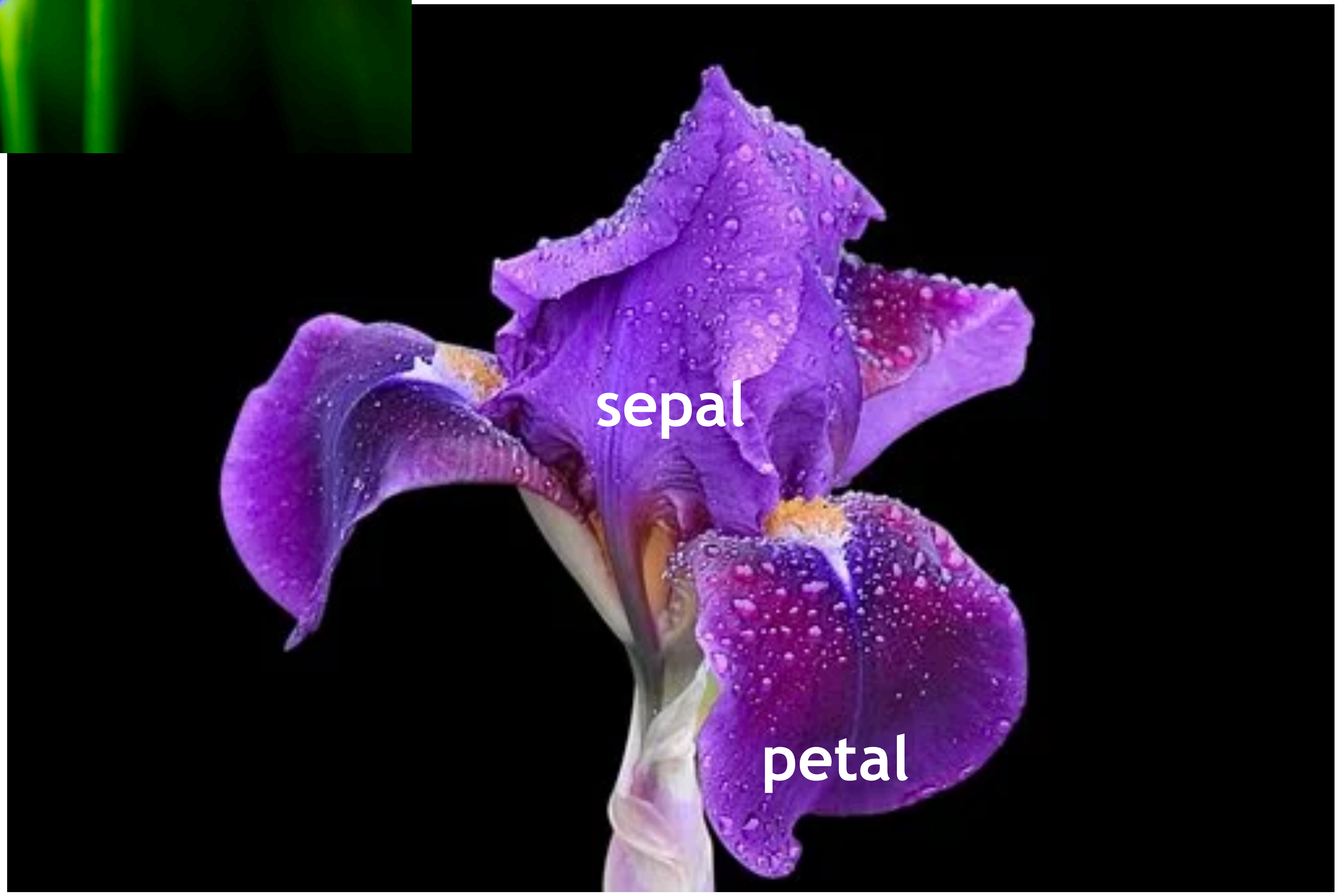
k-NN, model selection

10-701 Introduction to Machine Learning
Geoff Gordon and Pradeep Ravikumar

(thanks to Matt Gormley and Henry Chai for some content)

Continuous features

We’ve been looking mostly at discrete features so far, but many problems use continuous ones — images, physical measurements, …



Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

M=4

N=7 (here)
N=150 (total)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

N=7 (here)
N=150 (total)

Species	Sepal Length	Sepal Wi
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

