k-Nearest Neighbor

Gautam Kamath

Binary Classification

- Given: $\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \ldots$
  - $x^{(i)}$: "feature vector." Often $x_i \in \mathbf{R}^d$
  - $y^{(i)}$: "label." For binary classification, $y^{(i)} \in \{-1, +1\}$
    - You may also see $y^{(i)} \in \{0,1\}$
- Idea: "Learn" a function h such that $h(x) = y$
  - Given a feature vector, what is the label?

Pass class example

- Feature vector $x^{(i)}$: (homework score, exam score)
- Label $y^{(i)}$: Passed the class?
- Dataset (draw on board):
  - $\big((90, 80), +1\big), \big((40, 30), -1\big), \big((50, 40), -1\big)$
- Can always memorize training data
  - But we want to *generalize*!
  - $\big((50{,}60), ?\big)$

Image Classifier example

- Feature vector $x^{(i)}$:

$=$

$$\begin{bmatrix} 9 & 13 & 5 \\ 1 & 11 & 7 \\ 2 & 6 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \\ 2 & 6 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ 7 \\ 2 & 6 & 3 \end{bmatrix}$$

- Label $y^{(i)}$: Is this a panda or not?

Statistical Learning

- Setup: Given $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \sim_{i.i.d.} P$
  - Independent and identically distributed – may be limiting, but common assn
- Goal: Learn $h : \mathbf{R}^d \to \{-1, +1\}$ such that $\Pr_{(x,y)\sim P}[h(x) = y]$ is large
  - Importantly, P is unknown (otherwise could use the "Bayes classifier")
  - What happens if we get something "out of distribution"?
    - (Draw two clusters on board, wrong label and unpredictable examples)
  - When is learning "possible"?
    - Consider $y \sim \text{Bernoulli}\left(\frac{1}{2}\right)$ (not dependent on x)
    - Effective learning needs some correlation between x and y
    - *Inductive bias*: what does learning method *assume* about relationship between x and y?

1-Nearest Neighbors (1NN)

- Inductive bias (rather informal/imprecise): Similar feature vectors should have the same label
- Let $\text{dist}(x, x') = \|x - x'\|_2 = \sqrt{\sum_j \left(x_j - x'_j\right)^2}$
- 1-Nearest Neighbor:
  - Find the example in the training set (x^*, y^*) closest to feature vector x

$$x^* = \arg \min_{x^{(i)} \in \{x^{(1)}, \ldots, x^{(n)}\}} \left\|x - x^{(i)}\right\|_2$$

  - Output y^*
- (draw example, an intuitive case, and a “buried” counterintuive case)

k-Nearest Neighbors (kNN)

- k-Nearest Neighbor:
  - Find the k examples in the training set $(x^*_{(1)}, y^*_{(1)}), \ldots, (x^*_{(k)}, y^*_{(k)})$ closest to feature vector x
  - Output majority of $y^*_{(1)}, \ldots, y^*_{(k)}$
- (revisit previous example, show how it can fix it)
- ~~Inductive bias (rather informal/imprecise): Similar feature vectors should have the same label~~
- Inductive bias (more precisely): Similar feature vectors should have similar probability of being labeled $+1$

