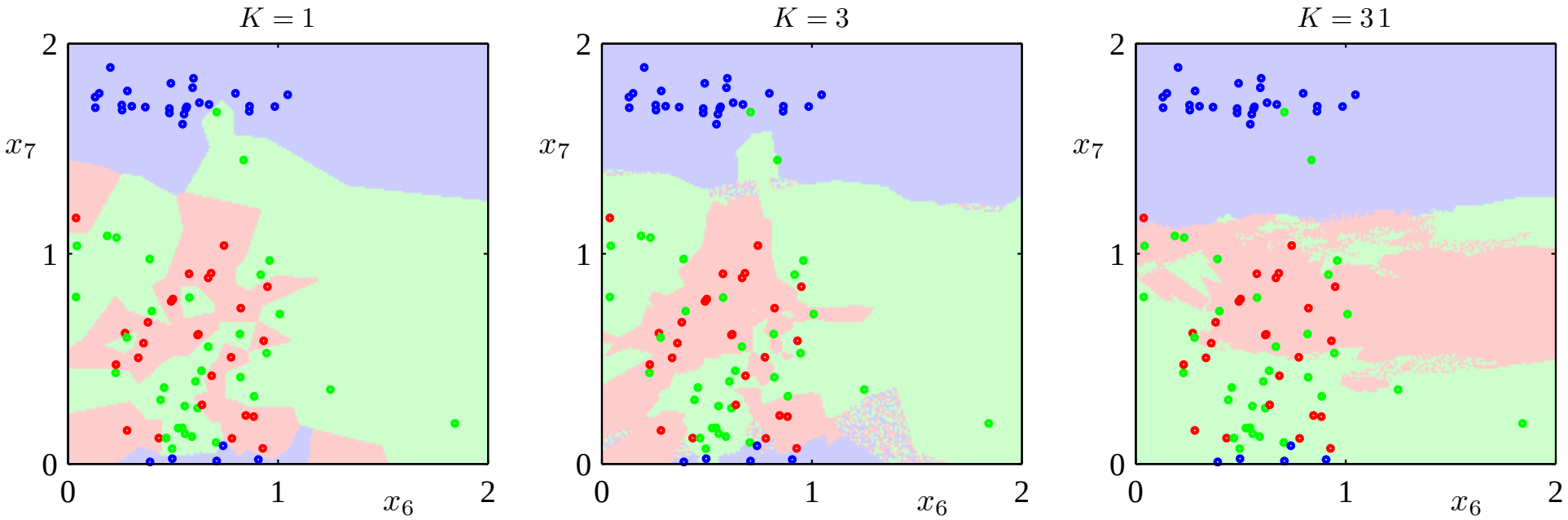






How to choose an optimal K?

How to choose an optimal K?



When K increases, the decision boundary becomes smooth.

Mini-summary

Advantages of NNC

- Computationally, simple and easy to implement – just compute distances
- Has strong theoretical guarantees that it is "doing the right thing"

Mini-summary

Advantages of NNC

- Computationally, simple and easy to implement – just compute distances
- Has strong theoretical guarantees that it is "doing the right thing"

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. This type of method is called *nonparametric*.
- Choosing the right distance measure and K can be involved.

Outline

1. Administration
2. First learning algorithm: Nearest neighbor classifier
3. Deeper understanding of NNC
   - Measuring performance
   - The ideal classifier
   - Comparing NNC to the ideal classifier
4. Some practical aspects of NNC
5. What we have learned

Is NNC too simple to do the right thing?

To answer this question, we proceed in 3 steps

1. We define a performance metric for a classifier/algorithm.
2. We then propose an ideal classifier.
3. We then compare our simple NNC classifier to the ideal one and show that it performs nearly as well.

How to measure performance of a classifier?

Intuition

We should compute accuracy — the percentage of data points being correctly classified, or the error rate — the percentage of data points being incorrectly classified.

Two versions: which one to use?

- Defined on the training data set

$$A^{\text{TRAIN}} = \frac{1}{\mathsf{N}} \sum_n \mathbb{I}[f(\boldsymbol{x}_n) == y_n], \quad \varepsilon^{\text{TRAIN}} = \frac{1}{\mathsf{N}} \sum_n \mathbb{I}[f(\boldsymbol{x}_n) \neq y_n]$$

How to measure performance of a classifier?

Intuition

We should compute accuracy — the percentage of data points being correctly classified, or the error rate — the percentage of data points being incorrectly classified.

Two versions: which one to use?

