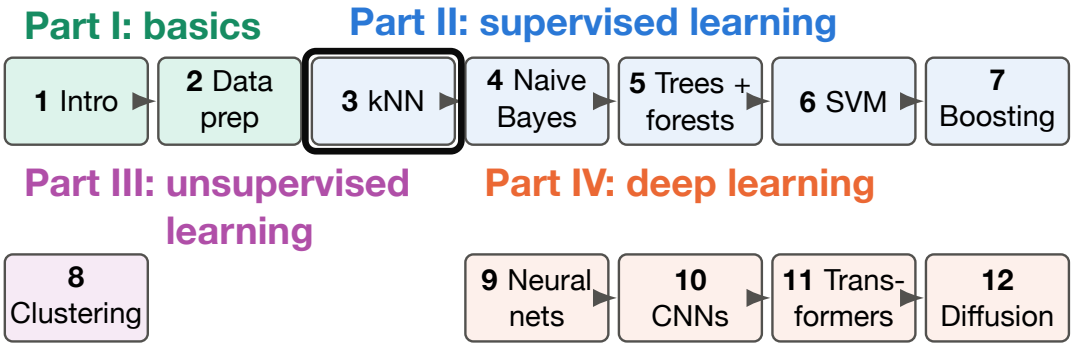
k-nearest neighbors

229352 Lecture 3: k-nearest neighbors

Course map


Part I: basics
Part II: supervised learning
1 Intro
2 Data prep
3 kNN
4 Naive Bayes
5 Trees + forests
6 SVM
7 Boosting
Part III: unsupervised learning
Part IV: deep learning
8 Clustering
9 Neural nets
10 CNNs
11 Trans-formers
12 Diffusion

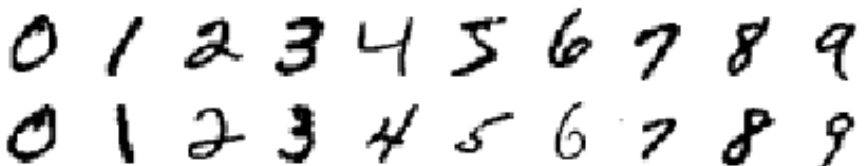
Recognizing handwritten digits

Given a 28×28 image, guess which digit it is.

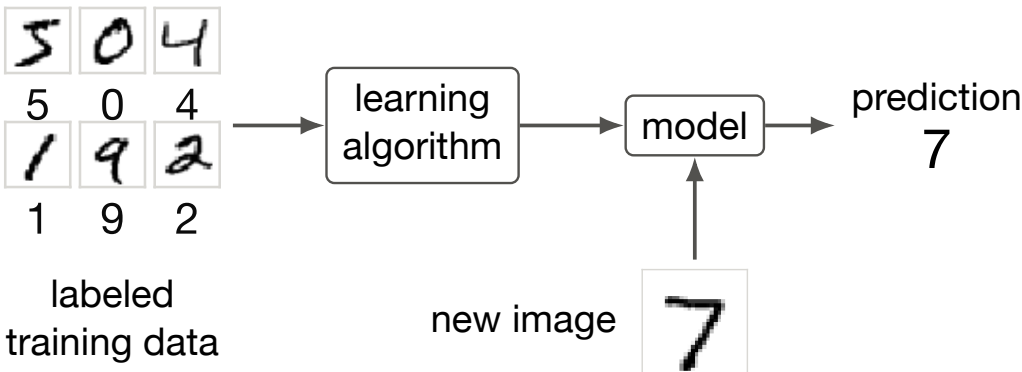


$\Longrightarrow$ 3

Some more examples:



Supervised learning



Supervised learning models

- Logistic regression
- k-nearest neighbors (k-NN) $\longleftarrow$ **today**
- Naive Bayes
- Decision tree
- Random forest
- Support vector machine (SVM)

Notation: a data point

$x^{(8)} =$ 

$y^{(8)} = 3$

$x^{(i)}$: the i-th training image

$y^{(i)}$: its label, a digit from 0 to 9

The superscript (i) is an index.

Later, a subscript j numbers the coordinates of one image: x_j.

