

Lecture 14

Regression on indicators, the Bayes classifier, k -NN

COURSE MATH 637: Mathematical Techniques in Data Science

WEBSITE <https://jacobcalvert.com/teaching/math637fall2026/>

INSTRUCTOR Jacob Calvert (calvert@udel.edu)

OFFICE HOURS MW 2:45pm–3:45pm, Ewing Hall 509

Announcements & Reminders

- Midterm will be on **Monday, October 19th** during class.
 - Quiz and discussion questions are representative.
 - There will be no standalone quiz on classification basics.
 - We will have at least one in-class review session.
- After the midterm, our focus will shift to project work.
- Homework 2 and quiz 3 grades and solutions are now available.

Plan for today

- From regression to classification 10 min
- Estimate, then decide 11 min
- Where the linear fit fails 7 min
- k -nearest neighbors 14 min
- The curse of dimensionality 6 min

Definition: Classification

Definition (Classification)

Supervised prediction of a **categorical label**: Y takes one of K values, which we name $1, \dots, K$.

- will this customer default? ($K = 2$, from balance and income)

Definition: Classification

Definition (Classification)

Supervised prediction of a **categorical label**: Y takes one of K values, which we name $1, \dots, K$.

- will this customer default? ($K = 2$, from balance and income)
- which digit is this? ($K = 10$, from 784 pixel intensities)

Definition: Classification

Definition (Classification)

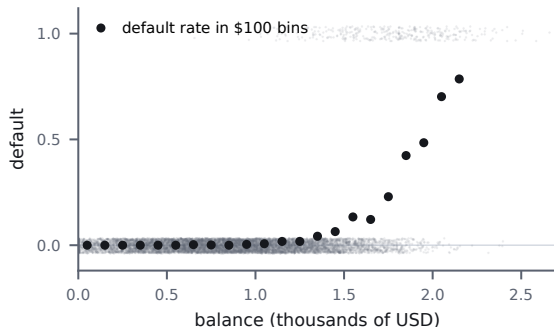
Supervised prediction of a **categorical label**: Y takes one of K values, which we name $1, \dots, K$.

- will this customer default? ($K = 2$, from balance and income)
- which digit is this? ($K = 10$, from 784 pixel intensities)
- which cultivar produced this wine? ($K = 3$, from chemistry)

Example: Credit card default

$n = 10,000$ customers:

- x_i : credit card **balance** (\$1k)
- $y_i = 1$ if **default**, else 0
- 333 defaults (3.33%)
- each dot = default rate among customers in bin



James, Witten, Hastie, Tibshirani and Taylor, *An Introduction to Statistical Learning with Applications in Python* (2023)

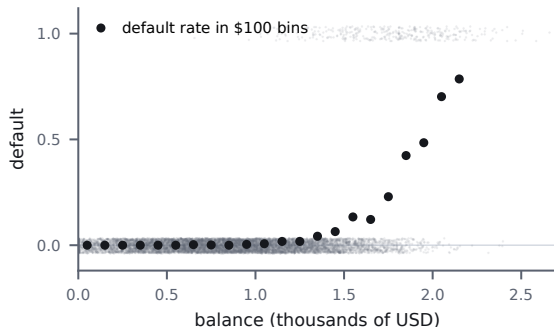
Example: Credit card default

$n = 10,000$ customers:

- x_i : credit card **balance** (\$1k)
- $y_i = 1$ if **default**, else 0
- 333 defaults (3.33%)
- each dot = default rate among customers in bin

Question

What does the dot estimate?



James, Witten, Hastie, Tibshirani and Taylor, *An Introduction to Statistical Learning with Applications in Python* (2023)

Idea: Regress the label on balance

Recall: Under squared loss, the best prediction of Y from $X = x$ is the average response

$$f(x) = \mathbb{E}[Y \mid X = x].$$

Idea: Regress the label on balance

Recall: Under squared loss, the best prediction of Y from $X = x$ is the average response

$$f(x) = \mathbb{E}[Y \mid X = x].$$

Least squares on $y_i \in \{0, 1\}$, exactly as before:

$$\hat{f}(x) = -0.075 + 0.130 x.$$

Idea: Regress the label on balance

Recall: Under squared loss, the best prediction of Y from $X = x$ is the average response

$$f(x) = \mathbb{E}[Y \mid X = x].$$

Least squares on $y_i \in \{0, 1\}$, exactly as before:

$$\hat{f}(x) = -0.075 + 0.130 x.$$

For a response that is 0 or 1, the average response is a probability:

$$\mathbb{E}[Y \mid X = x] = \mathbb{P}(Y = 1 \mid X = x).$$

Idea: Regress the label on balance

Recall: Under squared loss, the best prediction of Y from $X = x$ is the average response

$$f(x) = \mathbb{E}[Y \mid X = x].$$

Least squares on $y_i \in \{0, 1\}$, exactly as before:

$$\hat{f}(x) = -0.075 + 0.130 x.$$

For a response that is 0 or 1, the average response is a probability:

$$\mathbb{E}[Y \mid X = x] = \mathbb{P}(Y = 1 \mid X = x).$$

Idea: Least squares on binary label estimates probability of “label = 1.”