- Defined on the training data set

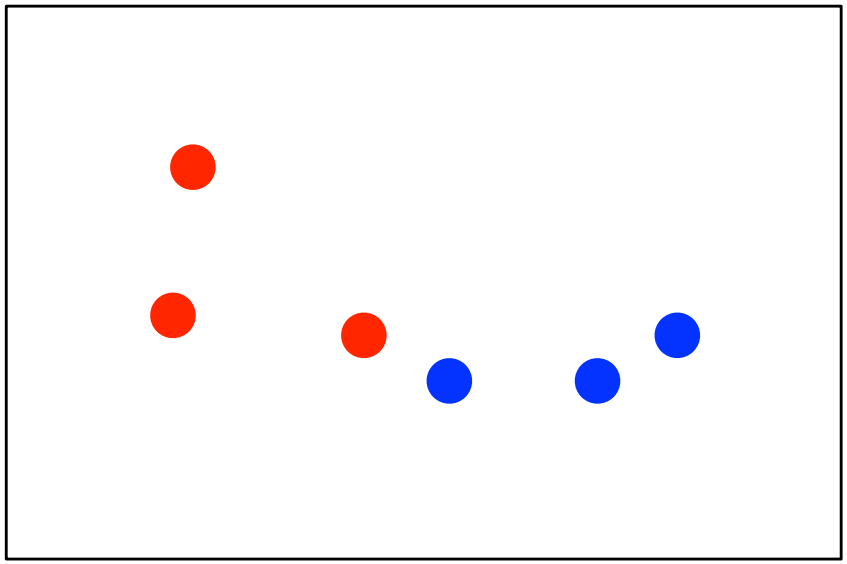
$$A^{\text{TRAIN}} = \frac{1}{\text{N}} \sum_{n} \mathbb{I}[f(\boldsymbol{x}_n) == y_n], \quad \varepsilon^{\text{TRAIN}} = \frac{1}{\text{N}} \sum_{n} \mathbb{I}[f(\boldsymbol{x}_n) \neq y_n]$$

- Defined on the test (evaluation) data set

$$A^{\text{TEST}} = \frac{1}{\text{M}} \sum_{m} \mathbb{I}[f(\boldsymbol{x}_m) == y_m], \quad \varepsilon^{\text{TEST}} = \frac{1}{\text{M}} \sum_{M} \mathbb{I}[f(\boldsymbol{x}_m) \neq y_m]$$