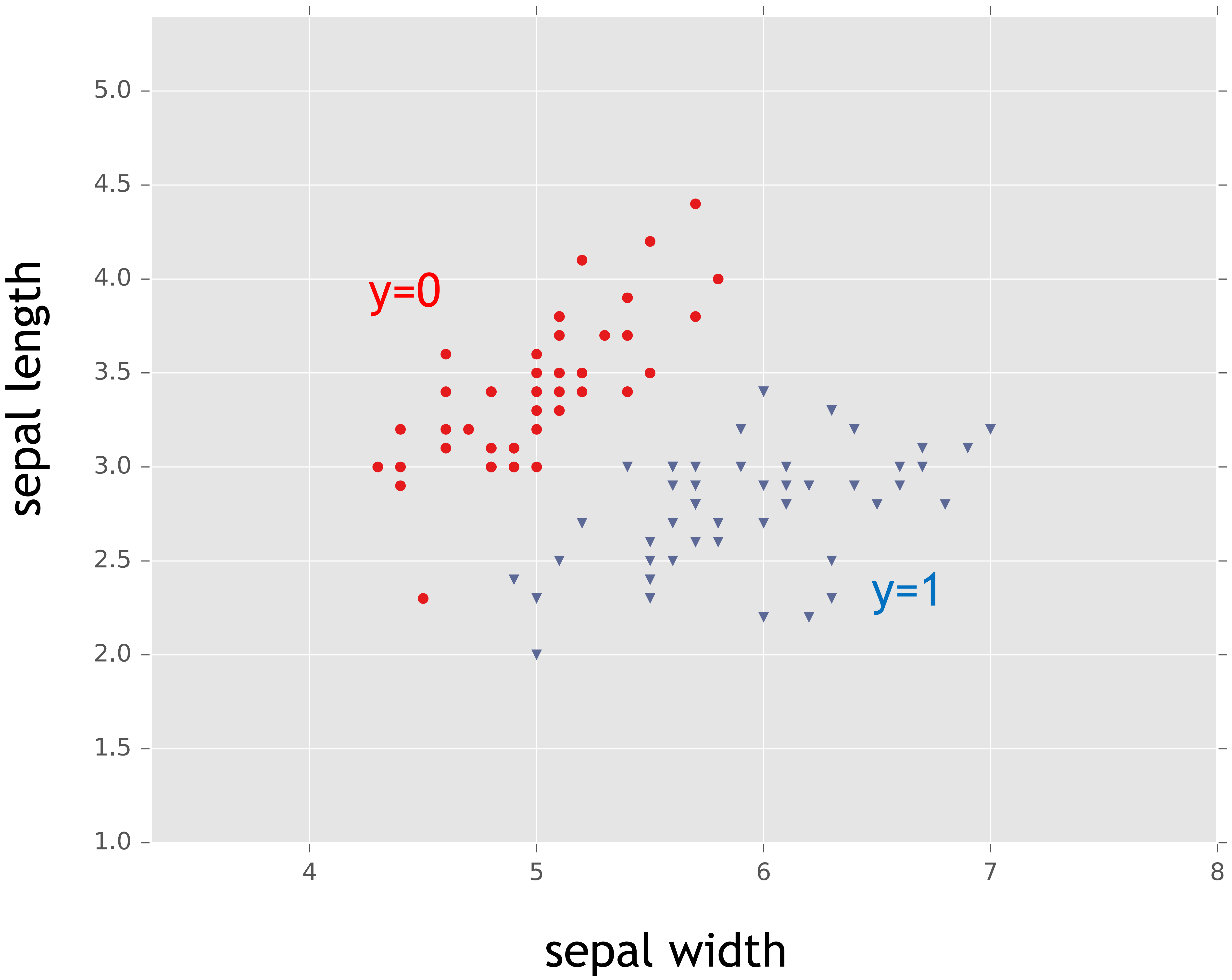
Deleted two of the four features, so that input space is 2D

M=2

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

scatter plot of data



Geoff Gordon

Classification & Real-Valued Features

Def: Hypothesis (aka Decision Rule) for Binary Classification w/ real features

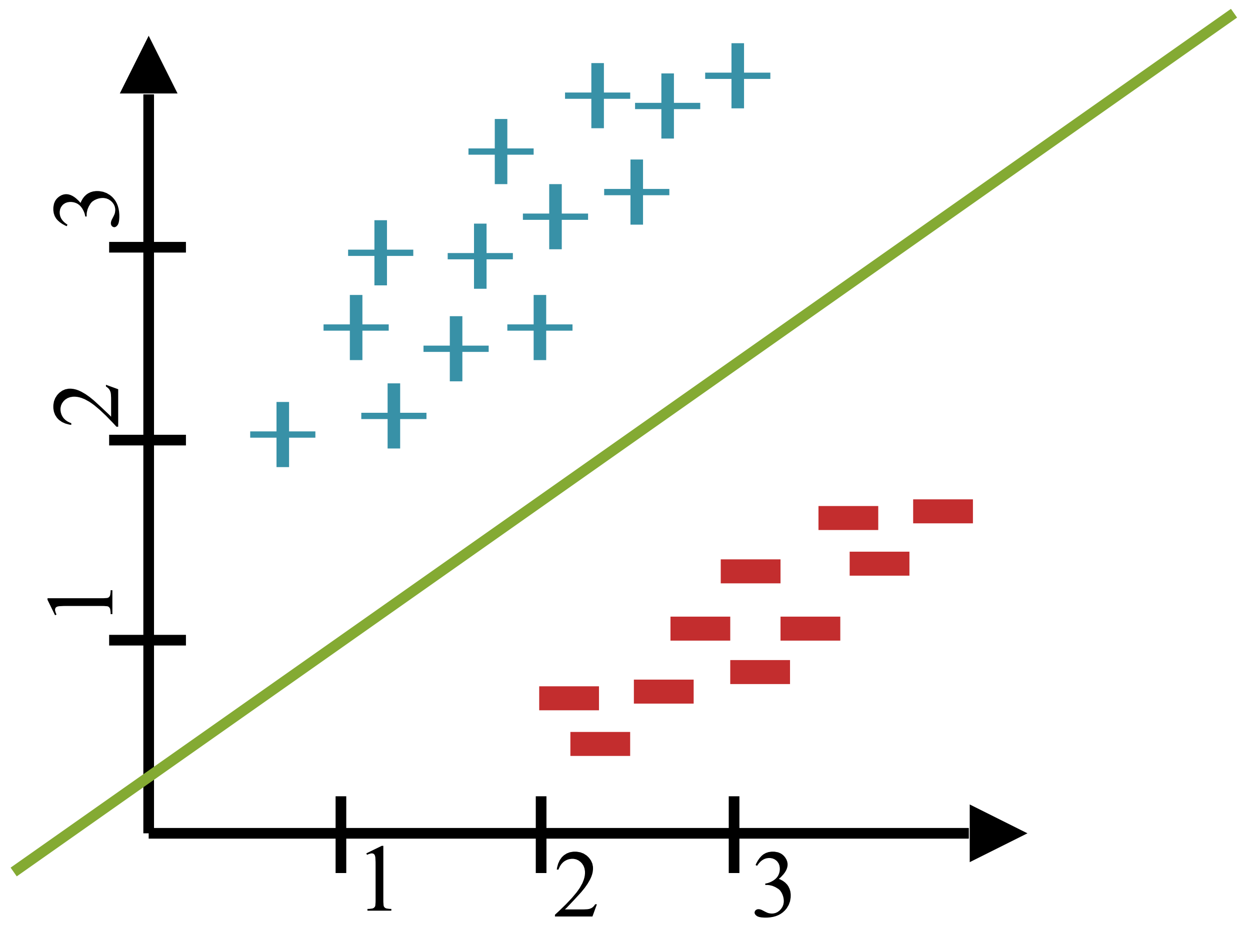
$h : \mathbb{R}^M \rightarrow \{ + , - \}$