kNN Inductive Bias

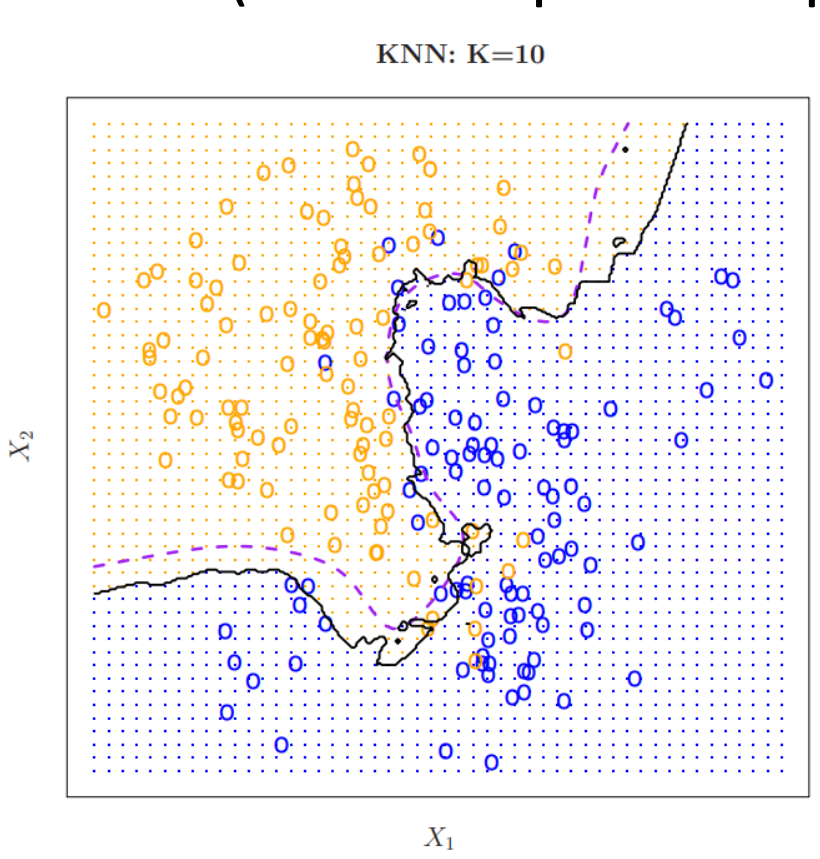
- Let $\eta(x) = \Pr[\, Y = 1 \mid X = x \,]$
  - Given feature vector x, what is the conditional probability that the label is 1?
    - Conditional class probability, “regression function”
- Inductive bias: Similar feature vectors should have similar conditional class probabilities
  - One formalization: the conditional class probability is *Lipschitz*
  - $|\eta(x) - \eta(x')| \leq L \cdot \mathrm{dist}(x, x')$
    - L is the Lipschitz constant, $\mathrm{dist}(\cdot,\cdot)$ is some distance measure
- kNN essentially approximates the regression function
  - $\hat{\eta}_k(x) = \frac{1}{k} \sum_{i \in N_k(x)} \mathbf{1}\{y^{(i)} = +1\}$