Example

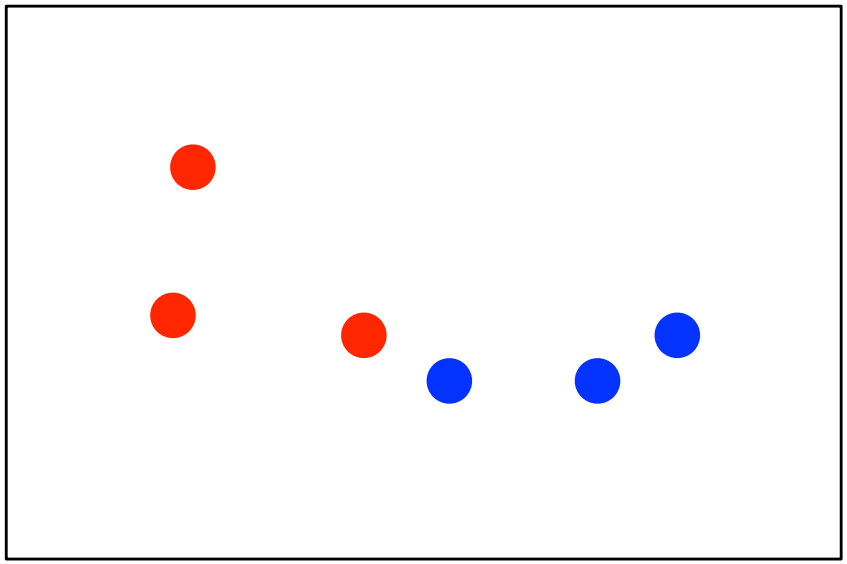
Training data



What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

Example

Training data

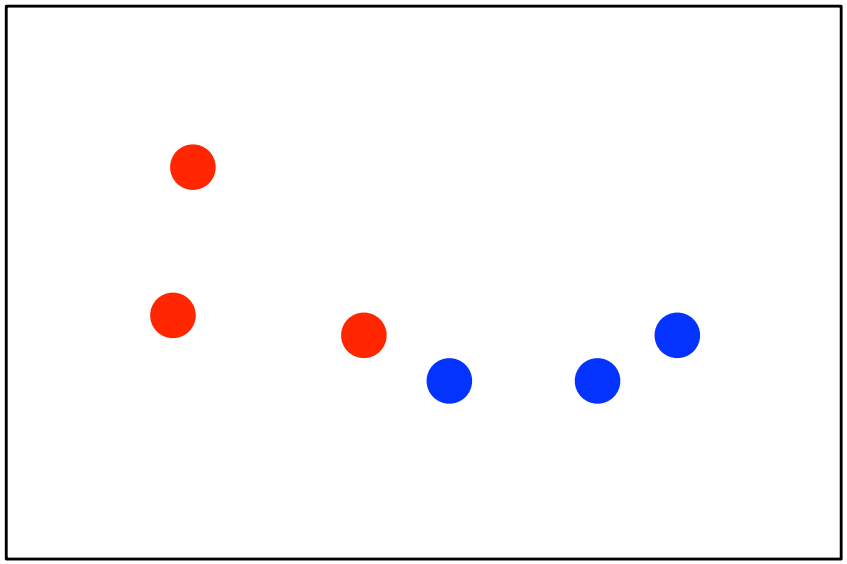


What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

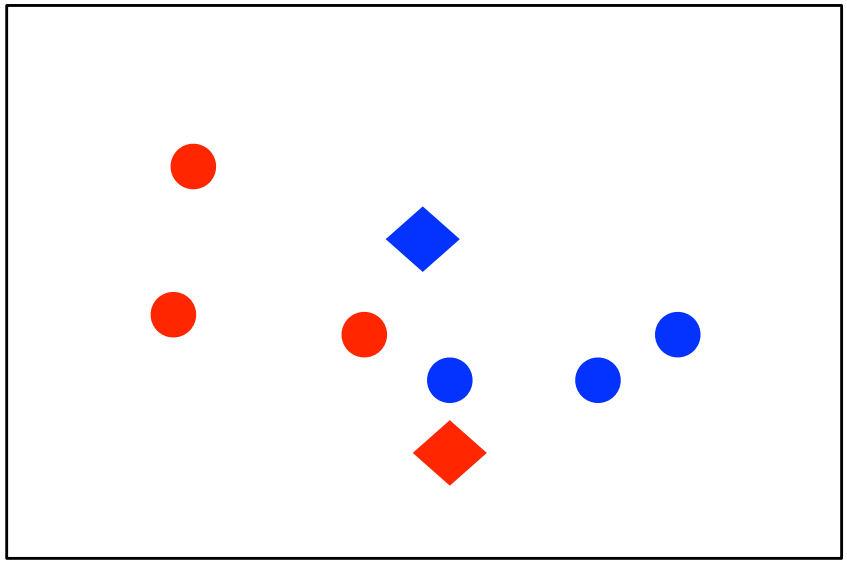
$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

Example

Training data



Test data



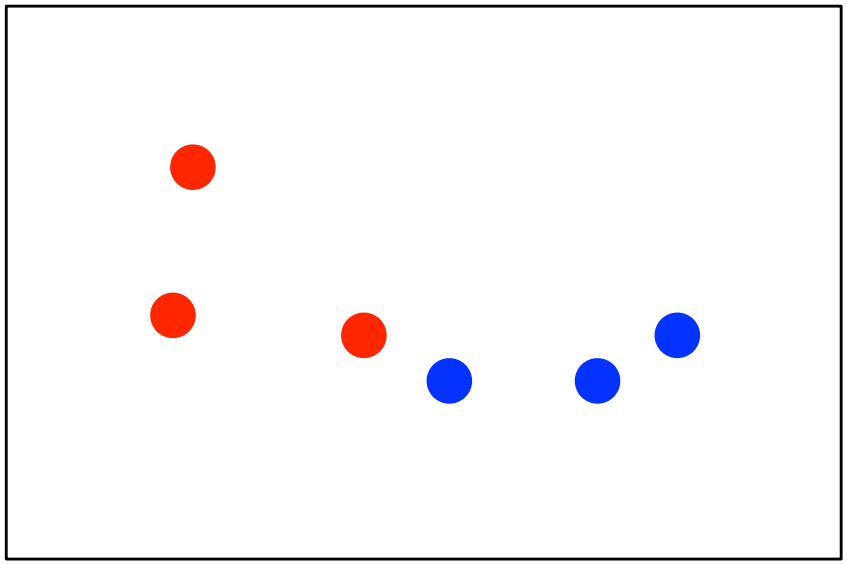
What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

What are A^{TEST} and $\varepsilon^{\text{TEST}}$?