Train time: learn h by looking only at N training examples

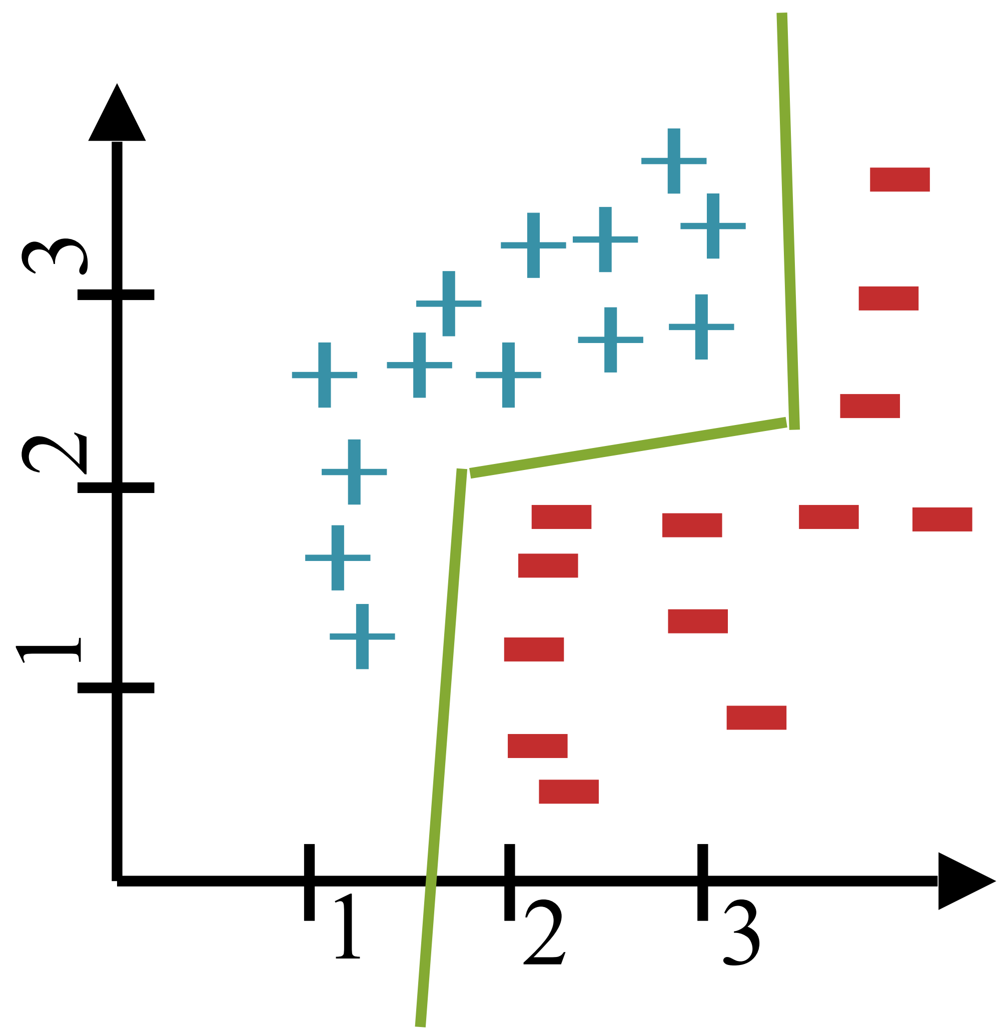
Test time: Given new features $\mathbf{x}$, predict $\hat{y} = h(\mathbf{x})$, check $\hat{y} = y$?