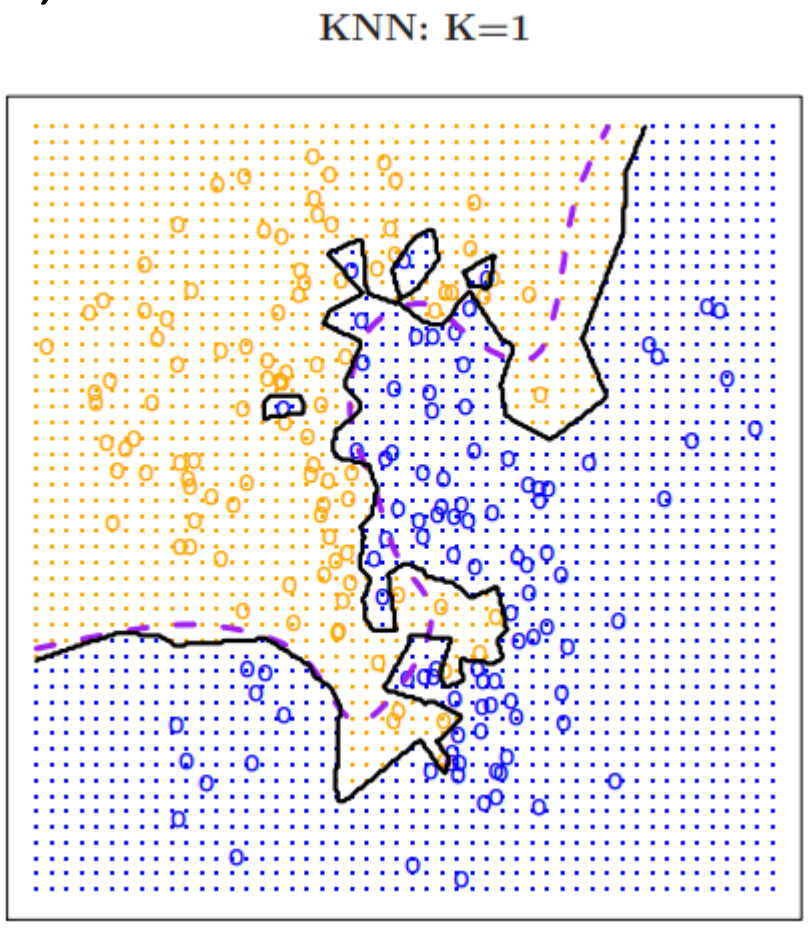
A little more abstractly

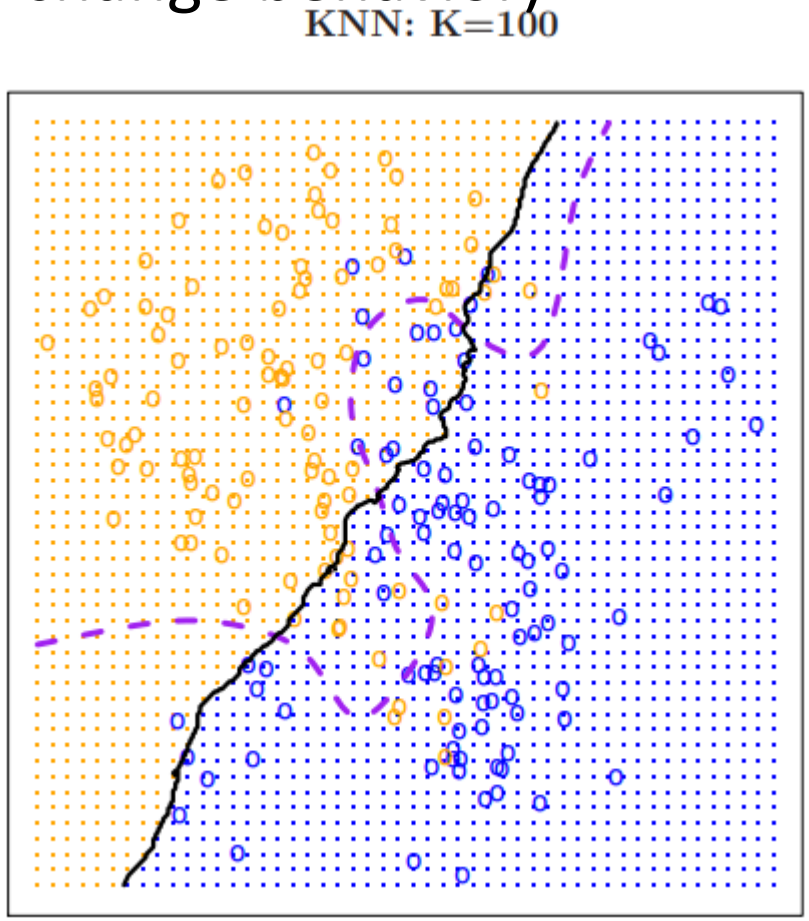
- kNN:
  - Find the **closest** k points to the feature vector x
  - **Aggregate** their labels
- Thus far (and most intuitive): distance is ℓ_2 norm, aggregation is majority vote
  - But could change either/both!
  - Especially distance, e.g., ℓ_p norm, some fixed or learned mapping, etc.
    - Will revisit a bit later

The role of k

- (Draw simple example, show how values of k can change behavior)







Figures from ISL

The role of k

- What can go wrong when choosing k?
- Too small (e.g., $k = 1$)
  - May *overfit* to individual points and random noise in the training data
- Too large (e.g., $k = n$)
  - May *underfit* by averaging over too many points, and overlook structure
- How do we choose?

