CS 412 Introduction to Machine Learning

K nearest neighbors

Instructor: Wei Tang

Department of Computer Science
University of Illinois at Chicago
Chicago IL 60607

https://tangw.people.uic.edu
tangw@uic.edu

Slides credit: Sargur N. Srihari, Noah Snavely

Regression vs Classification

Regression

Age	Income	Loan Amount
21	20000	0
37	55000	150000
29	35000	120000
23	17000	550000
34	70000	250000
47	84000	0
25	30000	90000

In regression problem, output is number

Classification

Age	Income	Loan Status
21	20000	Rejected
37	55000	Approved
29	35000	Approved
23	17000	Rejected
34	70000	Approved
47	84000	Rejected
25	30000	Approved

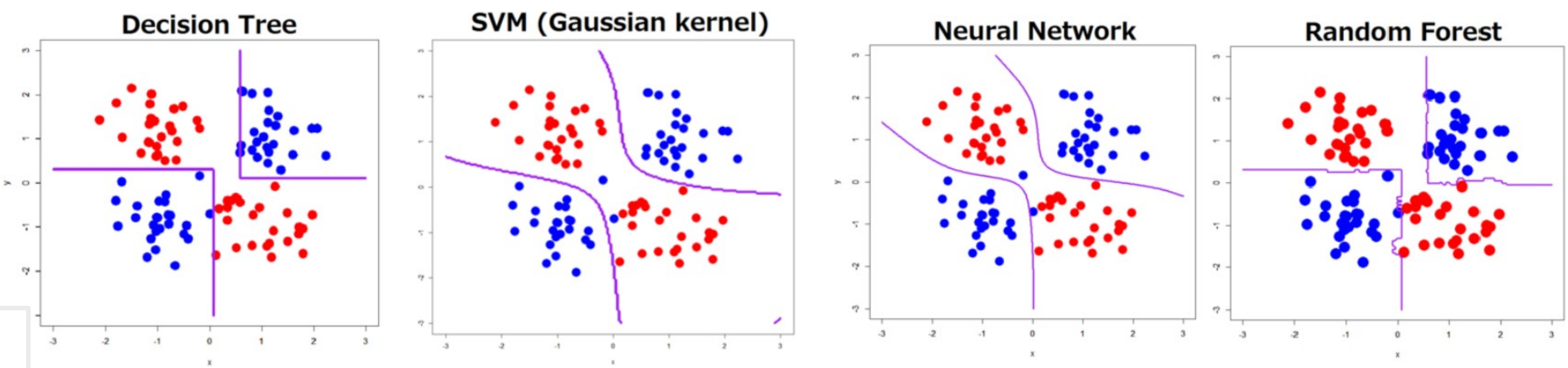
In classification problem, output is class label

Regression vs Classification

- In *Regression* we assign input vector $\boldsymbol{x}$ to one or more continuous target variables t
  - Linear regression has simple analytical and computational properties
- In *Classification* we assign input vector $\boldsymbol{x}$ to one of K discrete classes C_k, $k = 1, \ldots, K$
  - Common classification scenario: classes considered disjoint
    - Each input assigned to only one class
  - Input space is thereby divided into decision regions

Boundaries of decision regions

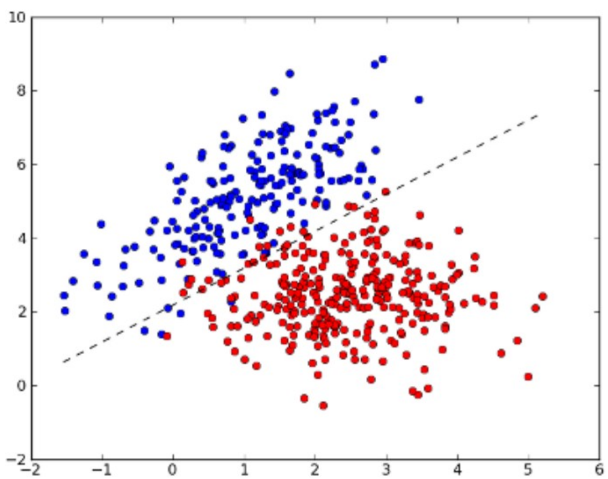
Boundaries are called *decision boundaries* or *decision surfaces*



Linear Classification Models

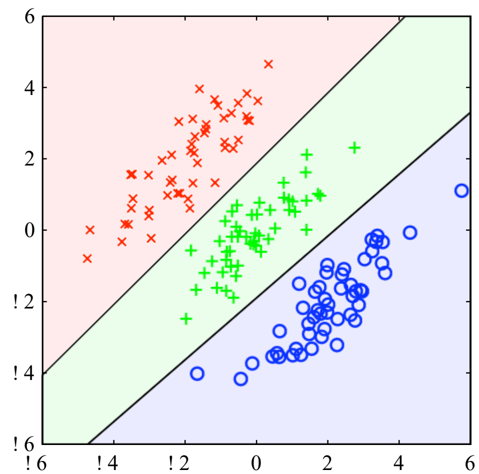
- Decision surfaces are linear functions of input $\boldsymbol{x}$
  - Defined by $(D-1)$ dimensional hyperplanes within D dimensional input space