Definition: Class probabilities

Definition

For $Y \in \{1, \dots, K\}$, the **indicators** are $Y_j = 1\{Y = j\}$, and the **class probabilities** are

$$\eta_j(x) = \mathbb{P}(Y = j \mid X = x) = \mathbb{E}[Y_j \mid X = x].$$

- Regress each indicator Y_j on X : $\hat{f}_j(x)$ estimates $\eta_j(x)$

Definition: Class probabilities

Definition

For $Y \in \{1, \dots, K\}$, the **indicators** are $Y_j = 1\{Y = j\}$, and the **class probabilities** are

$$\eta_j(x) = \mathbb{P}(Y = j \mid X = x) = \mathbb{E}[Y_j \mid X = x].$$

- Regress each indicator Y_j on X : $\hat{f}_j(x)$ estimates $\eta_j(x)$
- Why not regress the label itself? **Regressing a digit's value would, e.g., treat a 7 as closer to an 8 than to a 1.**

Definition: Class probabilities

Definition

For $Y \in \{1, \dots, K\}$, the **indicators** are $Y_j = 1\{Y = j\}$, and the **class probabilities** are

$$\eta_j(x) = \mathbb{P}(Y = j \mid X = x) = \mathbb{E}[Y_j \mid X = x].$$

- Regress each indicator Y_j on X : $\hat{f}_j(x)$ estimates $\eta_j(x)$
- Why not regress the label itself? **Regressing a digit's value would, e.g., treat a 7 as closer to an 8 than to a 1.**

Definition: Class probabilities

Definition

For $Y \in \{1, \dots, K\}$, the **indicators** are $Y_j = 1\{Y = j\}$, and the **class probabilities** are

$$\eta_j(x) = \mathbb{P}(Y = j \mid X = x) = \mathbb{E}[Y_j \mid X = x].$$

- Regress each indicator Y_j on X : $\hat{f}_j(x)$ estimates $\eta_j(x)$
- Why not regress the label itself? **Regressing a digit's value would, e.g., treat a 7 as closer to an 8 than to a 1.**

Check

Assume $[1 \ X]$ has full col. rank. Why is $\hat{f}_1(x) + \dots + \hat{f}_K(x) = 1$ at every x ?

Definition: Class probabilities

Check

Assume $[1 \ X]$ has full col. rank. Why is $\hat{f}_1(x) + \dots + \hat{f}_K(x) = 1$ at every x ?

- each row of $[y_1 \ \dots \ y_K]$ has exactly one 1, so $y_1 + \dots + y_K = \mathbf{1}$

Definition: Class probabilities

Check

Assume $[1 \ X]$ has full col. rank. Why is $\hat{f}_1(x) + \dots + \hat{f}_K(x) = 1$ at every x ?

- each row of $[y_1 \ \dots \ y_K]$ has exactly one 1, so $y_1 + \dots + y_K = 1$
- least squares is linear in the response, and regressing 1 on $[1 \ X]$ gives intercept 1 and coefficients 0

Definition: Class probabilities

Check

Assume $[1 \ X]$ has full col. rank. Why is $\hat{f}_1(x) + \dots + \hat{f}_K(x) = 1$ at every x ?

- each row of $[y_1 \ \dots \ y_K]$ has exactly one 1, so $y_1 + \dots + y_K = 1$
- least squares is linear in the response, and regressing 1 on $[1 \ X]$ gives intercept 1 and coefficients 0
- full col. rank makes these coefficients unique, so the K fitted coefficient vectors sum to $(1, 0, \dots, 0)$: the fits sum to 1 at every x

Two steps: **estimate**, then **decide**

1. **estimate** the class probabilities: $\hat{f}_j(x)$ for $\eta_j(x)$

Two steps: **estimate**, then **decide**

1. **estimate** the class probabilities: $\hat{f}_j(x)$ for $\eta_j(x)$
2. **decide**: use these estimates to predict a label, e.g., $\arg \max_j \hat{f}_j(x)$

Two steps: **estimate**, then **decide**

1. **estimate** the class probabilities: $\hat{f}_j(x)$ for $\eta_j(x)$
2. **decide**: use these estimates to predict a label, e.g., $\arg \max_j \hat{f}_j(x)$

Two steps: **estimate**, then **decide**

1. **estimate** the class probabilities: $\hat{f}_j(x)$ for $\eta_j(x)$
2. **decide**: use these estimates to predict a label, e.g., $\arg \max_j \hat{f}_j(x)$

Question

Suppose we knew $\eta_1(x), \dots, \eta_K(x)$ exactly. How would you decide?

Plan for today

- From regression to classification 10 min
- **Estimate, then decide** 11 min
- Where the linear fit fails 7 min
- k -nearest neighbors 14 min
- The curse of dimensionality 6 min

Definition: Classifier and risk

Definition

A **classifier** is a function g from inputs to labels. Its **risk** is the probability of error on a new example:

$$R(g) = \mathbb{P}(g(X) \neq Y).$$

- the loss is $1\{g(X) \neq Y\}$: 1 for a mistake, 0 otherwise

Definition: Classifier and risk

Definition

A **classifier** is a function g from inputs to labels. Its **risk** is the probability of error on a new example:

$$R(g) = \mathbb{P}(g(X) \neq Y).$$

- the loss is $1\{g(X) \neq Y\}$: 1 for a mistake, 0 otherwise
- $R(g) = 1 - \text{accuracy on new data}$

Theorem: The Bayes classifier

Theorem (Bayes classifier)

The classifier that picks the largest **true** class probability,

$$g^*(x) = \arg \max_j \eta_j(x),$$

has the smallest risk among all classifiers: $R(g^*) \leq R(g)$ for every g .

Proof. Try it! Fix $X = x$ first.



Theorem: The Bayes classifier

Theorem (Bayes classifier)

The classifier that picks the largest **true** class probability,

$$g^*(x) = \arg \max_j \eta_j(x),$$

has the smallest risk among all classifiers: $R(g^*) \leq R(g)$ for every g .

Proof.

$$\begin{aligned} \mathbb{P}(g(X) \neq Y \mid X = x) &= 1 - \eta_{g(x)}(x) \\ &\geq 1 - \max_j \eta_j(x) = \mathbb{P}(g^*(X) \neq Y \mid X = x). \end{aligned}$$

Average both sides over X .