The MNIST data set



- Training set: 60,000 images and their labels
- Test set: 10,000 images and their labels

MNIST (LeCun, Cortes, Burges), via OpenML data set 554; the first 144 training images

Nearest-neighbor classification

Training images $x^{(1)}, x^{(2)}, \ldots, x^{(60{,}000)}$

Labels $y^{(1)}, y^{(2)}, \ldots, y^{(60{,}000)}$: digits from 0 to 9



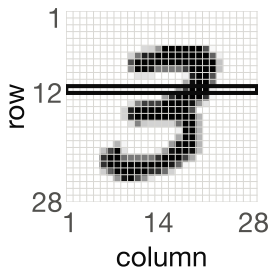
new image x

How do we classify a new image x?

- Find its nearest training image; call its index i^*, so it is $x^{(i^*)}$.
- Return the label $y^{(i^*)}$ of that neighbor.

Images as vectors

How do we measure the distance between two images?



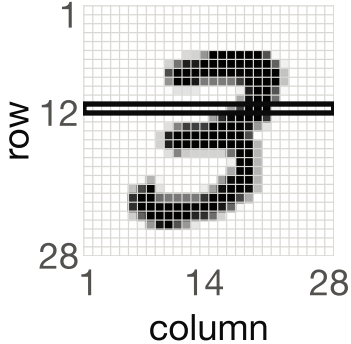
Read the 28×28 pixels row by row:

$$x = (x_1, x_2, \ldots, x_{784})$$

x_j: gray level of pixel j
0: white; 255: black

So we need a distance between vectors.

One image as a vector

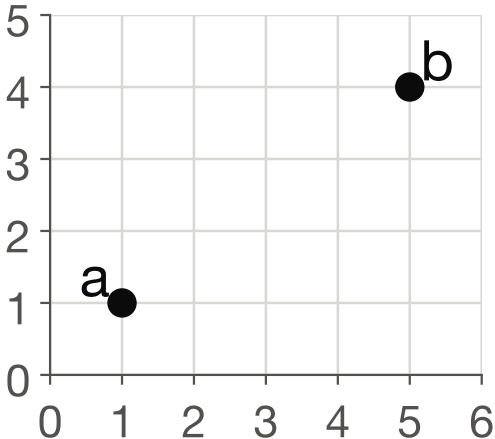


The image $x^{(8)}$ (label 3).

Boxed: row 12.

$$x^{(8)} = (0, 0, 0, \ldots, \underbrace{88,\ 189,\ 252,\ 252,\ 252,\ 14}_{x^{(8)}_{325}, \ldots, x^{(8)}_{330}\text{: in row 12}}, \ldots, 0, 0, 0)$$

Distance between two points



Euclidean distance in two dimensions is

Euclidean distance in higher dimensions

Two images a and b:

$$a = (a_1, a_2, a_3, \ldots, a_{784}) \qquad b = (b_1, b_2, b_3, \ldots, b_{784})$$

The Euclidean distance between a and b is

$$\|a - b\|_2 = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_{784} - b_{784})^2}$$

$$= \sqrt{\sum_{j=1}^{784} (a_j - b_j)^2}$$

$\|\cdot\|_2$: the 2-norm, the square root of a sum of squares.

Nearest neighbor in Euclidean distance

To classify a new image x:

1. Compute the distance $\|x - x^{(i)}\|_2$ to every training image, $i = 1, \ldots, 60{,}000$.
2. Nearest neighbor: $i^* = \arg\min_i \|x - x^{(i)}\|_2$.
3. Predict its label: $\widehat{y} = y^{(i^*)}$.

This is the nearest-neighbor rule (**1-NN**).

A new image and its nearest neighbor

new
image x



label 7

nearest
training image



label 7

$i^* = 53844$, at distance 676.6:
1-NN predicts 7, which is correct

Checkpoint 1

1-NN on the MNIST training set

Predict each of its 60,000 images.

What is the accuracy?

accuracy: the share of images with the correct label

Checkpoint 1

1-NN on the MNIST training set

Predict each of its 60,000 images.