K=2



Not linearly separable

K=3



Linearly separable

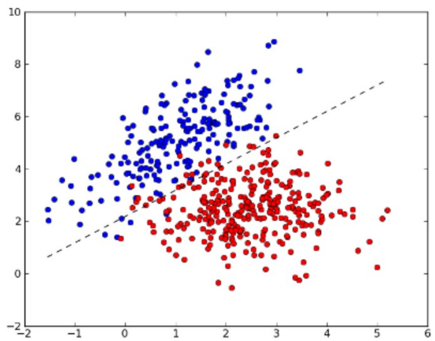
Straight line is 1-D decision boundary in 2-D space

A plane is 2-D surface in 3-D space

Data sets whose classes can be separated exactly by linear decision surfaces are said to be Linearly separable

Representing the target in Classification

- In *regression*:
  - target variable t is a real number (or vector of real numbers $\mathbf{t}$) which we wish to predict
- In *classification:*
  - there are various ways of using target values to represent class labels, depending on whether
    - Model is probabilistic
    - Model is non-probabilistic



Representing Class in Probabilistic Model

- Two class: Binary representation is convenient
  - Discrete $t \in \{0, 1\}$, $t = 1$ represents C_1,
    $t = 0$ means class C_2
    - Can interpret value of t as probability that class is C_1
    - Probabilities taking only extreme values of 0 and 1
- For $K > 2$: Use a $1\text{-of-}K$ coding scheme
  - $\mathbf{t}$ is a vector of length K
    - Eg. if $K = 5$, a pattern of class 2 has $\mathbf{t} = (0, 1, 0, 0, 0)^{\mathrm{T}}$
    - Value of t_k interpreted as probability of class C_k
      - If t_k assume real values then we allow different class probabilities

Three Approaches to Classification

1. Non-probabilistic models
   – Directly assign $\boldsymbol{x}$ to a specific class
     - E.g., K nearest neighbor
2. Probabilistic Models
   - Discriminative approach
     - Model $p(C_k|\boldsymbol{x})$ in *inference* stage (direct)
     - Use it to make *optimal* decisions
     - E.g., Logistic Regression
   - Generative approach
     - Model class-conditional density $p(\boldsymbol{x}|C_k)$
     - Together with $p(C_k)$ use Bayes rule to compute posterior
     - E.g., Naive Bayes classifier

Probabilistic Models: Generative/Discriminative

- Model $p(C_k \mid \boldsymbol{x})$ in an *inference* stage and use it to make optimal decisions
- Approaches to computing the $p(C_k \mid \boldsymbol{x})$
  1. Generative
     - Model class conditional densities by $p(\boldsymbol{x} \mid C_k)$ together with prior probabilities $p(C_k)$
     - Then use Bayes rule to compute posterior

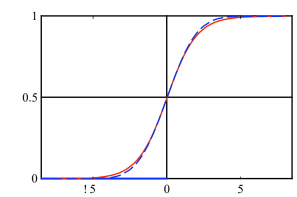
$$p(C_k \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid C_k)p(C_k)}{p(\boldsymbol{x})}$$

  2. Discriminative
     - Directly model conditional probabilities $p(C_k \mid \boldsymbol{x})$

Regression to Classification

- Linear Regression: model prediction $y\,(\boldsymbol{x},\boldsymbol{w})$ is a linear function of parameters $\boldsymbol{w}$
  - In simple case model is also a linear function of $\boldsymbol{x}$
    - Thus has the form $y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0$ where y is a real no.
- Classification: we need to predict class labels or posterior probabilities in range $(0,1)$
  - For this, we use a generalization where we transform the linear function of $\boldsymbol{w}$ using a nonlinear function $f\,(.)$, so that

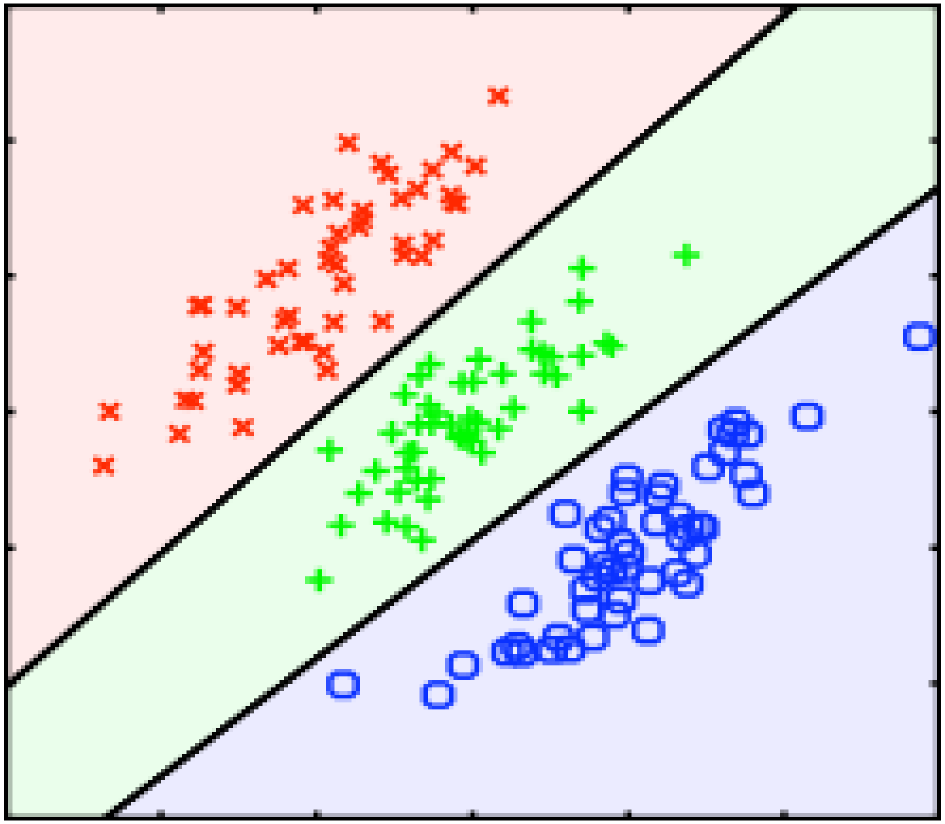
$$y(\boldsymbol{x})=f\,(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + w_0)$$