Definition: Bayes error

Definition

The **Bayes error** is the risk of the Bayes classifier:

$$R^* = R(g^*) = \mathbb{E} \left[1 - \max_j \eta_j(X) \right].$$

- no classifier using the same features has less risk, regardless of how much data we have

Definition: Bayes error

Definition

The **Bayes error** is the risk of the Bayes classifier:

$$R^* = R(g^*) = \mathbb{E} \left[1 - \max_j \eta_j(X) \right].$$

- no classifier using the same features has less risk, regardless of how much data we have
- it plays a role analogous to that of noise in the bias–variance decomposition

Definition: Bayes error

Definition

The **Bayes error** is the risk of the Bayes classifier:

$$R^* = R(g^*) = \mathbb{E} \left[1 - \max_j \eta_j(X) \right].$$

- no classifier using the same features has less risk, regardless of how much data we have
- it plays a role analogous to that of noise in the bias–variance decomposition
- adding informative features can lower it

Definition: Bayes error

Definition

The **Bayes error** is the risk of the Bayes classifier:

$$R^* = R(g^*) = \mathbb{E} \left[1 - \max_j \eta_j(X) \right].$$

- no classifier using the same features has less risk, regardless of how much data we have
- it plays a role analogous to that of noise in the bias–variance decomposition
- adding informative features can lower it
- “Bayes” refers to Bayes’ rule for the label of a new example

Example: Two Gaussian classes

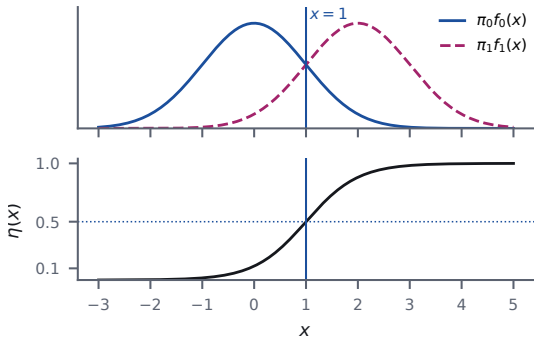
Example

$\mathbb{P}(Y = 0) = \mathbb{P}(Y = 1) = \frac{1}{2}$ and

$$X \mid Y = 0 \sim \mathcal{N}(0, 1),$$

$$X \mid Y = 1 \sim \mathcal{N}(2, 1).$$

Find g^* and R^* .

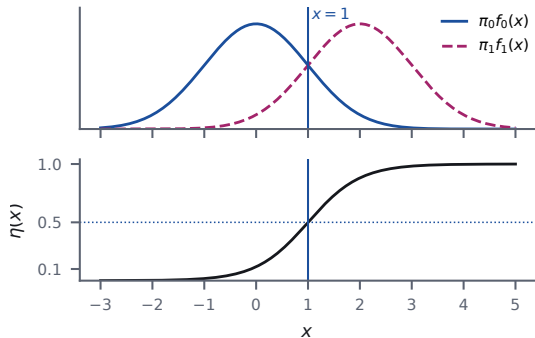


Recall that g^* is the “best guess” and R^* the associated error:

$$g^*(x) = \arg \max_j \eta_j(x) \quad \text{and} \quad R^* = \mathbb{E}[1 - \max_j \eta_j(X)].$$

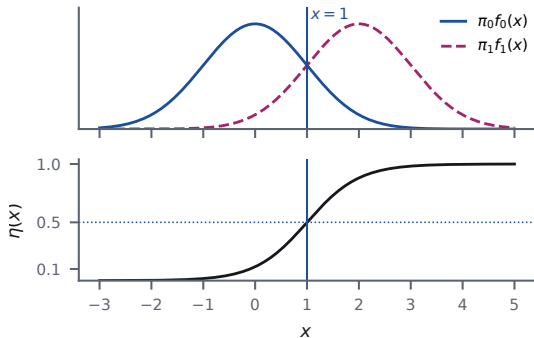
Example: Two Gaussian classes

- By symmetry, $g^*(x) = 1$ when $x \geq 1$ and zero otherwise



Example: Two Gaussian classes

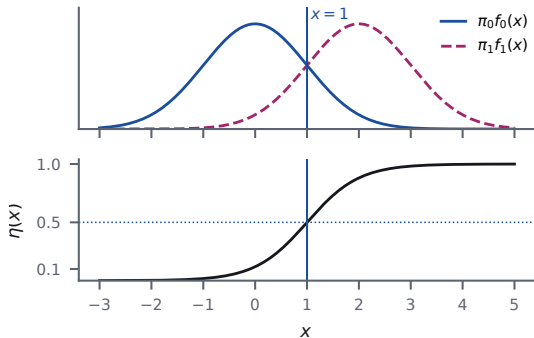
- By symmetry, $g^*(x) = 1$ when $x \geq 1$ and zero otherwise
- R^* is sum of $\mathbb{P}(X > 1, Y = 0)$ and $\mathbb{P}(X < 1, Y = 1)$



Example: Two Gaussian classes

- By symmetry, $g^*(x) = 1$ when $x \geq 1$ and zero otherwise
- R^* is sum of $\mathbb{P}(X > 1, Y = 0)$ and $\mathbb{P}(X < 1, Y = 1)$
- Since $Y = 0$ and $Y = 1$ equally likely, R^* is

$$\frac{1}{2}(1 - \Phi(1)) + \frac{1}{2}\Phi(-1) \approx 0.159$$



Not all mistakes **cost** the same

Example (Default data)

Trivially predicting that no one defaults is 96.7% accurate, but it misses every default.

Not all mistakes **cost** the same

Problem

Suppose a false alarm costs 1 while a missed default costs 9. If a customer's probability of default is $\eta(x)$, when should we flag the customer?