Decision boundary = dividing line / curve / surface / whatever between classes

linear decision boundary



nonlinear decision boundary



Classification & Real-Valued Features

Def: Hypothesis (aka Decision Rule) for Binary Classification w/ real features

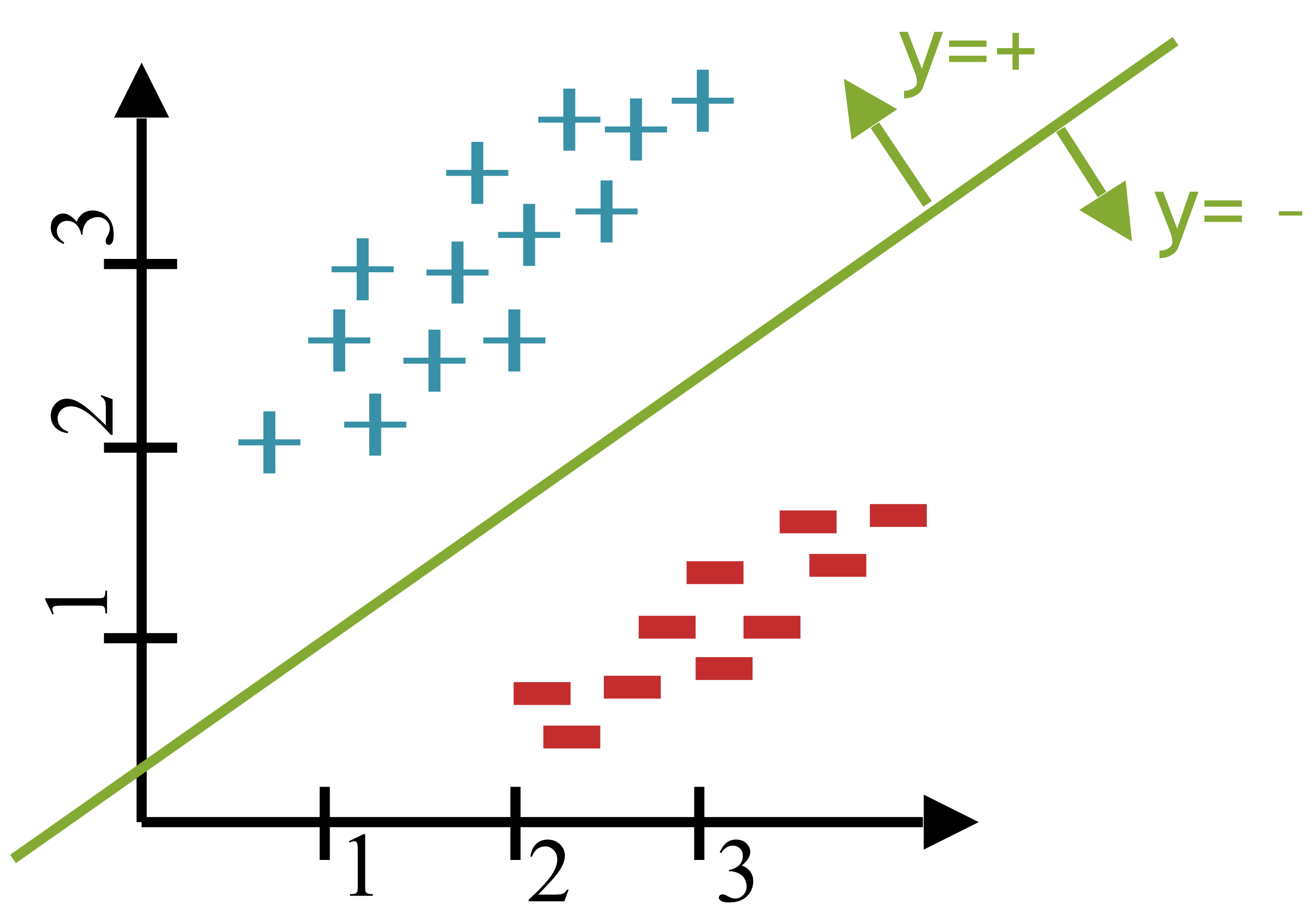
$h : \mathbb{R}^M \rightarrow \{ + , - \}$