$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

Example

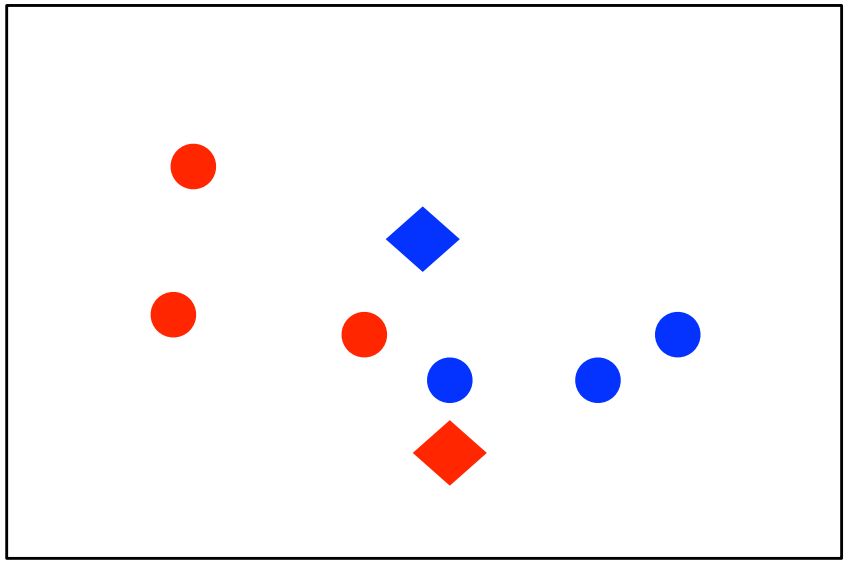
Training data



What are A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 100\%, \quad \varepsilon^{\text{TRAIN}} = 0\%$$

Test data



What are A^{TEST} and $\varepsilon^{\text{TEST}}$?

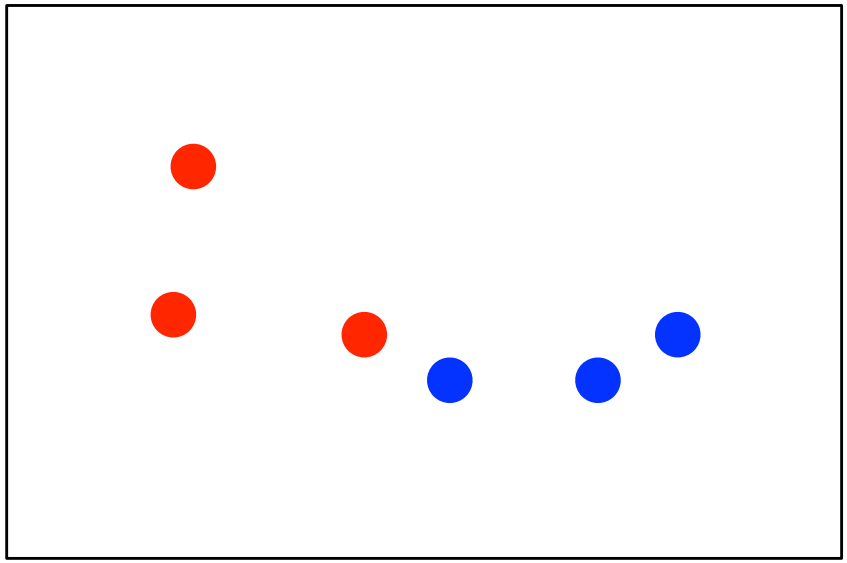
$$A^{\text{TEST}} = 0\%, \quad \varepsilon^{\text{TEST}} = 100\%$$

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will not be itself. So the error rate would not become 0 necessarily.

Training data



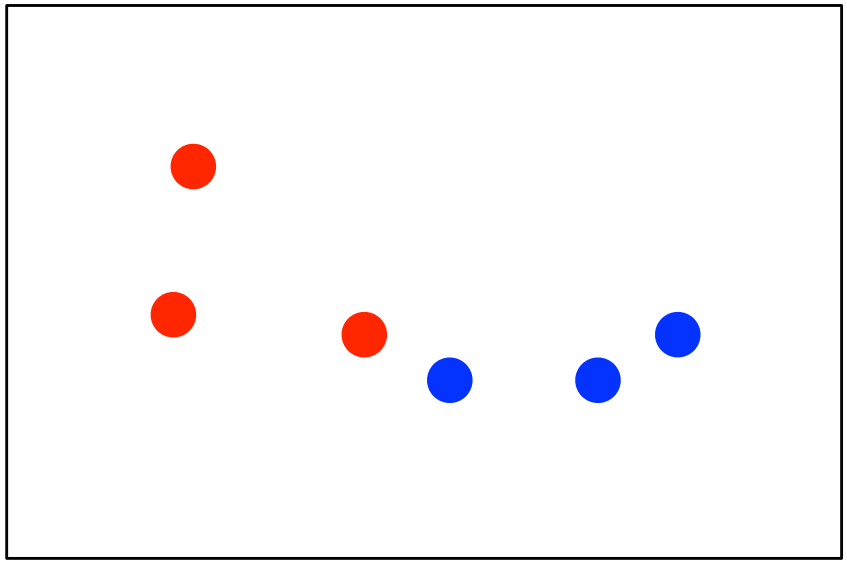
What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

Leave-one-out (LOO)

Idea

- For each training instance $\boldsymbol{x}_n$, take it out of the training set and then label it.
- For NNC, $\boldsymbol{x}_n$'s nearest neighbor will not be itself. So the error rate would not become 0 necessarily.