What is the accuracy?

accuracy: the share of images with the correct label

100%: each image is its own nearest neighbor

at distance 0; no two MNIST training images are identical

Accuracy of 1-NN on MNIST



the first 144 of the 10,000 test images

1-NN predicts each of the 10,000 test images.

Question: What is the accuracy?

Test accuracy and test error

Test set: 10,000 images outside the training set.

$$\text{test accuracy} = \frac{\#\text{ classified correctly}}{\#\text{ test images}}$$

$$\text{test error} = \frac{\#\text{ misclassified}}{\#\text{ test images}} = 1 - \text{test accuracy}$$

1-NN misclassifies 309 of the 10,000 MNIST test images.

Test error 3.09%; test accuracy 96.91%.

Examples of errors



the first eight errors in test order, with their labels

How can we do better?

Summary: 1-NN

- An image is a vector of 784 pixel values; the Euclidean distance between two images a and b is $\|a - b\|_2 = \sqrt{\sum_j (a_j - b_j)^2}$.
- 1-NN predicts the label of the nearest training image.
- On MNIST each training image is its own nearest neighbor.
- The test error estimates the error rate on new images: 1-NN misclassifies 309 of the 10,000 MNIST test images.

k-nearest neighbor classification

To classify a new image x:

1. Fix a number k. Find the k training images nearest to x; their indices i form the set $N_k(x)$.
2. Predict the most common label among the $y^{(i)}$, $i \in N_k(x)$. The winning label may have fewer than half of the votes.

$k = 1$ is the nearest-neighbor rule.

If two labels tie, the library scikit-learn (default `weights="uniform"`) picks the label that comes first in sorted order, that is, the smaller digit.

A 5-NN vote

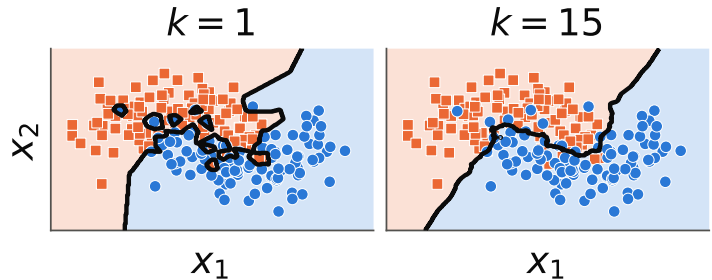


the first test image that 1-NN misclassifies
and 5-NN classifies correctly;

the neighbors are shown nearest first

votes: 8 three times, 3 once, 5 once; 5-NN predicts 8

Decision boundaries for $k = 1$ and $k = 15$



simulated data: class 1, class 0; shading: prediction
decision boundary: the curve where
the predicted class changes

$k = 1$ fits every single point; $k = 15$ is smoother

Choosing k

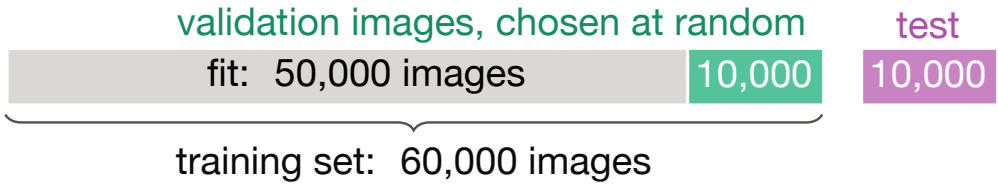
We computed the 1-NN test error (3.09%) only as an example; we do not use the test set to choose k.

$k = 1, 3, 5, 7, 9, 11$: which k does best on new images?

Choose the k with the smallest test error? Then the test set would help choose k, and its error would no longer estimate the error on new images.

Choose k before the final evaluation on the test set: hold out part of the training set as a **validation set**.