The role of k

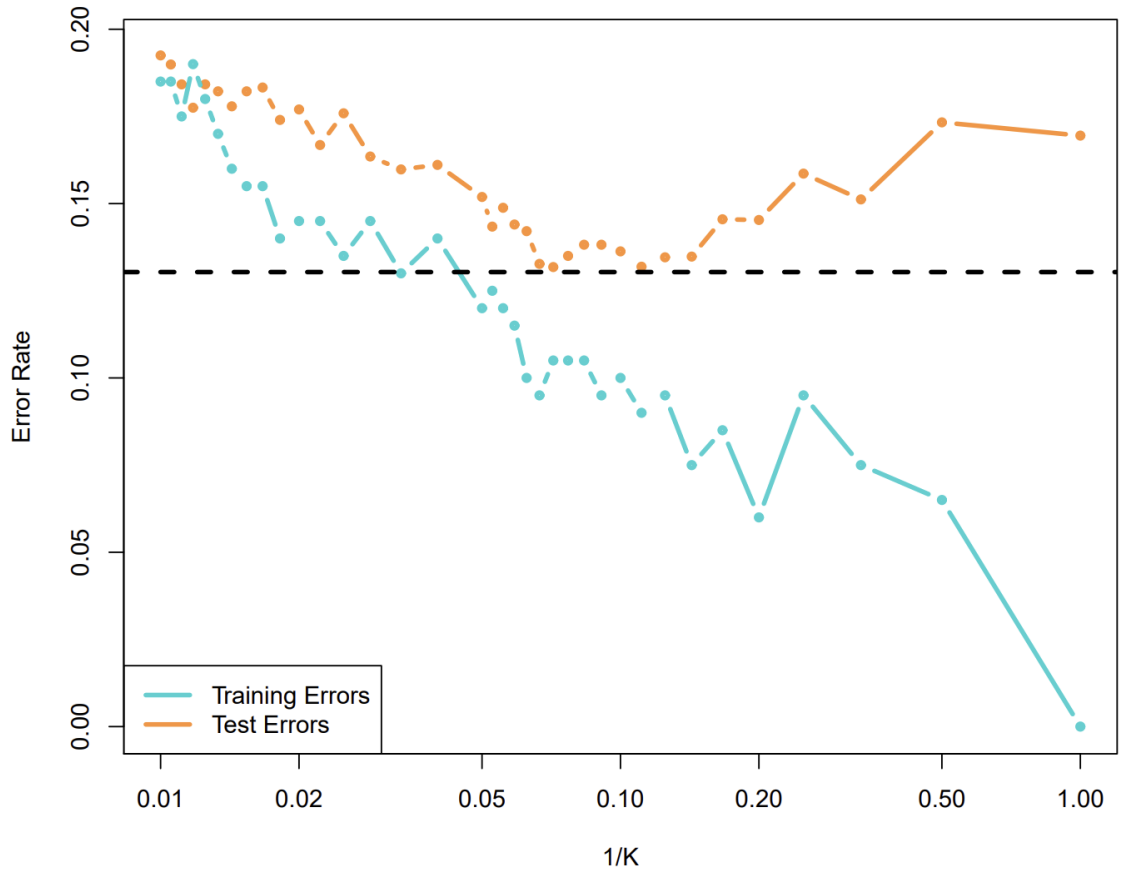


Figure from ISL

Hyperparameter selection

- Hyperparameter: any design choice in the learning process
  - (e.g., k, distance choice, aggregation method)
- Types of datasets
  - Training (fit model), validation (choose hyperparams), test (evaluate model)
- Use validation to mitigate overfitting to training data
- Can try different hyperparameters using validation – but not too many/adaptively or you'll overfit to training + validation
  - E.g., commit to $k = \{3, 5, 7, 9, 11\}$, train all models on training data, choose the best one via the validation set