Training data



What are the LOO-version of A^{TRAIN} and $\varepsilon^{\text{TRAIN}}$?

$$A^{\text{TRAIN}} = 66.67\% (\text{i.e.}, 4/6)$$
$$\varepsilon^{\text{TRAIN}} = 33.33\% (\text{i.e.}, 2/6)$$

Drawback of the metrics

They are dataset-specific

- Given a different training (or test) dataset, A^{TRAIN} (or A^{TEST}) will change.
- Thus, if we get a dataset "randomly", these variables would be random quantities.

$$A^{\text{TEST}}_{\mathcal{D}_1}, A^{\text{TEST}}_{\mathcal{D}_2}, \cdots, A^{\text{TEST}}_{\mathcal{D}_q}, \cdots$$

Drawback of the metrics

They are dataset-specific

- Given a different training (or test) dataset, A^{TRAIN} (or A^{TEST}) will change.
- Thus, if we get a dataset "randomly", these variables would be random quantities.

$$A^{\text{TEST}}_{\mathcal{D}_1}, A^{\text{TEST}}_{\mathcal{D}_2}, \cdots, A^{\text{TEST}}_{\mathcal{D}_q}, \cdots$$

These are called "empirical" accuracies (or errors).

Drawback of the metrics

They are dataset-specific

- Given a different training (or test) dataset, A^{TRAIN} (or A^{TEST}) will change.
- Thus, if we get a dataset "randomly", these variables would be random quantities.

$$A^{\text{TEST}}_{\mathcal{D}_1}, A^{\text{TEST}}_{\mathcal{D}_2}, \cdots, A^{\text{TEST}}_{\mathcal{D}_q}, \cdots$$

These are called "empirical" accuracies (or errors).

Can we understand the algorithm itself in a "more certain" nature, by removing the uncertainty caused by the datasets?

Expected mistakes

Setup

- Assume our data $(\boldsymbol{x}, y)$ is drawn from the joint and *unknown* distribution $p(\boldsymbol{x}, y)$
- Define a classification mistake on a single data point $\boldsymbol{x}$ with the ground-truth label y as:

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$$

Expected mistakes

Setup

- Assume our data $(\boldsymbol{x}, y)$ is drawn from the joint and *unknown* distribution $p(\boldsymbol{x}, y)$
- Define a classification mistake on a single data point $\boldsymbol{x}$ with the ground-truth label y as:

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$$

- Expected classification mistake on a single data point $\boldsymbol{x}$

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

Expected mistakes

Setup

- Assume our data $(\boldsymbol{x}, y)$ is drawn from the joint and *unknown* distribution $p(\boldsymbol{x}, y)$
- Define a classification mistake on a single data point $\boldsymbol{x}$ with the ground-truth label y as:

$$L(f(\boldsymbol{x}), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases}$$

- Expected classification mistake on a single data point $\boldsymbol{x}$

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

- The average classification mistake by the classifier itself

$$R(f) = \mathbb{E}_{\boldsymbol{x} \sim p(\boldsymbol{x})} R(f, \boldsymbol{x}) = \mathbb{E}_{(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)} L(f(\boldsymbol{x}), y)$$

(law of iterated expectations, tower property, smoothing)

Terminology

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.

Terminology

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.
- Expected conditional risk

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

Terminology

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.
- Expected conditional risk

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

- Expected risk

$$R(f) = \mathbb{E}_{(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)} L(f(\boldsymbol{x}), y)$$

Terminology

- $L(f(\boldsymbol{x}), y)$ is called *0/1 loss function* — many other forms of loss functions exist for different learning problems.
- Expected conditional risk

$$R(f, \boldsymbol{x}) = \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y)$$

- Expected risk

$$R(f) = \mathbb{E}_{(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)} L(f(\boldsymbol{x}), y)$$

- Empirical risk

$$R_{\mathcal{D}}(f) = \frac{1}{\mathsf{N}} \sum_{n} L(f(\boldsymbol{x}_n), y_n)$$

(This is our empirical error from earlier.)