Not all mistakes **cost** the same

Problem

Suppose a false alarm costs 1 while a missed default costs 9. If a customer's probability of default is $\eta(x)$, when should we flag the customer?

	defaults (prob. $\eta(x)$)	does not default (prob. $1 - \eta(x)$)	expected cost
flag	0	1 (false alarm)	$(1 - \eta(x)) \cdot 1$
pass	9 (missed default)	0	$\eta(x) \cdot 9$

Not all mistakes **cost** the same

Problem

Suppose a false alarm costs 1 while a missed default costs 9. If a customer's probability of default is $\eta(x)$, when should we flag the customer?

	defaults	does not default	expected cost
flag	0	1 (false alarm)	$(1 - \eta(x)) \cdot 1$
pass	9 (missed default)	0	$\eta(x) \cdot 9$

- flag when it costs less on average: $1 - \eta(x) \leq 9 \eta(x)$, i.e., $\eta(x) \geq \frac{1}{10}$

Not all mistakes **cost** the same

Problem

Suppose a false alarm costs 1 while a missed default costs 9. If a customer's probability of default is $\eta(x)$, when should we flag the customer?

	defaults	does not default	expected cost
flag	0	1 (false alarm)	$(1 - \eta(x)) \cdot 1$
pass	9 (missed default)	0	$\eta(x) \cdot 9$

- flag when it costs less on average: $1 - \eta(x) \leq 9 \eta(x)$, i.e., $\eta(x) \geq \frac{1}{10}$
- in general, flag when $\eta(x) \geq \frac{c_{FP}}{c_{FP} + c_{FN}}$; equal costs give Bayes classifier

Not all mistakes **cost** the same

Problem

Suppose a false alarm costs 1 while a missed default costs 9. If a customer's probability of default is $\eta(x)$, when should we flag the customer?

	defaults	does not default	expected cost
flag	0	1 (false alarm)	$(1 - \eta(x)) \cdot 1$
pass	9 (missed default)	0	$\eta(x) \cdot 9$

- flag when it costs less on average: $1 - \eta(x) \leq 9 \eta(x)$, i.e., $\eta(x) \geq \frac{1}{10}$
- in general, flag when $\eta(x) \geq \frac{c_{FP}}{c_{FP} + c_{FN}}$; equal costs give Bayes classifier

Not all mistakes **cost** the same

Problem

Suppose a false alarm costs 1 while a missed default costs 9. If a customer's probability of default is $\eta(x)$, when should we flag the customer?

	defaults	does not default	expected cost
flag	0	1 (false alarm)	$(1 - \eta(x)) \cdot 1$
pass	9 (missed default)	0	$\eta(x) \cdot 9$

- flag when it costs less on average: $1 - \eta(x) \leq 9 \eta(x)$, i.e., $\eta(x) \geq \frac{1}{10}$
- in general, flag when $\eta(x) \geq \frac{c_{FP}}{c_{FP} + c_{FN}}$; equal costs give Bayes classifier

Idea: Same probabilities, different decision. The estimate doesn't depend on the costs but the decision does.

Plan for today

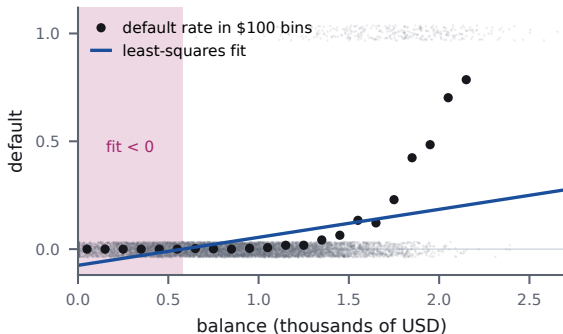
- From regression to classification 10 min
- Estimate, then decide 11 min
- Where the linear fit fails 7 min
- k -nearest neighbors 14 min
- The curse of dimensionality 6 min

Problem 1: The fitted values leave $[0, 1]$

Example. Recall that least squares on the default data gives

$$\hat{f}(x) = -0.075 + 0.130x.$$

- the fit predicts a negative value for 3,123 customers

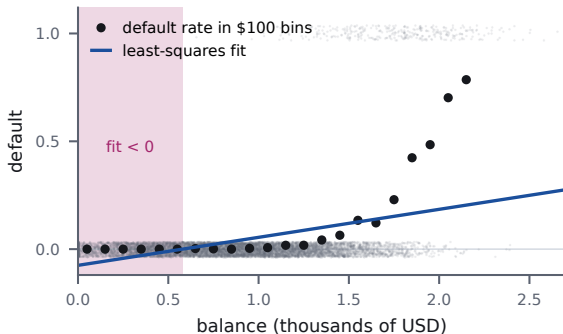


Problem 1: The fitted values leave $[0, 1]$

Example. Recall that least squares on the default data gives

$$\hat{f}(x) = -0.075 + 0.130x.$$

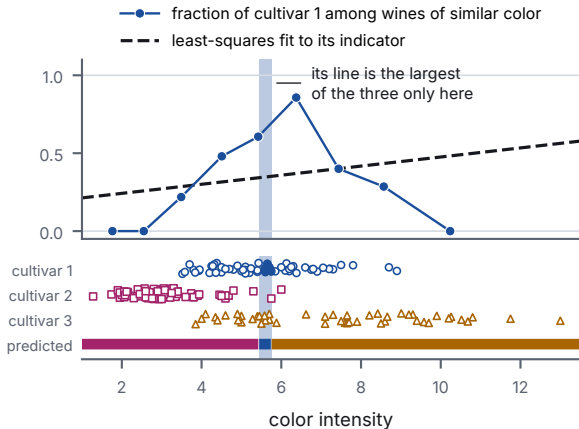
- the fit predicts a negative value for **3,123** customers
- we can't interpret this as a probability



Problem 2: Wine cultivars from color

Example. The fraction of cultivar 1 among wines with color intensity x depends nonlinearly on x .