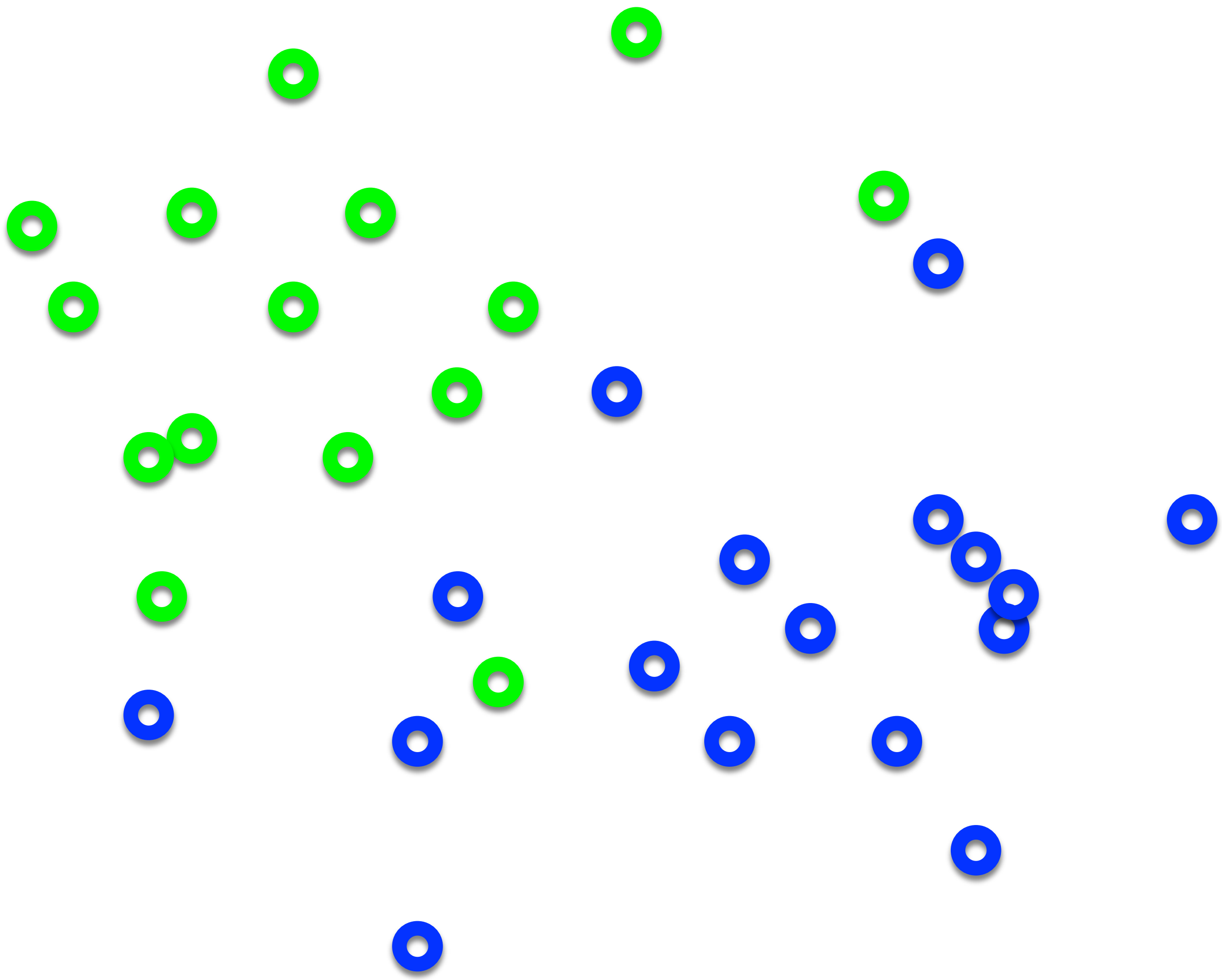
- $f\,(.)$ is known as an *activation function*