Ex: binary classification

Expected conditional risk of a single data point x

$$\begin{aligned} R(f, \boldsymbol{x}) &= \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y) \\ &= P(y = 1|\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 0] + P(y = 0|\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 1] \end{aligned}$$

Ex: binary classification

Expected conditional risk of a single data point x

$$\begin{aligned} R(f, \boldsymbol{x}) &= \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y) \\ &= P(y = 1|\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 0] + P(y = 0|\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 1] \end{aligned}$$

Let $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have

$$\begin{aligned} R(f, \boldsymbol{x}) &= \eta(\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 0] + (1 - \eta(\boldsymbol{x})) \mathbb{I}[f(\boldsymbol{x}) = 1] \\ &= 1 - \underbrace{\{\eta(\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x})) \mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}} \end{aligned}$$

Ex: binary classification

Expected conditional risk of a single data point x

$$\begin{aligned} R(f, \boldsymbol{x}) &= \mathbb{E}_{y \sim p(y|\boldsymbol{x})} L(f(\boldsymbol{x}), y) \\ &= P(y = 1|\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 0] + P(y = 0|\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 1] \end{aligned}$$

Let $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have

$$\begin{aligned} R(f, \boldsymbol{x}) &= \eta(\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 0] + (1 - \eta(\boldsymbol{x})) \mathbb{I}[f(\boldsymbol{x}) = 1] \\ &= 1 - \underbrace{\{\eta(\boldsymbol{x}) \mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x})) \mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}} \end{aligned}$$

Exercise: please verify the last equality.

Bayes optimal classifier

Imagine we had access to posterior probability $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$

$$f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad ??? \\ 0 & \text{if} \quad ??? \end{cases}$$

Bayes optimal classifier

Imagine we had access to posterior probability $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$

$$f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \eta(\boldsymbol{x}) \geq 1/2 \\ 0 & \text{if } \eta(\boldsymbol{x}) < 1/2 \end{cases} \quad \text{equivalently } f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } p(y = 1|\boldsymbol{x}) \geq p(y = 0|\boldsymbol{x}) \\ 0 & \text{if } p(y = 1|\boldsymbol{x}) < p(y = 0|\boldsymbol{x}) \end{cases}$$

Bayes optimal classifier

Imagine we had access to posterior probability $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$

$$f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \eta(\boldsymbol{x}) \geq 1/2 \\ 0 & \text{if } \eta(\boldsymbol{x}) < 1/2 \end{cases} \quad \text{equivalently } f^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if } p(y = 1|\boldsymbol{x}) \geq p(y = 0|\boldsymbol{x}) \\ 0 & \text{if } p(y = 1|\boldsymbol{x}) < p(y = 0|\boldsymbol{x}) \end{cases}$$

Theorem

For any labeling function $f(\cdot)$, $R(f^*, \boldsymbol{x}) \leq R(f, \boldsymbol{x})$. *Similarly,* $R(f^*) \leq R(f)$. *Namely,* $f^*(\cdot)$ *is optimal.*

Proof

Definition of expected conditional risk:

$$R(f, \boldsymbol{x}) = 1 - \underbrace{\{\eta(\boldsymbol{x})\mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}}$$

$$R(f^*, \boldsymbol{x}) = 1 - \{\eta(\boldsymbol{x})\mathbb{I}[f^*(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f^*(\boldsymbol{x}) = 0]\}$$

Since $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have:

Proof

Definition of expected conditional risk:

$$R(f, \boldsymbol{x}) = 1 - \underbrace{\{\eta(\boldsymbol{x})\mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}}$$

$$R(f^*, \boldsymbol{x}) = 1 - \{\eta(\boldsymbol{x})\mathbb{I}[f^*(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f^*(\boldsymbol{x}) = 0]\}$$

Since $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have:

$$\begin{aligned} R(f, \boldsymbol{x}) - R(f^*, \boldsymbol{x}) = \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\ + (1 - \eta(\boldsymbol{x})) \{\mathbb{I}[f^*(\boldsymbol{x}) = 0] - \mathbb{I}[f(\boldsymbol{x}) = 0]\} \end{aligned}$$

Proof

Definition of expected conditional risk:

$$R(f, \boldsymbol{x}) = 1 - \underbrace{\{\eta(\boldsymbol{x})\mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}}$$

$$R(f^*, \boldsymbol{x}) = 1 - \{\eta(\boldsymbol{x})\mathbb{I}[f^*(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f^*(\boldsymbol{x}) = 0]\}$$