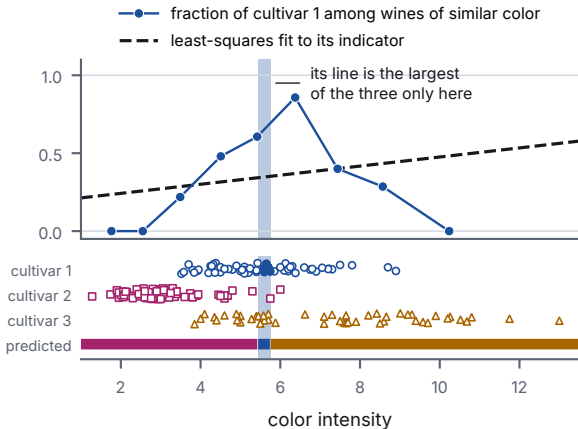
- Least squares on the indicators, as before



Problem 2: Wine cultivars from color

Example. The fraction of cultivar 1 among wines with color intensity x depends nonlinearly on x .

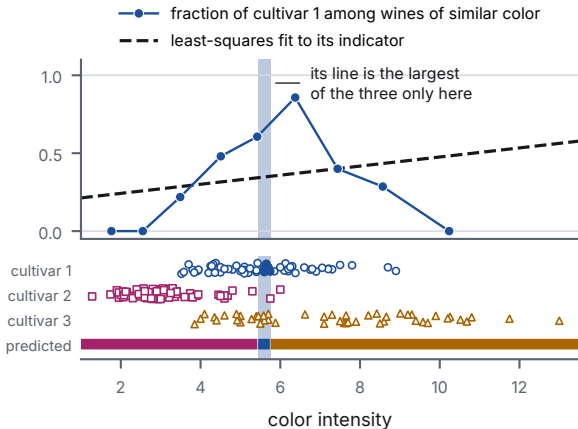
- Least squares on the indicators, as before
- Other lines (not shown) exceed cultivar 1 fit outside band



Problem 2: Wine cultivars from color

Example. The fraction of cultivar 1 among wines with color intensity x depends nonlinearly on x .

- Least squares on the indicators, as before
- Other lines (not shown) exceed cultivar 1 fit outside band
- Only 7/59 cultivar 1 wines classified correctly



Two **distinct** problems with the linear fit

	Problem 1 (credit)	Problem 2 (wine)
true probability	only rises	rises, then falls
the fit	rises too	nearly flat
what fails	its range : leaves $[0, 1]$	its shape : cannot fall
symptom	3,123 negative fits	7 of 59 cultivar 1 correct

Two **distinct** problems with the linear fit

	Problem 1 (credit)	Problem 2 (wine)
true probability	only rises	rises, then falls
the fit	rises too	nearly flat
what fails	its range : leaves $[0, 1]$	its shape : cannot fall
symptom	3,123 negative fits	7 of 59 cultivar 1 correct

Key idea: Clipping fitted values to $[0, 1]$ fixes Problem 1, not Problem 2.

Aside: Squared loss is not the problem

Check

For binary response and any number q ,

$$\mathbb{E}[(Y - q)^2 \mid X = x] = \eta(x)(1 - \eta(x)) + (q - \eta(x))^2,$$

which is **minimized** at $q = \eta(x)$.

Aside: Squared loss is not the problem

Check

For binary response and any number q ,

$$\mathbb{E}[(Y - q)^2 \mid X = x] = \eta(x)(1 - \eta(x)) + (q - \eta(x))^2,$$

which is **minimized** at $q = \eta(x)$.

- this means least squares on indicators favors the true $\eta_j(x)$

Aside: Squared loss is not the problem

Check

For binary response and any number q ,

$$\mathbb{E}[(Y - q)^2 \mid X = x] = \eta(x)(1 - \eta(x)) + (q - \eta(x))^2,$$

which is **minimized** at $q = \eta(x)$.

- this means least squares on indicators favors the true $\eta_j(x)$
- problems on previous slide come from forcing η_j to be linear in x

This week: Three ways to estimate η_j

method	main idea
k -NN (today)	average the class indicators of nearby examples
generative classifiers	model each class's x , then use Bayes' rule
logistic regression	model linear log-odds

Plan for today

- From regression to classification 10 min
- Estimate, then decide 11 min
- Where the linear fit fails 7 min
- *k*-nearest neighbors 14 min
- The curse of dimensionality 6 min

Definition: k -nearest neighbors

Definition (k -NN)

Let $N_k(x)$ = set of indices of the k training inputs closest to x . Estimate

$$\hat{\eta}_j(x) = \frac{1}{k} \sum_{i \in N_k(x)} \mathbf{1}\{y_i = j\}, \quad j = 1, \dots, K.$$

Predict $\hat{g}(x) = \arg \max_j \hat{\eta}_j(x)$, the most common label among neighbors.

Idea: k -NN is a **local** least-squares fit

Question

What number c minimizes

$$\sum_{i \in N_k(x)} (1\{y_i = j\} - c)^2?$$

Idea: k -NN is a **local** least-squares fit

Theorem

The k -NN estimate $\hat{\eta}_j(x)$ is the number c that minimizes

$$\sum_{i \in N_k(x)} (1\{y_i = j\} - c)^2.$$

Idea: k -NN is a **local** least-squares fit

Theorem

The k -NN estimate $\hat{\eta}_j(x)$ is the number c that minimizes

$$\sum_{i \in N_k(x)} (1\{y_i = j\} - c)^2.$$

Idea: k -NN is least squares on the indicators, with a constant fitted separately near each x instead of one line fitted to all the data.