Decision surface of linear classifier

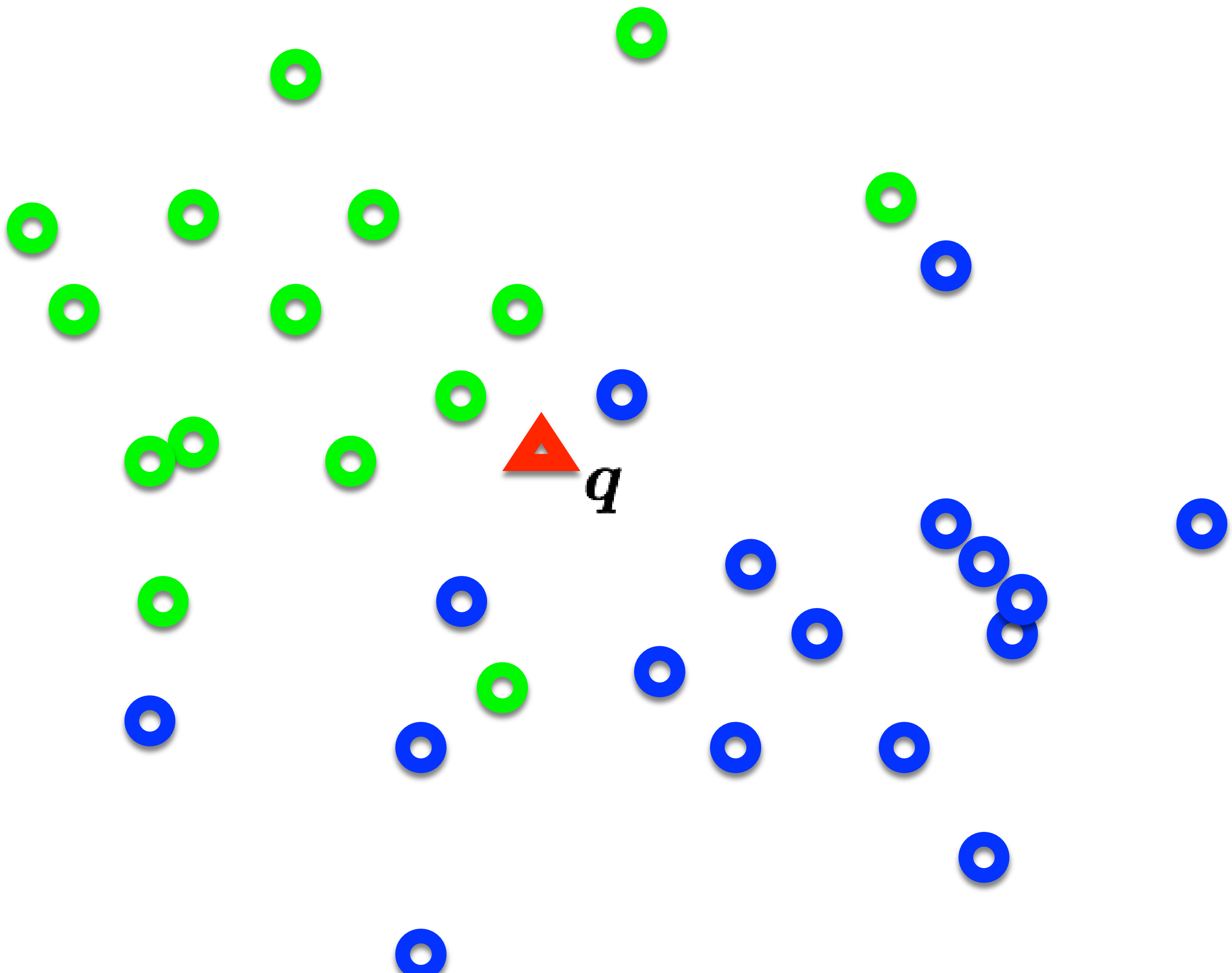
- Decision surfaces of $y(\boldsymbol{x})=f(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0)$ correspond to $y(\boldsymbol{x})=\text{constant}$ or $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0=\text{constant}$