Since $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have:

$$\begin{aligned} R(f, \boldsymbol{x}) - R(f^*, \boldsymbol{x}) &= \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\ &\quad + (1 - \eta(\boldsymbol{x})) \{\mathbb{I}[f^*(\boldsymbol{x}) = 0] - \mathbb{I}[f(\boldsymbol{x}) = 0]\} \\ &= \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\ &\quad + (1 - \eta(\boldsymbol{x})) \left\{\mathbb{I}[f(\boldsymbol{x}) = 1] - \mathbb{I}[f^*(\boldsymbol{x}) = 1]\right) \} \end{aligned}$$

Proof

Definition of expected conditional risk:

$$R(f, \boldsymbol{x}) = 1 - \underbrace{\{\eta(\boldsymbol{x})\mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}}$$

$$R(f^*, \boldsymbol{x}) = 1 - \{\eta(\boldsymbol{x})\mathbb{I}[f^*(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f^*(\boldsymbol{x}) = 0]\}$$

Since $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have:

$$\begin{aligned}
R(f, \boldsymbol{x}) - R(f^*, \boldsymbol{x}) &= \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\
&\quad + (1 - \eta(\boldsymbol{x})) \{\mathbb{I}[f^*(\boldsymbol{x}) = 0] - \mathbb{I}[f(\boldsymbol{x}) = 0]\} \\
&= \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\
&\quad + (1 - \eta(\boldsymbol{x})) \left\{\mathbb{I}[f(\boldsymbol{x}) = 1] - \mathbb{I}[f^*(\boldsymbol{x}) = 1]\right\} \\
&= (2\eta(\boldsymbol{x}) - 1) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\}
\end{aligned}$$

Proof

Definition of expected conditional risk:

$$R(f, \boldsymbol{x}) = 1 - \underbrace{\{\eta(\boldsymbol{x})\mathbb{I}[f(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f(\boldsymbol{x}) = 0]\}}_{\text{expected conditional accuracy}}$$

$$R(f^*, \boldsymbol{x}) = 1 - \{\eta(\boldsymbol{x})\mathbb{I}[f^*(\boldsymbol{x}) = 1] + (1 - \eta(\boldsymbol{x}))\mathbb{I}[f^*(\boldsymbol{x}) = 0]\}$$

Since $\eta(\boldsymbol{x}) = P(y = 1|\boldsymbol{x})$, we have:

$$\begin{aligned}
R(f, \boldsymbol{x}) - R(f^*, \boldsymbol{x}) &= \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\
&\quad + (1 - \eta(\boldsymbol{x})) \{\mathbb{I}[f^*(\boldsymbol{x}) = 0] - \mathbb{I}[f(\boldsymbol{x}) = 0]\} \\
&= \eta(\boldsymbol{x}) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\
&\quad + (1 - \eta(\boldsymbol{x})) \left\{\mathbb{I}[f(\boldsymbol{x}) = 1] - \mathbb{I}[f^*(\boldsymbol{x}) = 1]\right\} \\
&= (2\eta(\boldsymbol{x}) - 1) \{\mathbb{I}[f^*(\boldsymbol{x}) = 1] - \mathbb{I}[f(\boldsymbol{x}) = 1]\} \\
&\geq 0
\end{aligned}$$

Bayes optimal classifier in general form

For multi-class classification problem

$$f^*(\boldsymbol{x}) = \arg\max_{c \in [\mathsf{C}]} p(y = c|\boldsymbol{x})$$

when $\mathsf{C} = 2$, this reduces to detecting whether or not $\eta(\boldsymbol{x}) = p(y = 1|\boldsymbol{x})$ is greater than $1/2$.

Bayes optimal classifier in general form

For multi-class classification problem

$$f^*(\boldsymbol{x}) = \arg\max_{c \in [\mathsf{C}]} p(y = c|\boldsymbol{x})$$

when $\mathsf{C} = 2$, this reduces to detecting whether or not $\eta(\boldsymbol{x}) = p(y = 1|\boldsymbol{x})$ is greater than $1/2$.

Remarks

- The Bayes optimal classifier is generally not computable as it assumes the knowledge of $p(\boldsymbol{x}, y)$ or $p(y|\boldsymbol{x})$.
- However, it is useful as a conceptual tool to formalize how well a classifier can do *without* knowing the joint distribution.

Comparing NNC to Bayes optimal classifier

How well does our NNC do?

Theorem (Cover-Hart Inequality)

For the NNC rule f^{NNC} for binary classification, we have,

$$R(f^*) \leq R(f^{\mathrm{NNC}}) \leq 2R(f^*)(1 - R(f^*)) \leq 2R(f^*)$$

Namely, the expected risk by the classifier is at worst twice that of the Bayes optimal classifier.