Idea: k -NN is a **local** least-squares fit

Theorem

The k -NN estimate $\hat{\eta}_j(x)$ is the number c that minimizes

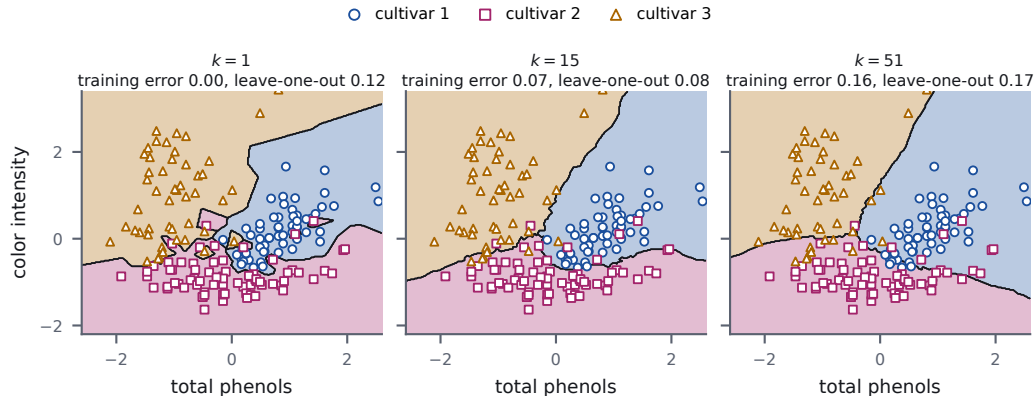
$$\sum_{i \in N_k(x)} (\mathbf{1}\{y_i = j\} - c)^2.$$

Idea: k -NN is least squares on the indicators, with a constant fitted separately near each x instead of one line fitted to all the data.

In other words, it models $\eta_j(x)$ as being *locally* linear, but not necessarily *globally* linear.

Example: k -NN on the wine data

Total phenols and color intensity, both standardized. Shading = predicted cultivar.



k controls **bias** and **variance**

- $k = 1$: zero training error, but boundary can change with every new sample

k controls **bias** and **variance**

- $k = 1$: zero training error, but boundary can change with every new sample
- $k \gg 1$: averages over distant points, gives smooth boundary but can “miss” structure

k controls **bias** and **variance**

- $k = 1$: zero training error, but boundary can change with every new sample
- $k \gg 1$: averages over distant points, gives smooth boundary but can “miss” structure
- k -NN fits about n/k local averages (e.g., $n = 178$, $k = 15$ gives ≈ 12)

k controls **bias** and **variance**

- $k = 1$: zero training error, but boundary can change with every new sample
- $k \gg 1$: averages over distant points, gives smooth boundary but can “miss” structure
- k -NN fits about n/k local averages (e.g., $n = 178$, $k = 15$ gives ≈ 12)

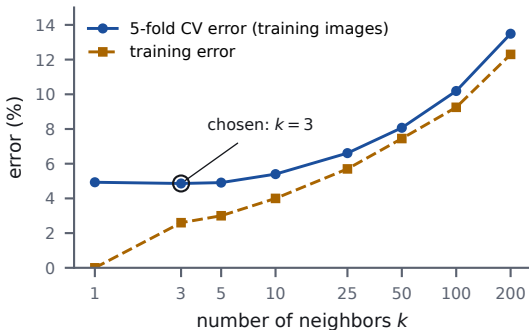
k controls **bias** and **variance**

- $k = 1$: zero training error, but boundary can change with every new sample
- $k \gg 1$: averages over distant points, gives smooth boundary but can “miss” structure
- k -NN fits about n/k local averages (e.g., $n = 178$, $k = 15$ gives ≈ 12)

Idea: Choose k by cross-validation, exactly as we chose ridge penalty λ .

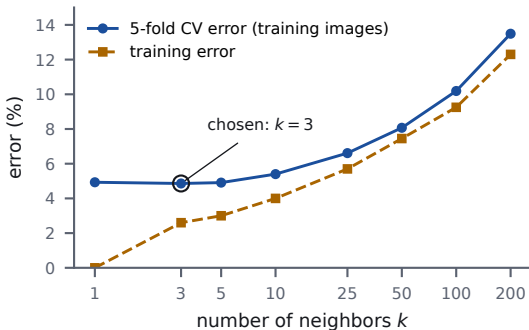
Example: Handwritten digits

- 10,000 training, 2,000 test images
- 28×28 pixels: $p = 784$
- choose k by 5-fold CV on the training images: $k = 3$



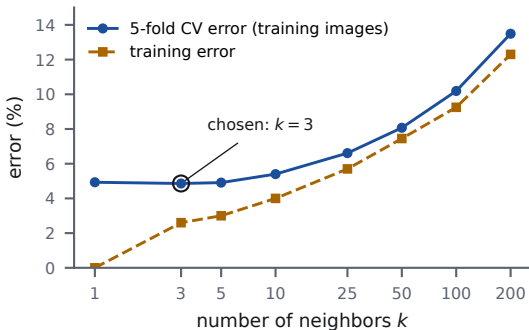
Example: Handwritten digits

- 10,000 training, 2,000 test images
- 28×28 pixels: $p = 784$
- choose k by 5-fold CV on the training images: $k = 3$
- then use the test images once: 3.35% test error