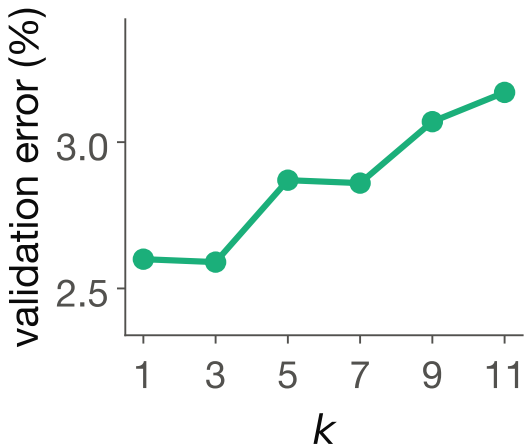
A validation set



1. For $k = 1, 3, \ldots, 11$: fit on the 50,000 images; compute the error on the 10,000 validation images.
2. Choose the k with the smallest validation error.
3. Fit with that k on all 60,000 training images; compute the test error once.

Validation error for each k

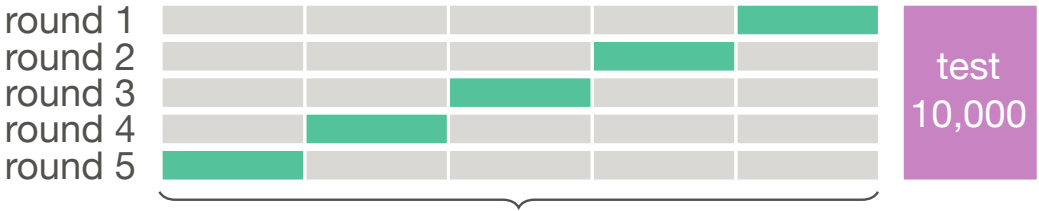
k	errors (of 10,000)	error (%)
1	260	2.60
3	259	2.59
5	287	2.87
7	286	2.86
9	307	3.07
11	317	3.17



Choosing k on the validation set

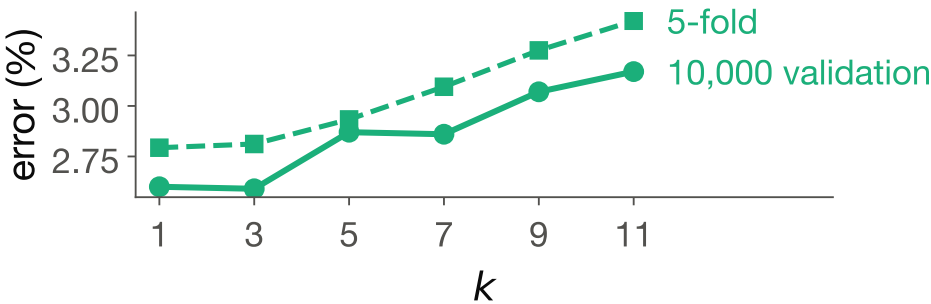
- Smallest validation error: $k = 3$ (2.59%).
- Fit $k = 3$ on all 60,000: test error 2.95%, computed once.

Cross-validation



- Each round, the aqua fold is the validation set; fit on the other four folds.
- Average the five errors for each k; choose the smallest.
- Fit with that k on all 60,000; compute the test error once.

5-fold cross-validation on MNIST



Five-fold cross-validation picks $k = 1$; $k = 1$ and $k = 3$ are almost tied.

Features on different scales

- k-NN compares points by distance: $\|a - b\|_2^2$ adds up $(a_j - b_j)^2$ over the features (coordinates) j.
- A feature with a large spread then dominates the distance.
- Standardize features that have different units (Lecture 2).
- MNIST: all pixels are gray levels in one unit, 0 to 255, so we use them as they are. Dividing every pixel by 255 does not change the nearest neighbors.

Summary: k-NN and the choice of k

- k-NN predicts the most common label among the k nearest training images.
- A small k fits single training points closely; a large k gives a smoother boundary.
- Choose k on a validation set or by cross-validation, without the test set; then compute the test error once.
- For k-NN, standardize features that have different units.

Checkpoint 2

k-NN on MNIST with $k = 60{,}000$:

every training image votes.

What does it predict for a new image?

Checkpoint 2

k-NN on MNIST with $k = 60{,}000$:

every training image votes.