Hyperparameter selection

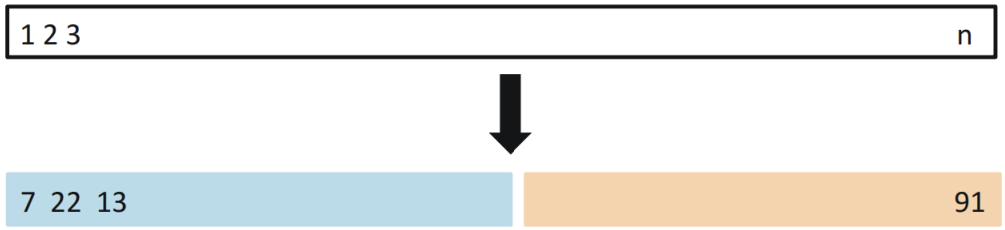


FIGURE 5.1. *A schematic display of the validation set approach. A set of n observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.*

Figure from ISL

Hyperparameter selection

- Hyperparameter: any design choice in the learning process
  - (e.g., k, distance choice, aggregation method)
- Types of datasets
  - Training (fit model), validation (choose hyperparams), test (evaluate model)
- Use validation to mitigate overfitting to training data
- Can try different hyperparameters using validation – but not too many/adaptively or you'll overfit to training + validation
  - E.g., commit to $k = \{3, 5, 7, 9, 11\}$, train all models on training data, choose the best one via the validation set
- What if we have no validation set?

Cross Validation

- Split training data into t sets (draw on board), e.g. $t = 10$ is common

For each λ:

For $i = 1$ to t:

$w_{\lambda,i}$ = train on all data but split i with hyperparameter λ

$\text{perf}_{\lambda,i}$ = performance of $w_{\lambda,i}$ on the split i

$\text{perf}_{\lambda} = \sum_i \text{perf}_{\lambda,i}$

Return λ which has the biggest perf_{λ}

Cross Validation

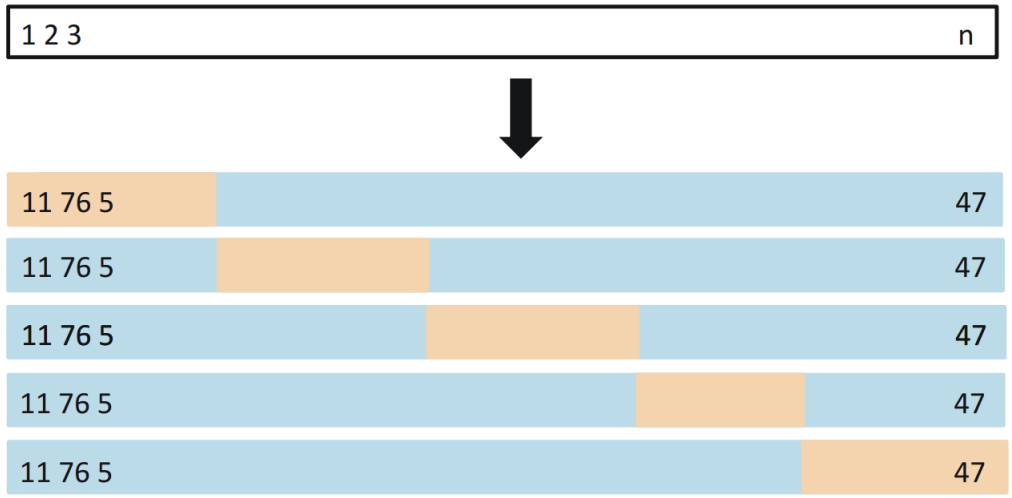


FIGURE 5.5. *A schematic display of 5-fold CV. A set of n observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.*

Figure from ISL

…Euclidean distance?

- Features are a human's height and weight (draw)
  - Case 1: height is in centimeters, weight is in nanograms
  - Case 2: height is in nanometers, weight is in kilograms
- In case 1, Euclidean distance ignores height, in case 2 it ignore weight
- Standardization: recenter and rescale data so data lives on same scale

$$\widetilde{x_j} = \frac{x_j - \mu_j}{\sigma_j}$$

- Subtract out mean of the coordinate, rescale by standard deviation of the coordinate
- Suppose features have same variance. Can Euclidean distance fail?