In short, NNC is doing a reasonable thing

Outline

1. Administration
2. First learning algorithm: Nearest neighbor classifier
3. Deeper understanding of NNC
4. Some practical aspects of NNC
   - How to tune to get the best out of it?
   - Preprocessing data
5. What we have learned

Hyperparameters in NNC

Two practical issues about NNC

- Choosing K, i.e., the number of nearest neighbors (default is 1)
- Choosing the right distance measure (default is Euclidean distance), for example, from the following generalized distance measure

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

for $p \geq 1$.

Those are not specified by the algorithm itself — resolving them requires empirical studies and are task/dataset-specific.

Tuning by using 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

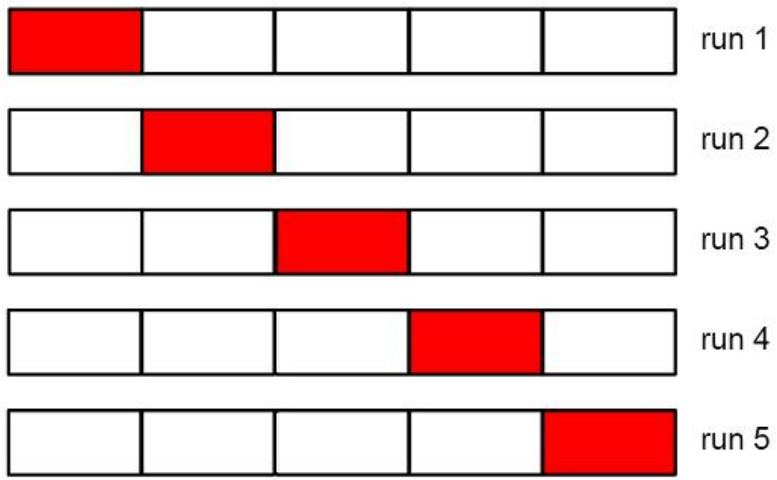
- For each possible value of the hyperparameter (say $K = 1, 3, \cdots, 100$)
  - Train a model using $\mathcal{D}^{\text{TRAIN}}$
  - Evaluate the performance of the model on $\mathcal{D}^{\text{VAL}}$
- Choose the model with the best performance on $\mathcal{D}^{\text{VAL}}$
- Evaluate the model on $\mathcal{D}^{\text{TEST}}$

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 (based on average, variance, etc.)

$S = 5$: 5-fold cross validation

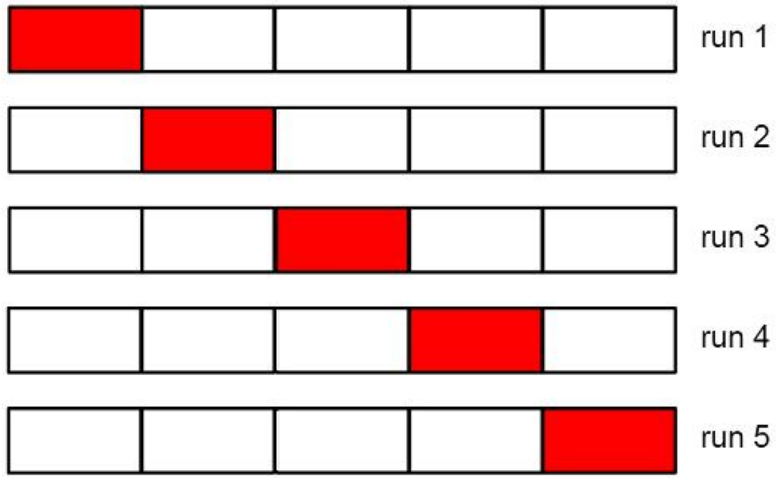


Cross-validation

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 (based on average, variance, etc.)

$S = 5$: 5-fold cross validation



Special case: when S = N, this will be leave-one-out.

Yet, another practical issue with NNC

Distances depend on units of the features!

Preprocess data

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

- Compute the means and standard deviations in each feature

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

- 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

Outline

1. Administration
2. First learning algorithm: Nearest neighbor classifier
3. Deeper understanding of NNC
4. Some practical aspects of NNC
5. What we have learned

Summary so far

- Described a simple learning algorithm
  - Used intensively in practical applications — you will get a taste of it in your homework
  - Discussed a few practical aspects, such as tuning hyperparameters, with (cross)validation
- Briefly studied its theoretical properties
  - Concepts: loss function, risks, Bayes optimal
  - Theoretical guarantees: explaining why NNC would work

Administration Summary

- Course is full, but I'm confident we can accommodate all students
  - Math quiz / HW1 will give you a sense of mathematical rigor of course
  - If you're not registered, be patient!
  - I am not giving out PTEs until after HW1
- HW1 is online
  - Due next Wednesday at the beginning of class
  - Submission details are included in the assignment
  - Join Piazza if you haven't already