Distribution of data from two classes

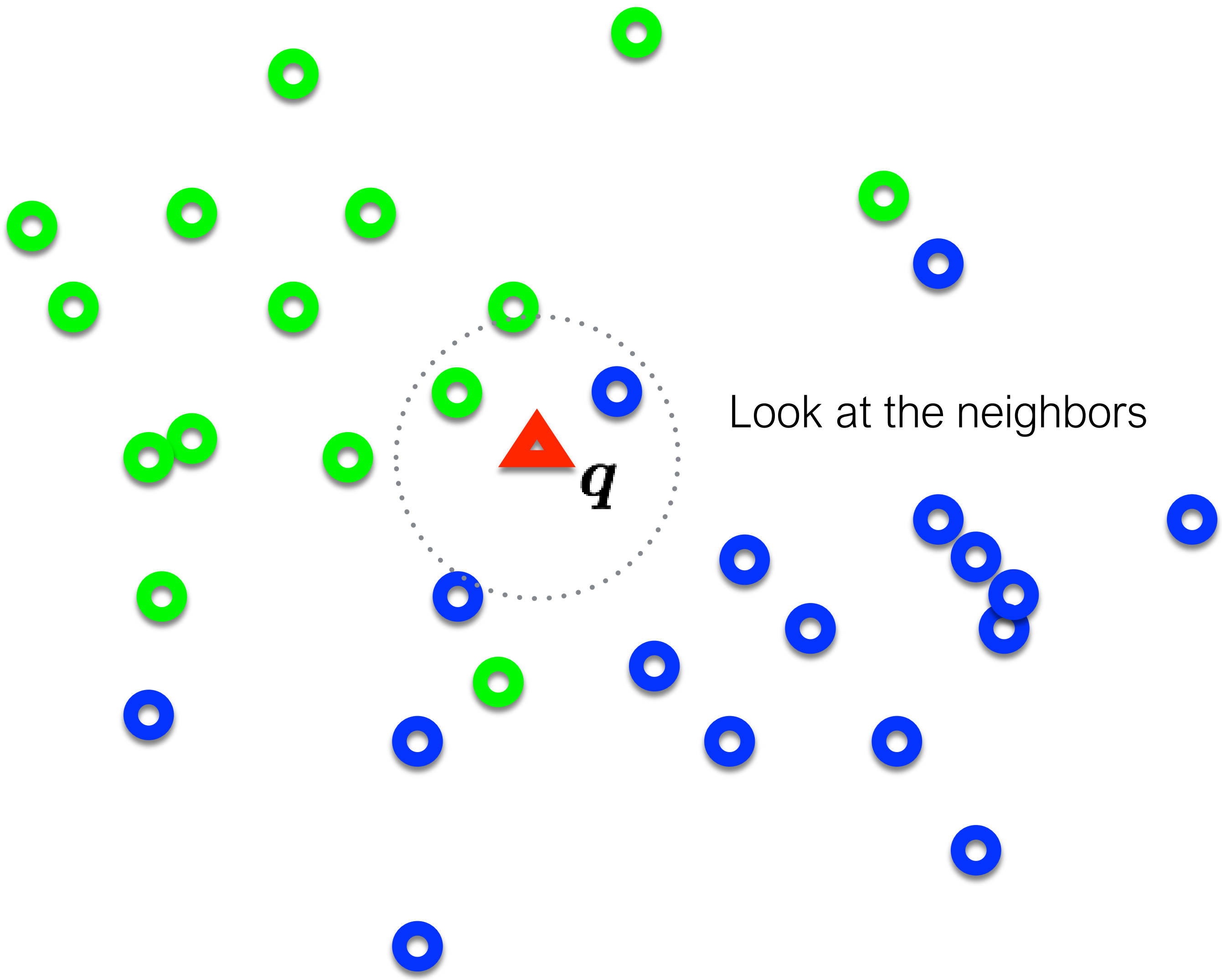


Distribution of data from two classes

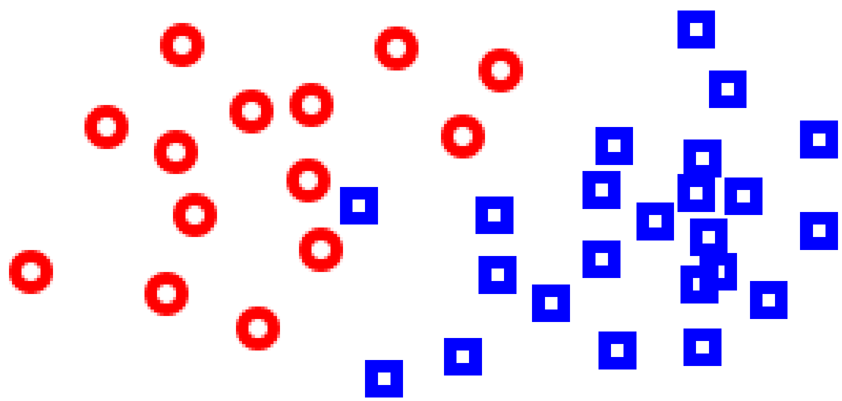


Which class does q belong too?

Distribution of data from two classes



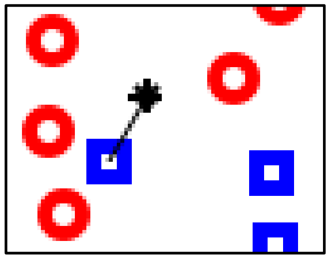
K-Nearest Neighbor (KNN) Classifier



Non-parametric pattern classification approach

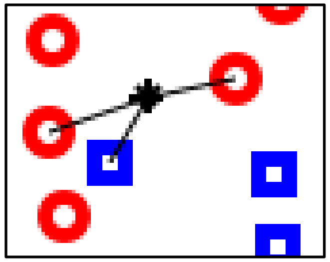
For a given query point q, assign the class of the nearest neighbor

k = 1



Compute the k nearest neighbors and assign the class by majority vote.

k = 3



Nearest Neighbor is competitive

MNIST Digit Recognition
- Handwritten digits
- 28x28 pixel images: d = 784
- 60,000 training samples
- 10,000 test samples

Yann LeCunn

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

What is the best distance metric between data points?

- Typically Euclidean distance
- Important to normalize.
  Dimensions have different scales

How many K?

- Typically k=1 is good
- Cross-validation (try different k!)

Distance metrics

$$D(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_N - y_N)^2}$$

Euclidean

$$D(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} = \frac{x_1 y_1 + \cdots + x_N y_N}{\sqrt{\sum_n x_n^2} \sqrt{\sum_n y_n^2}}$$