Features of differing importance

- Two features: height, and what day of week someone is born. Label: whether they make it to the NBA
- Height is clearly the much more predictive feature
  - Even if we did rescaling
- $X_1, X_2 \sim N(0,1)$
- $\Pr[\, Y = 1 \mid X = x \,] = f(X_1)$, the label depends only on one feature
- There are methods that can adapt the distance metric to the data

How good can kNN be?

- Suppose we had infinite data – would 1NN be "optimal"?
- What is optimal? *Bayes classifier*
- Recall: goal in classification is to solve $\min_h \Pr_{x,y\sim P}[h(x) \neq y]$
- Suppose we actually knew P (unrealistic), what would we predict?
  - $h^*(x) = \arg\max_{y\in\{\pm 1\}} \Pr[Y = y \mid X = x]$
    - Probability is with respect to P (which, in realistic settings we don't know)
    - Choose whichever label is most likely given the feature vector x
- Optimal classifier, none can do better
  - (Draw examples, one with perfect BC and one with worst possible)
- Probability of error at feature vector x: $\min\{\eta(x), 1 - \eta(x)\}$
  - Recall $\eta(x) = \Pr[\, Y = 1 \mid X = x \,]$

Can we achieve Bayes error?

- Suppose we had infinite data – would 1NN be "optimal"?
- Suppose $\Pr[Y = 1 \mid X = x] = 0.9$ (independent of x)
  - Bayes classifier: always predict 1. Error: wrong w 0.1 probability everywhere
- For any test point x, with infinite training data, the closest point will have label $+1$ with probability 0.9, and the true label will be -1 with probability 0.1, error
- Similarly: the closest point will have label -1 with probability 0.1 and the true label will be $+1$ with probability 0.9, error
- Overall error: $0.9 \cdot 0.1 + 0.1 \cdot 0.9 = 0.19 > 0.1$, the Bayes error
- Label noise causes 1NN error to be larger than Bayes

A more general statement about 1NN

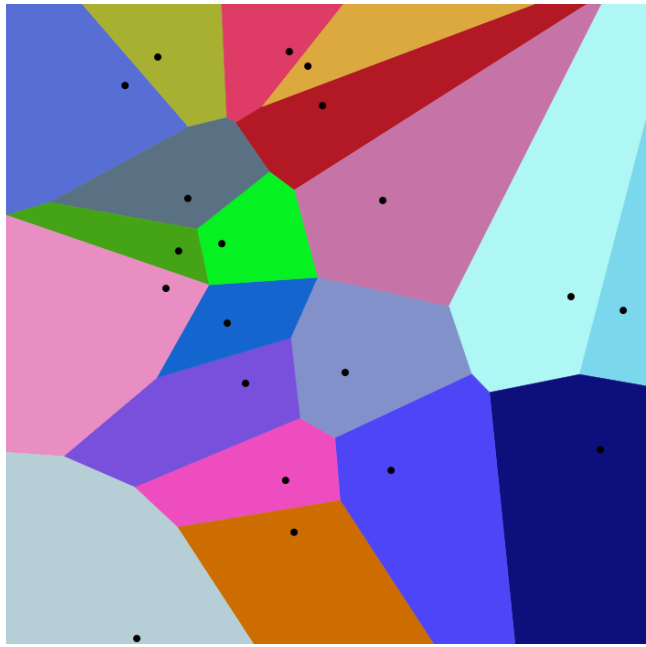
- Locally at x, Bayes error is $r^*(x) = \min\{\eta(x), 1-\eta(x)\}$
  - Expected error (over feature vector X) is $E[r^*(X)]$
- Locally at x, 1NN error (w/ infinite data) is $2r^*(x)\big(1-r^*(x)\big)$
  - Similar to before, $r^*(x)\big(1-r^*(x)\big) + \big(1-r^*(x)\big)r^*(x)$
- In expectation over feature vector X, 1NN error is:
  - $\mathrm{E}\big[2r^*(X)\big(1-r^*(X)\big)\big] = 2E[r^*(X)] - 2E\big[r^*(X)^2\big]$
  - $\leq 2E[r^*(X)] - 2E[r^*(X)]^2 = 2E[r^*(X)](1-E[r^*(X)])$
    - Inequality is Jensen's: $f(E[X]) \leq E[f(X)]$ for any convex function f
- $\mathrm{Err}_{1NN} \leq 2 \cdot \mathrm{Err}_{\mathrm{Bayes}} \cdot (1 - \mathrm{Err}_{\mathrm{Bayes}})$ [Cover-Hart '67]
  - Competitive with Bayes (within a factor of 2), but may be lossy (prev example)