Train time: learn h by looking only at N training examples

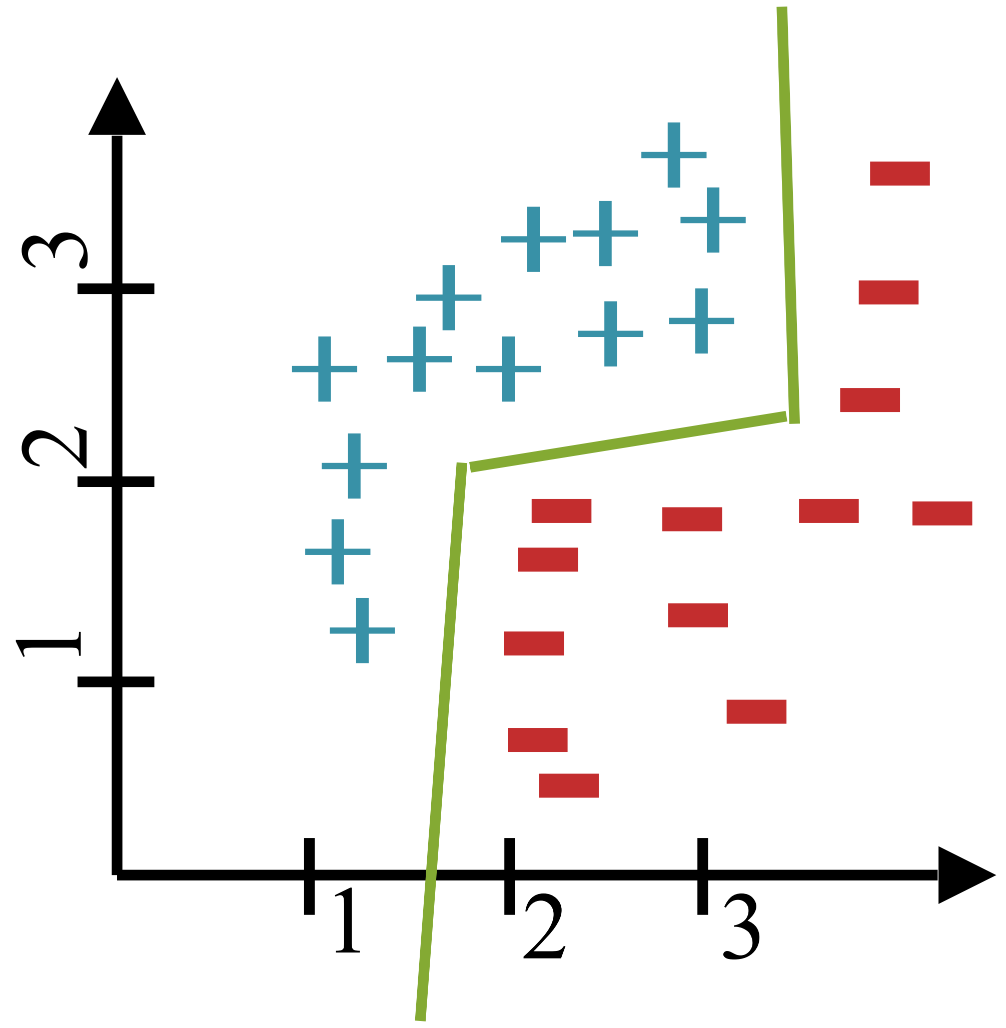
Test time: Given new features $\mathbf{x}$, predict $\hat{y} = h(\mathbf{x})$, check $\hat{y} = y$?

Decision boundary = dividing line / curve / surface / whatever between classes

linear decision boundary



nonlinear decision boundary



Classification & Real-Valued Features

Def: Hypothesis (aka Decision Rule) for Binary Classification w/ real features

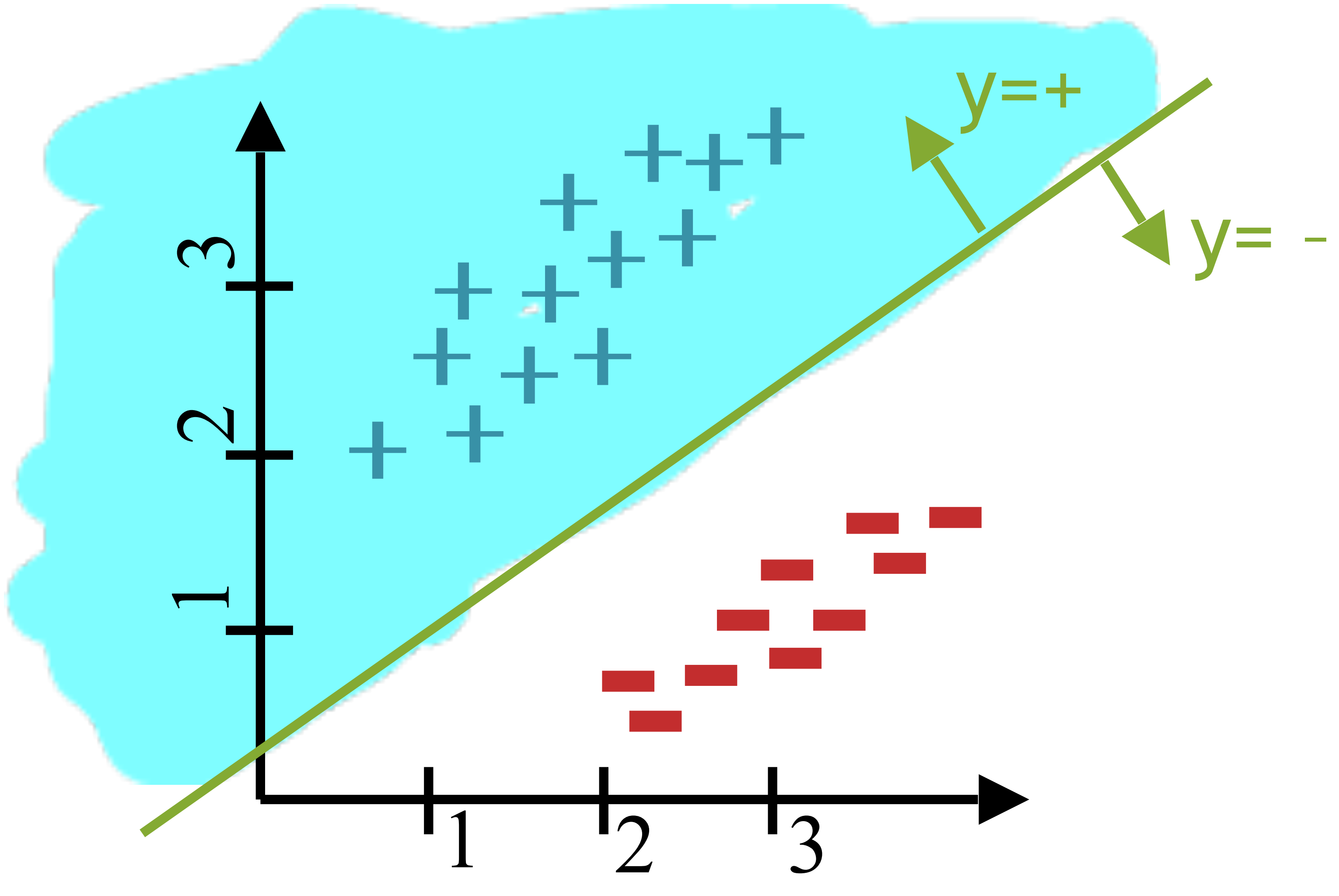
$h : \mathbb{R}^M \rightarrow \{ + , - \}$