Cosine

Choice of distance metric

- Hyperparameter

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

L2 (Euclidean) distance

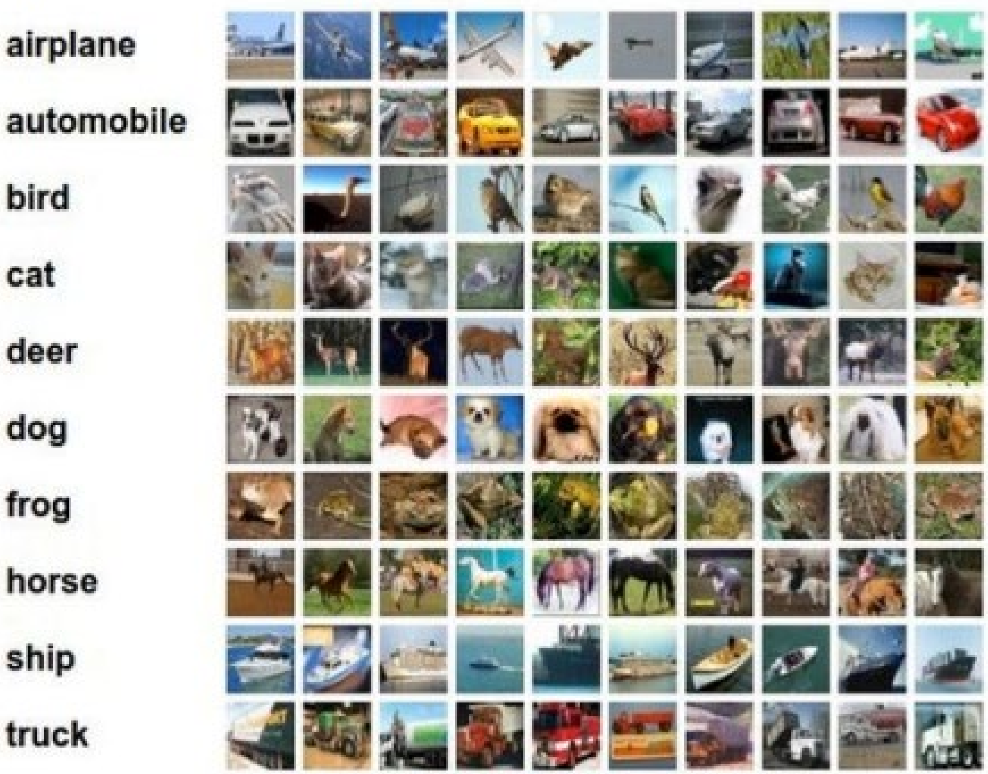
$$d_2(I_1, I_2) = \sqrt{\sum_p \left(I_1^p - I_2^p\right)^2}$$

- Two most commonly used special cases of p-norm

$$\|x\|_p = \left(|x_1|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}} \quad p \geq 1, x \in \mathbb{R}^n$$

CIFAR-10 and NN results

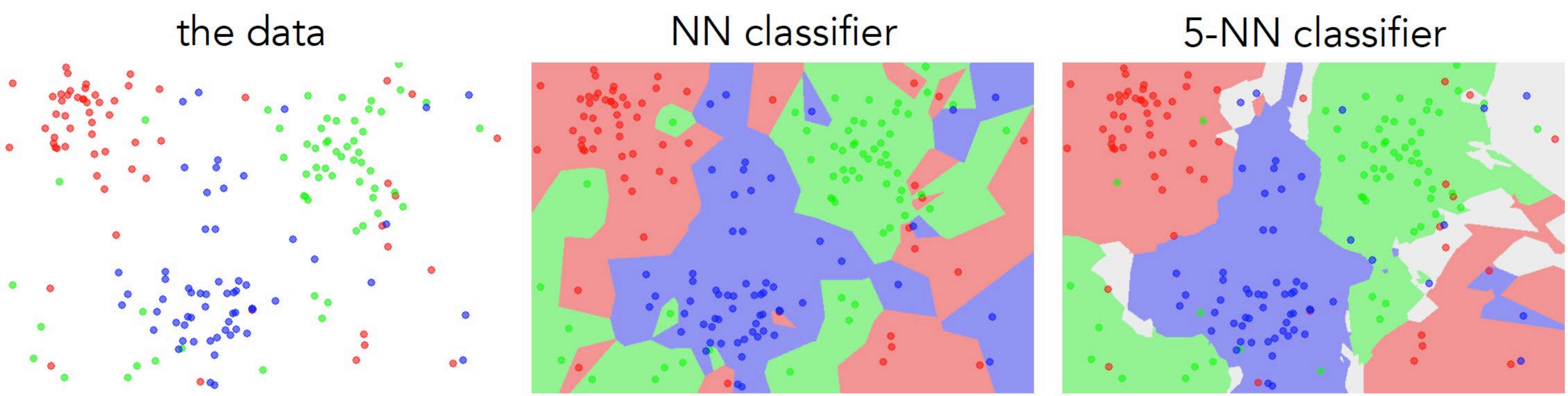
Example dataset: **CIFAR-10**
10 labels
50,000 training images
10,000 test images.



For every test image (first column), examples of nearest neighbors in rows

k-nearest neighbor

- Find the k closest points from training data
- Labels of the k points “vote” to classify



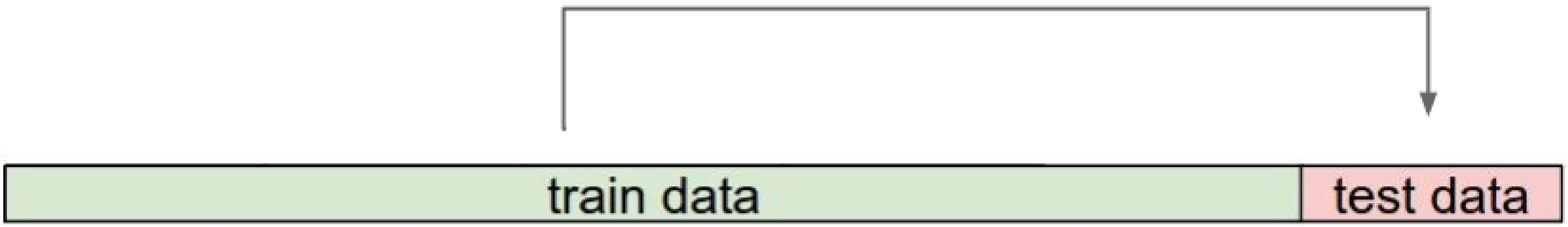
Hyperparameters

- What is the best distance to use?
- What is the best value of k to use?