What does it predict for a new image?

the most common label, 1, for every image

6,742 training images are 1s; test accuracy 11.35%

Cosine similarity

Two nonzero vectors $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_m)$, with the dot product $a \cdot b = \sum_{j=1}^{m} a_j b_j$ and the length (norm) $\|a\|_2 = \sqrt{a \cdot a}$:

$$s_{\cos}(a, b) = \frac{a \cdot b}{\|a\|_2 \, \|b\|_2} = \frac{a}{\|a\|_2} \cdot \frac{b}{\|b\|_2}$$

The second form scales a and b to length 1 first.

What cosine similarity measures

- $s_{\cos}(a, b)$ is the cosine of the angle between a and b, so $-1 \leq s_{\cos}(a, b) \leq 1$.
- $s_{\cos} = 1$: same direction; $s_{\cos} = 0$: perpendicular; $s_{\cos} = -1$: opposite.
- $s_{\cos}$ is a similarity: larger means more similar directions.
- The cosine distance is $1 - s_{\cos}(a, b)$, from 0 to 2.
- k-NN can use the cosine distance instead of the Euclidean distance: the nearest images have the smallest cosine distance.

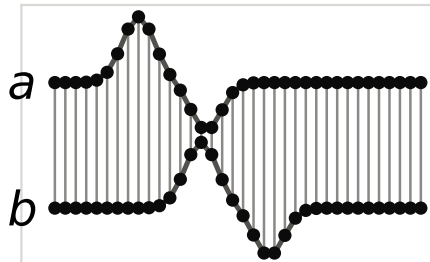
Cosine similarity: an example

$$a = (1, 2, 2) \qquad b = (3, 4, 0)$$

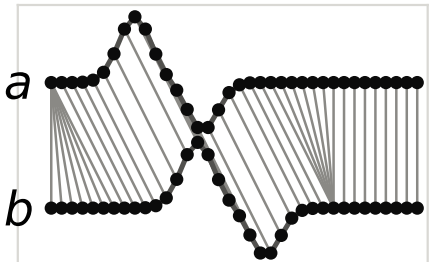
$$s_{\cos}(a, b) =$$

Comparing two time series

point by point



dynamic time warping



b is a shifted by six steps; lines: which points are compared

dynamic time warping (DTW) matches the two series

1-NN with DTW as the distance
compares time series that are shifted.

Dynamic time warping: the table D

Two sequences $a = (a_1, \ldots, a_P)$ and $b = (b_1, \ldots, b_Q)$.

A **matching** is a list of pairs (a_p, b_q) in which every a_p and every b_q occurs at least once, the order is kept, and each step goes forward in a, in b, or in both.

$|a_p - b_q|$: the cost of pairing a_p with b_q.

$D[p, q]$: the smallest cost of matching $a_1, \ldots, a_p$ with $b_1, \ldots, b_q$.

$$D[p,q] = |a_p - b_q| + \min\{\underbrace{D[p-1,q]}_{b_q \text{ repeats}}, \underbrace{D[p,q-1]}_{a_p \text{ repeats}}, \underbrace{D[p-1,q-1]}_{a_p \text{ and } b_q \text{ advance}}\}$$

Dynamic time warping: the steps

1. Set $D[0,0] = 0$ and put ∞
   in the rest of row 0 and column 0.

2. Fill the table row by row, $p = 1, \ldots, P$ and $q = 1, \ldots, Q$:

$$D[p,q] = |a_p - b_q| + \min\{D[p-1,q],\ D[p,q-1],\ D[p-1,q-1]\}$$

3. Return $\mathrm{DTW}(a,b) = D[P,Q]$.

The ∞ entries make every matching start with a_1 and b_1.

Dynamic time warping: an example

a \ b		**1**	**1**	**3**	**5**	**2**	**1**
	0	∞	∞	∞	∞	∞	∞
1	∞	0	0	2	6	7	7
3	∞	2	2	0	2	3	5
4	∞	5	5	1	1	3	6
2	∞	6	6	2	4	1	2
1	∞	6	6	4	6	2	1
1	∞	6	6	6	8	3	1

$a = (1, 3, 4, 2, 1, 1)$
$b = (1, 1, 3, 5, 2, 1)$

Fill D row by row.