Train time: learn h by looking only at N training examples

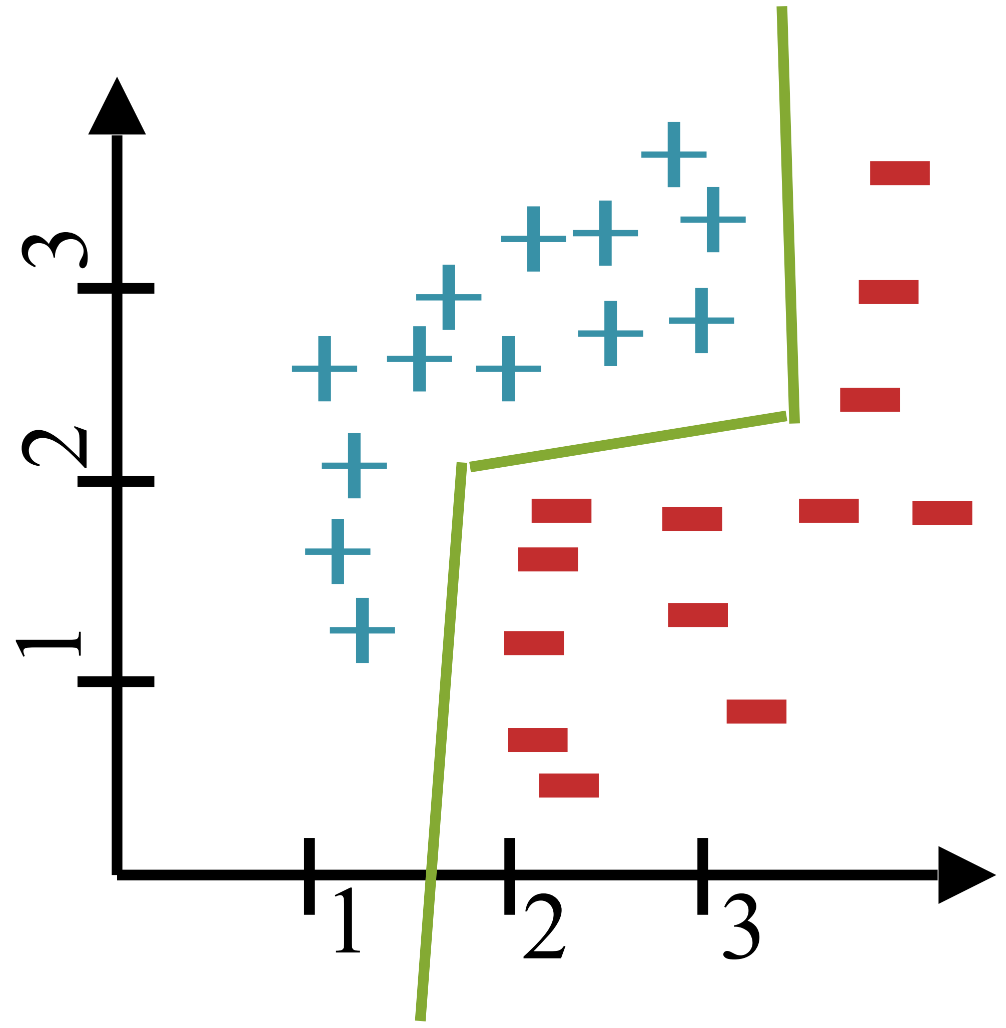
Test time: Given new features $\mathbf{x}$, predict $\hat{y} = h(\mathbf{x})$, check $\hat{y} = y$?