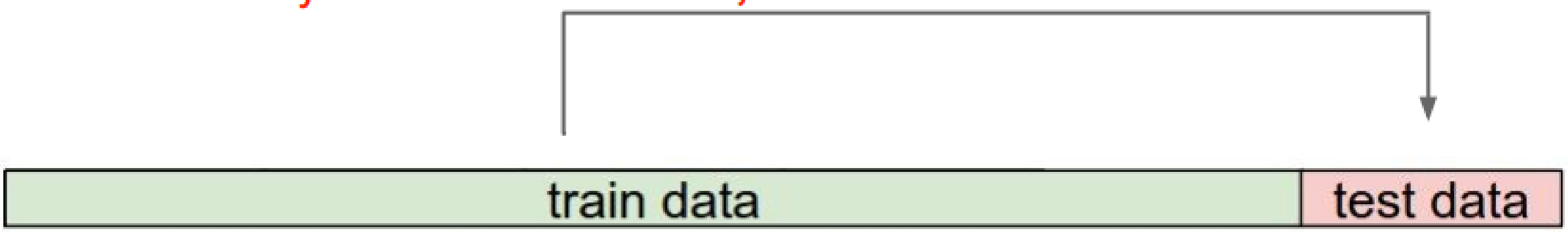
- i.e., how do we set the hyperparameters?

- Very problem-dependent
- Must try them all and see what works best

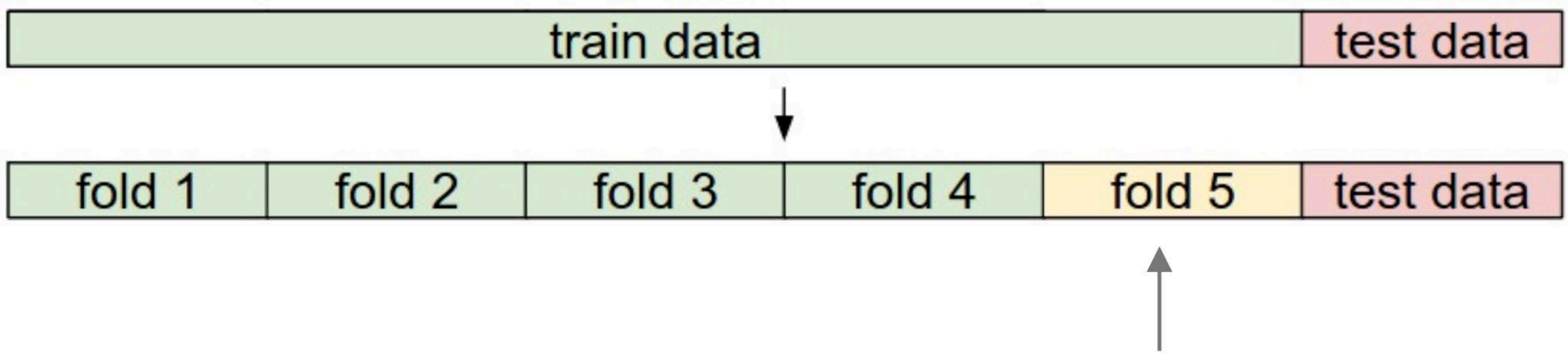
Try out what hyperparameters work best on test set.



Trying out what hyperparameters work best on test set:
Very bad idea. The test set is a proxy for the generalization performance!
Use only **VERY SPARINGLY,** at the end.