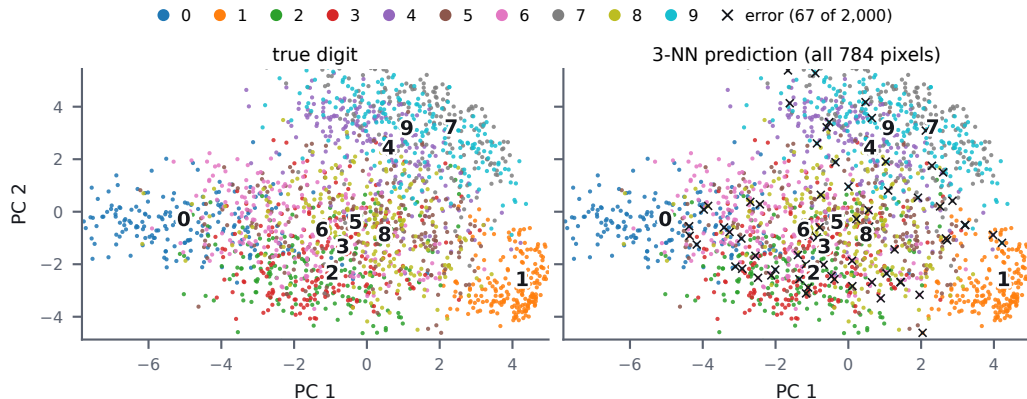
Example: Handwritten digits

- 10,000 training, 2,000 test images
- 28×28 pixels: $p = 784$
- choose k by 5-fold CV on the training images: $k = 3$
- then use the test images once: 3.35% test error
- training error 0 at $k = 1$



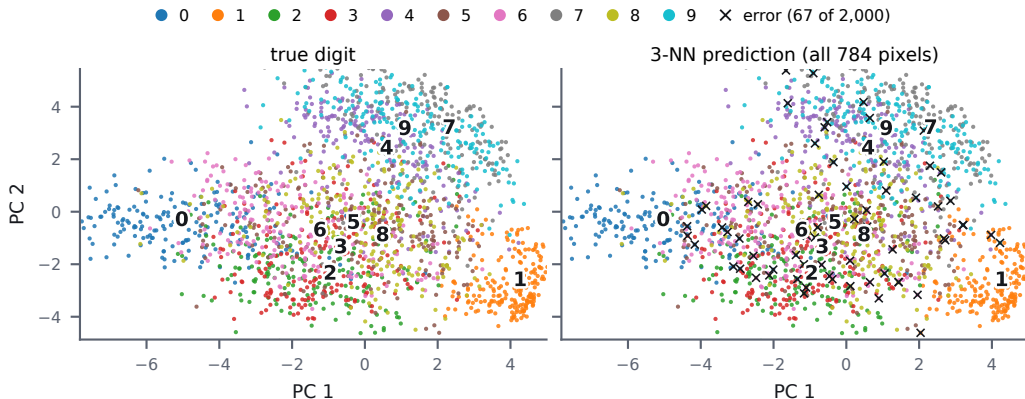
Example: Digits, seen in the first two PCs

Left = test images on the first two PCs. Right = chosen 3-NN using all 784 pixels.



Example: Digits, seen in the first two PCs

Left = test images on the first two PCs. Right = chosen 3-NN using all 784 pixels.



The digits overlap in this picture, yet 3-NN is 96.7% accurate. Why?

Distances depend on **units**

Example (All 13 wine measurements)

- proline is measured in mg/L, with variance 99,167; the next largest variance is 204
 - proline is 99.8% of the total variance, so it dominates squared distances on average
-
- 1-NN, 5-fold CV accuracy: 71.9% on the raw measurements, 95.5% after standardizing

Distances depend on **units**

Example (All 13 wine measurements)

- proline is measured in mg/L, with variance 99,167; the next largest variance is 204
 - proline is 99.8% of the total variance, so it dominates squared distances on average
-
- 1-NN, 5-fold CV accuracy: 71.9% on the raw measurements, 95.5% after standardizing

Distances depend on **units**

Example (All 13 wine measurements)

- proline is measured in mg/L, with variance 99,167; the next largest variance is 204
 - proline is 99.8% of the total variance, so it dominates squared distances on average
-
- 1-NN, 5-fold CV accuracy: 71.9% on the raw measurements, 95.5% after standardizing

Watch out: Fit the standardization on the training folds only, as part of the procedure being validated.