Decision boundary = dividing line / curve / surface / whatever between classes

linear decision boundary



nonlinear decision boundary



Classification & Real-Valued Features

Def: Hypothesis (aka Decision Rule) for Binary Classification w/ real features

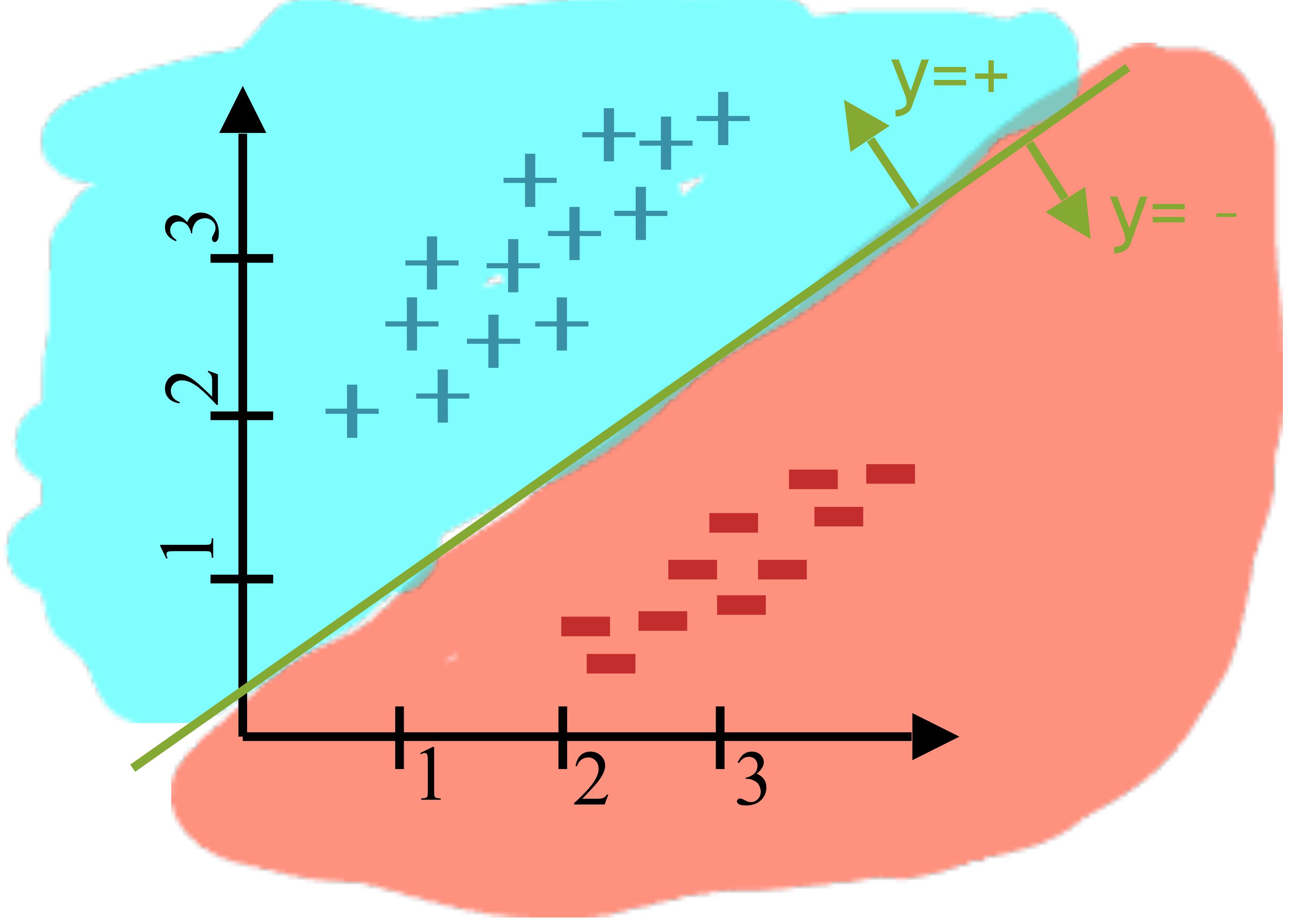
$h : \mathbb{R}^M \to \{ + , - \}$

Train time: learn h by looking only at N training examples

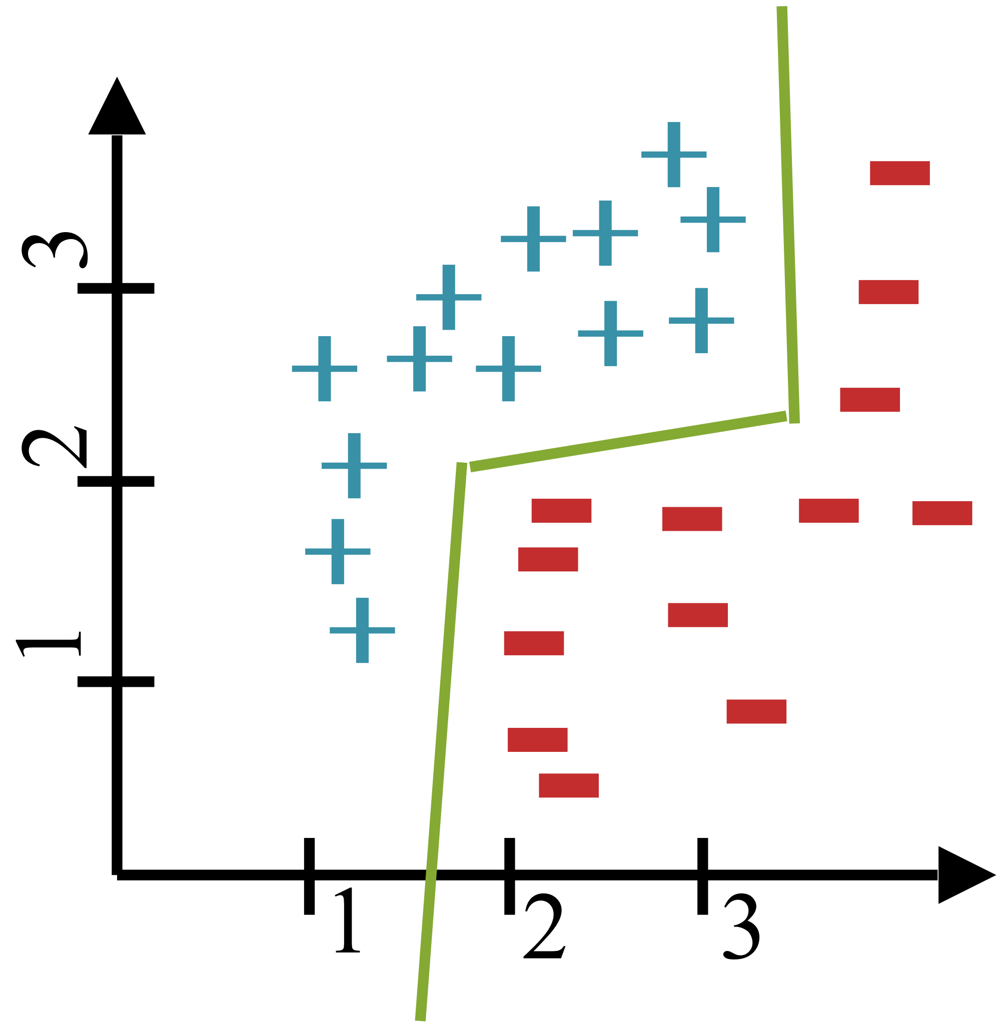
Test time: Given new features $\mathbf{x}$, predict $\hat{y} = h(\mathbf{x})$, check $\hat{y} = y$?