Validation

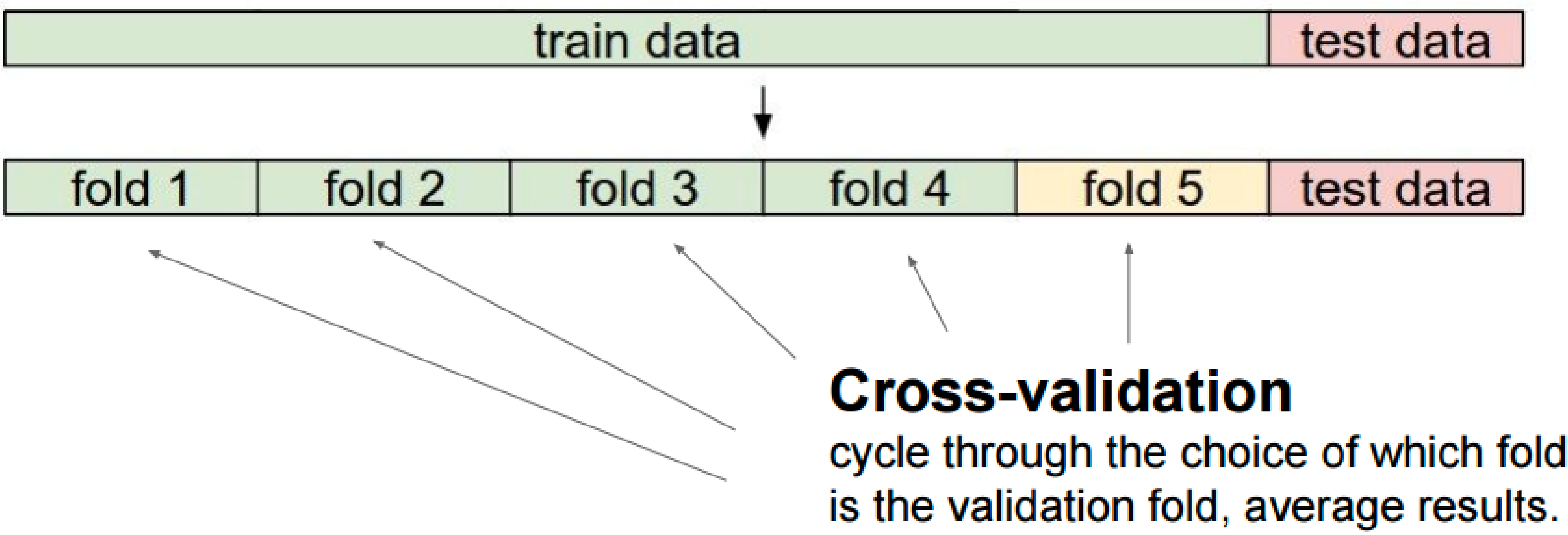


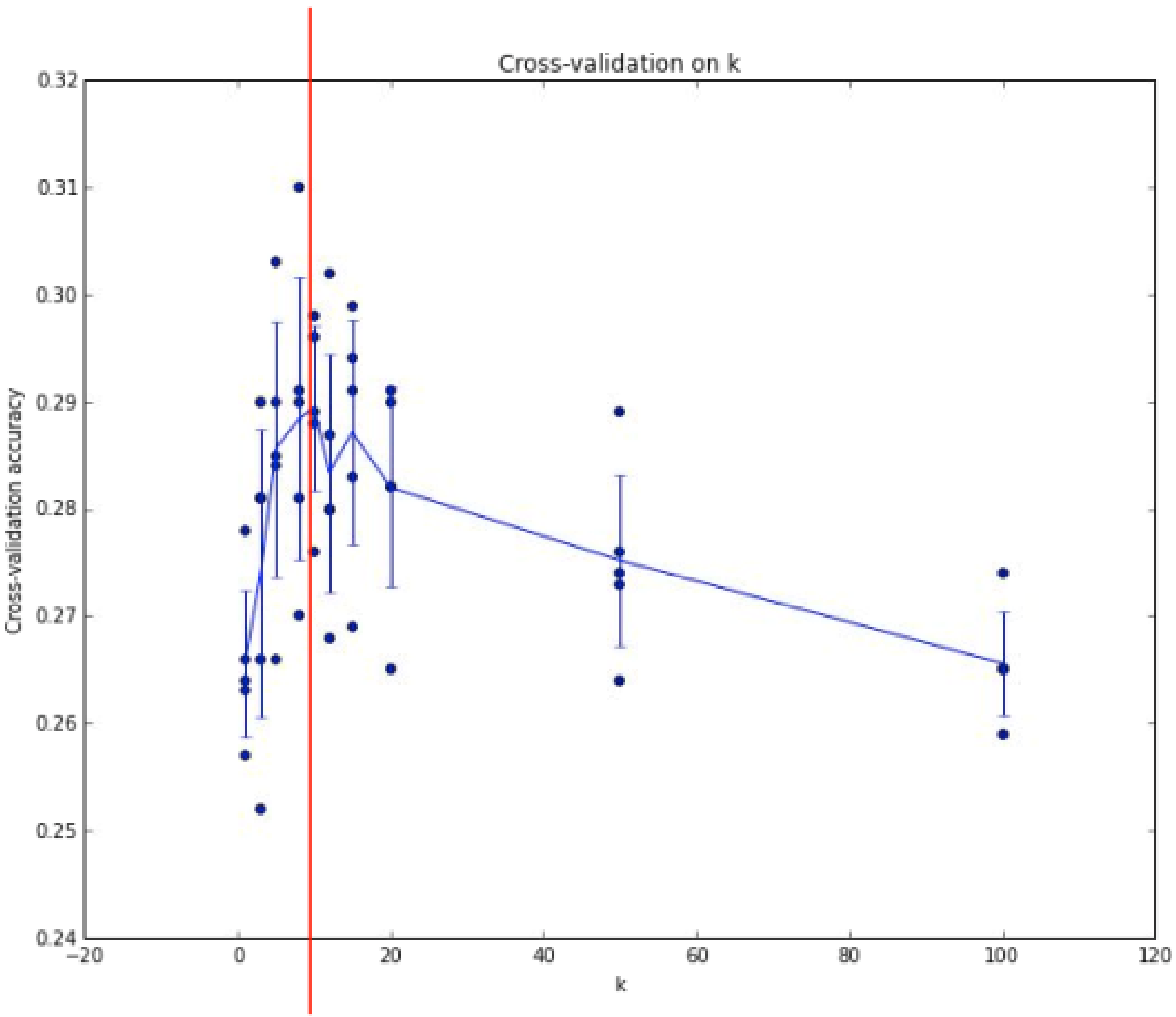
Validation data

use to tune hyperparameters

evaluate on test set ONCE at the end

Cross-validation





Example of
5-fold cross-validation
for the value of **k.**

Each point: single
outcome.

The line goes
through the mean, bars
indicated standard
deviation

(Seems that k ~= 7 works best
for this data)

How to pick hyperparameters?

- Methodology
  - Train and test
  - Train, validate, test

- Train for original model
- Validate to find hyperparameters
- Test to understand generalizability

Pros

- simple yet effective

Cons

- search is expensive (can be sped-up)
- storage requirements
- difficulties with high-dimensional data

kNN -- Complexity and Storage

- N training images, M test images

- Training: O(1)
- Testing: O(MN)

- Hmm…
  – Normally need the opposite
  – Slow training (ok), fast testing (necessary)