Doing better with larger k?

- Consistency: a method converges to the optimal answer given infinite data
  - We just showed that 1NN is not consistent
- If we let $k_n \to \infty$ and $\frac{k_n}{n} \to 0$, then $\mathrm{Err}_{k_n NN} \to \mathrm{Err}_{\mathrm{Bayes}}$.
  - The hyperparameter k increases with n (averages out label noise), and
  - The hyperparameter k doesn't grow as quickly as n (keeps looking locally)
  - Then kNN is consistent, with error converging to the Bayes optimal
- Nothing tells you how big n needs to be: curse of dimensionality may necessitate it to be exponentially large

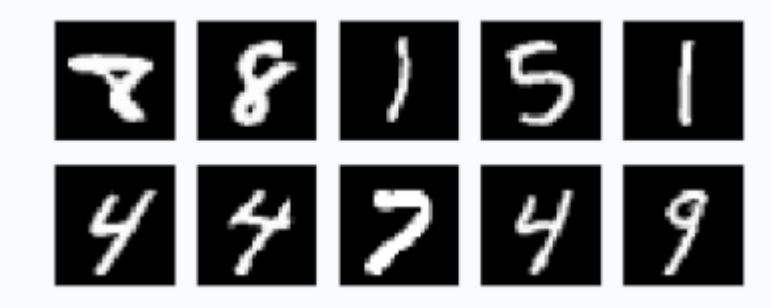
Time and Space Complexity

- A *non-parametric* method
- Training takes 0 time (just store dataset), but $O(nd)$ space
  - Comparable parametric methods need $O(d)$ space
- Classification of new point takes $O(ndk)$ time naively, $O(nd)$ space
  - Time can be reduced to $O(nd)$ time a bit more carefully
- Can do better in some cases
  - E.g., Voronoi diagram for 1-NN
    - Takes $O(d \log n)$ time, $n^{O(d)}$ space
    - Good in low-dimensional settings
  - Approximate nearest neighbors



Does it work?

- MNIST: black and white image classification
  - 60k train, 10k test points
  - $d = 28 \times 28 = 784$, 10 classes (0 through 9)
  - Canonical “easy ML task”



CLASSIFIER	PREPROCESSING	TEST ERROR RATE (%)	Reference
Linear Classifiers			
linear classifier (1-layer NN)	none	12.0	LeCun et al. 1998
K-nearest-neighbors, Euclidean (L2)	none	3.09	Kenneth Wilder, U. Chicago
K-nearest-neighbors, L3	none	2.83	Kenneth Wilder, U. Chicago
K-NN with non-linear deformation (IDM)	shiftable edges	0.54	Keysers et al. IEEE PAMI 2007
K-NN with non-linear deformation (P2DHMDM)	shiftable edges	0.52	Keysers et al. IEEE PAMI 2007
2-layer NN, 300 hidden units, mean square error	none	4.7	LeCun et al. 1998
Convolutional net LeNet-4	none	1.1	LeCun et al. 1998

Euclidean distance on pixels can be… bad

Something more modern: DINO (2021)

- Self-supervised learning on ImageNet (much harder dataset)
  - 1.28M images
  - 1K classes
  - Self-supervised = *without labels*
- Produces a feature map:
  - $x \to \phi(x)$
- Doing weighted kNN on this representation gives 78.3% acc