Choosing k in Python

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
pipe = make_pipeline(StandardScaler(),
                     KNeighborsClassifier())
grid = {"kneighborsclassifier__n_neighbors":
        [1, 3, 5, 11, 25, 51]}
search = GridSearchCV(pipe, grid, cv=5)
search.fit(X_train, y_train)
eta_hat = search.predict_proba(X_test)  # fractions
```

- the scaler is refit inside every fold; `predict_proba` returns $\hat{\eta}_j(x)$,
`predict` their argmax

Plan for today

- From regression to classification 10 min
- Estimate, then decide 11 min
- Where the linear fit fails 7 min
- k -nearest neighbors 14 min
- The curse of dimensionality 6 min

How **near** are the nearest neighbors?

Problem

The inputs are uniform on the cube $[0, 1]^p$. How long must the side of a smaller cube be to contain 10% of them?

How **near** are the nearest neighbors?

Problem

The inputs are uniform on the cube $[0, 1]^p$. How long must the side of a smaller cube be to contain 10% of them?

$$s^p = 0.1 \quad \implies \quad s = 0.1^{1/p}$$

p	1	2	10	100
side s	0.10	0.32	0.79	0.98

How **near** are the nearest neighbors?

Problem

The inputs are uniform on the cube $[0, 1]^p$. How long must the side of a smaller cube be to contain 10% of them?

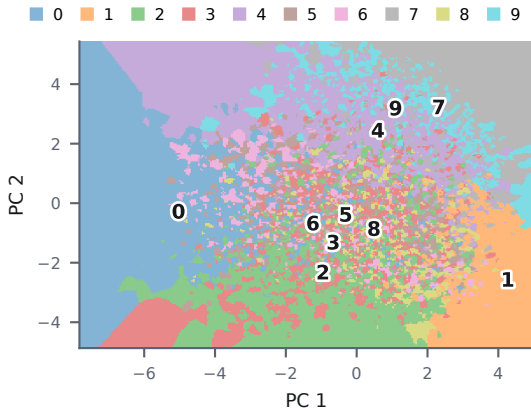
$$s^p = 0.1 \quad \implies \quad s = 0.1^{1/p}$$

p	1	2	10	100
side s	0.10	0.32	0.79	0.98

Key idea: With $p = 10$, a neighborhood holding 10% of the data spans 79% of the range of every coordinate. It is no longer local.

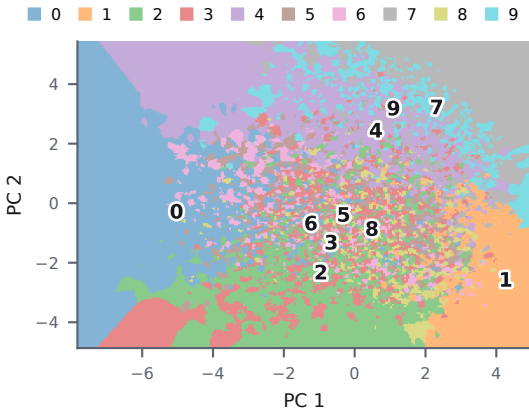
Why does k -NN work on **digits**?

- digits are far from uniform on $[0, 1]^{784}$



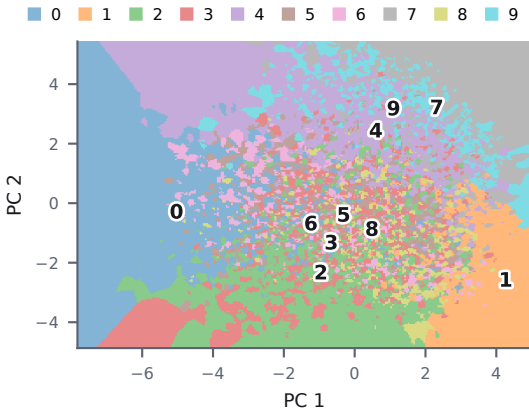
Why does k -NN work on **digits**?

- digits are far from uniform on $[0, 1]^{784}$
- they lie near a set with lower dimension



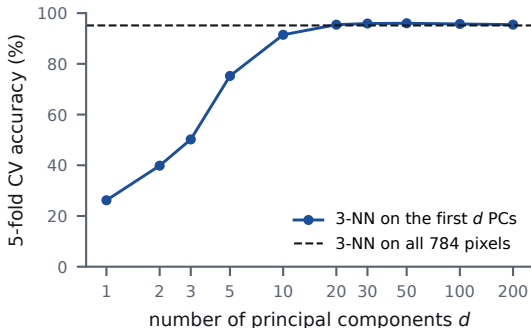
Why does k -NN work on **digits**?

- digits are far from uniform on $[0, 1]^{784}$
- they lie near a set with lower dimension
- but not a 2D one: 3-NN on the first two PCs alone has 5-fold CV accuracy **40%**



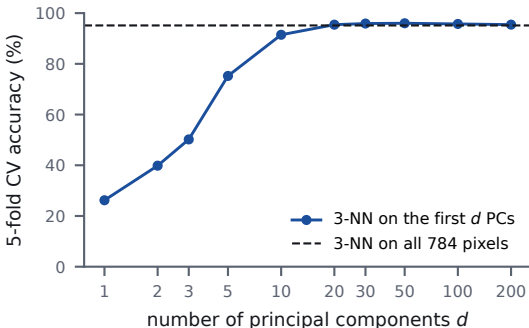
How many PCs does k -NN need?

- 3-NN on the first d PCs, 5-fold CV accuracy: 40% at $d = 2$, 91.5% at $d = 10$, 96.0% at $d = 50$



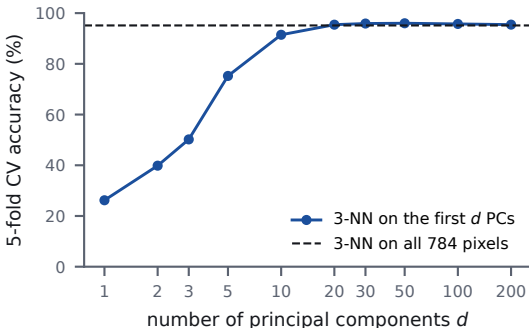
How many PCs does k -NN need?

- 3-NN on the first d PCs, 5-fold CV accuracy: 40% at $d = 2$, 91.5% at $d = 10$, 96.0% at $d = 50$
- CV picks $d = 50$; test accuracy 97.5%, against 96.7% for the chosen $k = 3$ on all 784 pixels



How many PCs does k -NN need?

- 3-NN on the first d PCs, 5-fold CV accuracy: 40% at $d = 2$, 91.5% at $d = 10$, 96.0% at $d = 50$
- CV picks $d = 50$; test accuracy 97.5%, against 96.7% for the chosen $k = 3$ on all 784 pixels
- fewer, better coordinates help a local method



Summary

- least squares on indicators estimates the class probabilities
 $\eta_j(x) = \mathbb{P}(Y = j \mid X = x)$
- estimate, then decide: the Bayes classifier $\arg \max_j \eta_j(x)$ has the smallest risk; unequal costs move the threshold to $c_{\text{FP}} / (c_{\text{FP}} + c_{\text{FN}})$
- a linear estimate of η_j has two distinct problems: its values leave $[0, 1]$, and it cannot rise and fall, which can mask a class
- k -NN estimates η_j by local averages; k trades bias for variance and is chosen by cross-validation
- local averages need the data to have low intrinsic dimension