Boxed: $D[4, 5]$; dashed: its three predecessor cells.

$$D[4,5] = |a_4 - b_5| + \min\{\text{dashed}\}$$
$$= |2-2| + \min\{3,\ 4,\ 1\} = 1$$

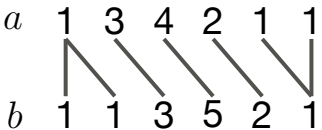
$$\mathrm{DTW}(a, b) = D[6, 6] = 1$$

The cheapest path

	b	**1**	**1**	**3**	**5**	**2**	**1**
a	0	∞	∞	∞	∞	∞	∞
1	∞	0	0	2	6	7	7
3	∞	2	2	0	2	3	5
4	∞	5	5	1	1	3	6
2	∞	6	6	2	4	1	2
1	∞	6	6	4	6	2	1
1	∞	6	6	6	8	3	1

Start at $D[6, 6]$; step back to the predecessor cell that gave the minimum.

Shaded $D[p, q]$:
a_p is matched with b_q.



The DTW cost is 1; pairing a_p with b_p for every p costs $\sum_p |a_p - b_p| = 7$.

k-NN regression

The label y is now a number on a continuous scale.

New point x. $N_k(x)$: the indices of its k nearest neighbors.

Prediction: the average of their labels,

$$\widehat{y} = \frac{1}{k} \sum_{i \in N_k(x)} y^{(i)}$$

Example of 5-NN regression

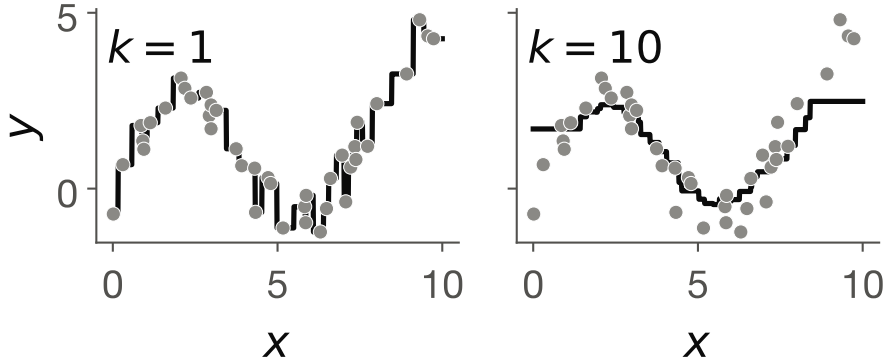
Training set $(x^{(i)}, y^{(i)})$:

$$(2,1) \quad (5,3) \quad (10,6)$$
$$(4,9) \quad (1,3) \quad (8,2)$$
$$(5,8) \quad (7,8) \quad (1,4)$$

New point: $x = 6$

The 5-NN prediction is

k-NN regression on simulated data



gray: 40 training points; black: the prediction $\widehat{y}$ at each x

$k = 1$ fits every single point; $k = 10$ is smoother

Summary: cosine, DTW and regression

- Cosine similarity compares directions: from -1 to 1, larger is more similar; the cosine distance is $1 - s_{\cos}$.
- Dynamic time warping compares shifted or stretched time series: its value is the cost of the cheapest matching, computed with the table D.
- k-NN regression predicts the average label of the k nearest training points.

k-NN in scikit-learn

```
from sklearn.datasets import fetch_openml
from sklearn.model_selection import GridSearchCV, KFold
from sklearn.neighbors import KNeighborsClassifier

X, y = fetch_openml("mnist_784", version=1, return_X_y=True,
                    as_frame=False, parser="liac-arff")
grid = {"n_neighbors": [1, 3, 5, 7, 9, 11]}
folds = KFold(5, shuffle=True, random_state=0)
search = GridSearchCV(KNeighborsClassifier(), grid, cv=folds)
search.fit(X[:6000], y[:6000])  # 6,000 images, for speed
print(search.best_params_)  # -> {'n_neighbors': 1}
# rows 60000 on: the 10,000 test images
print(search.score(X[60000:], y[60000:]))  # -> 0.9368
```