Decision boundary = dividing line / curve / surface / whatever between classes

linear decision boundary



nonlinear decision boundary



Classification & Real-Valued Features

Def: Hypothesis (aka Decision Rule) for Binary Classification w/ real features

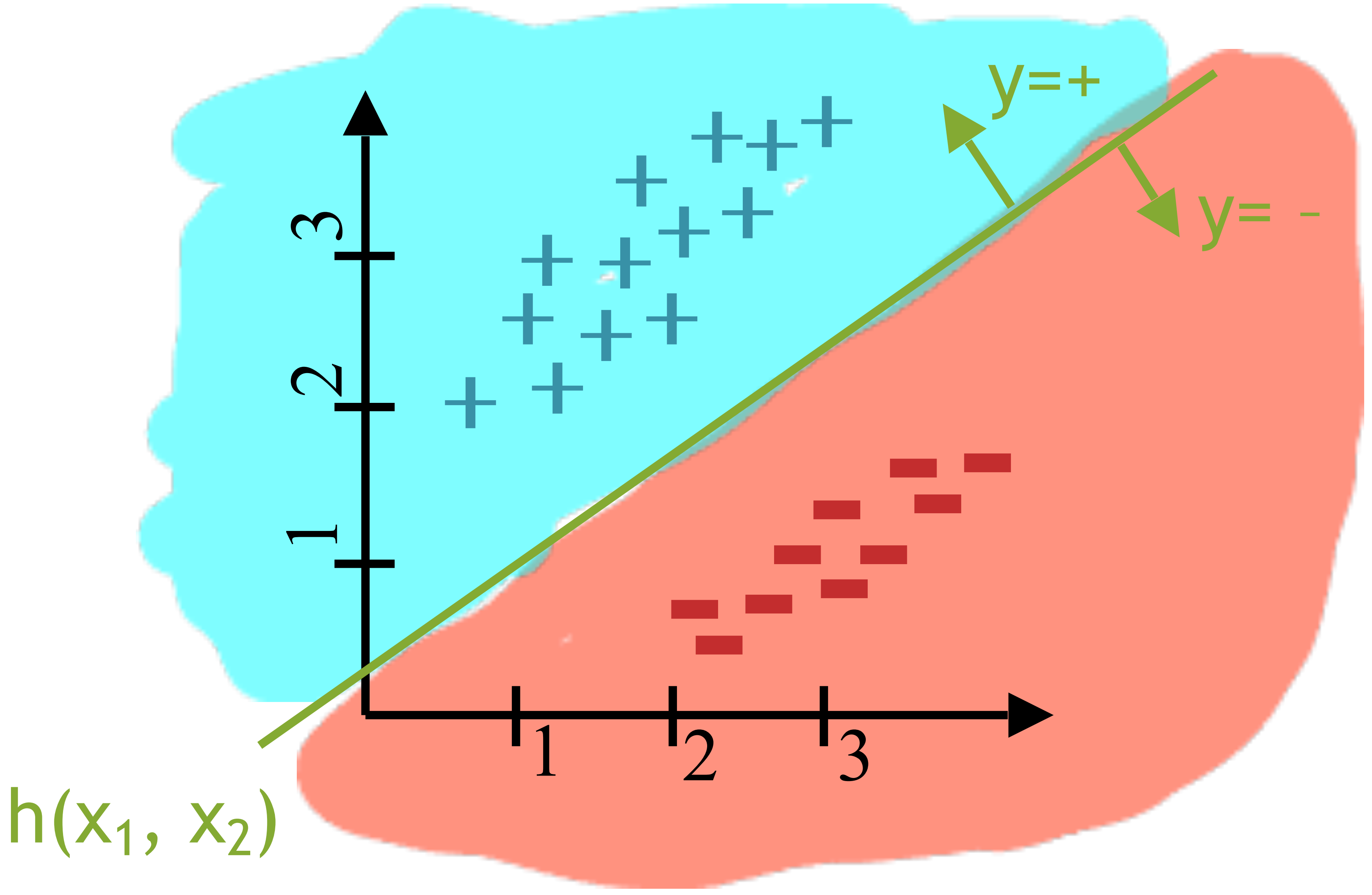
$h : \mathbb{R}^M \rightarrow \{ + , - \}$

Train time: learn h by looking only at N training examples

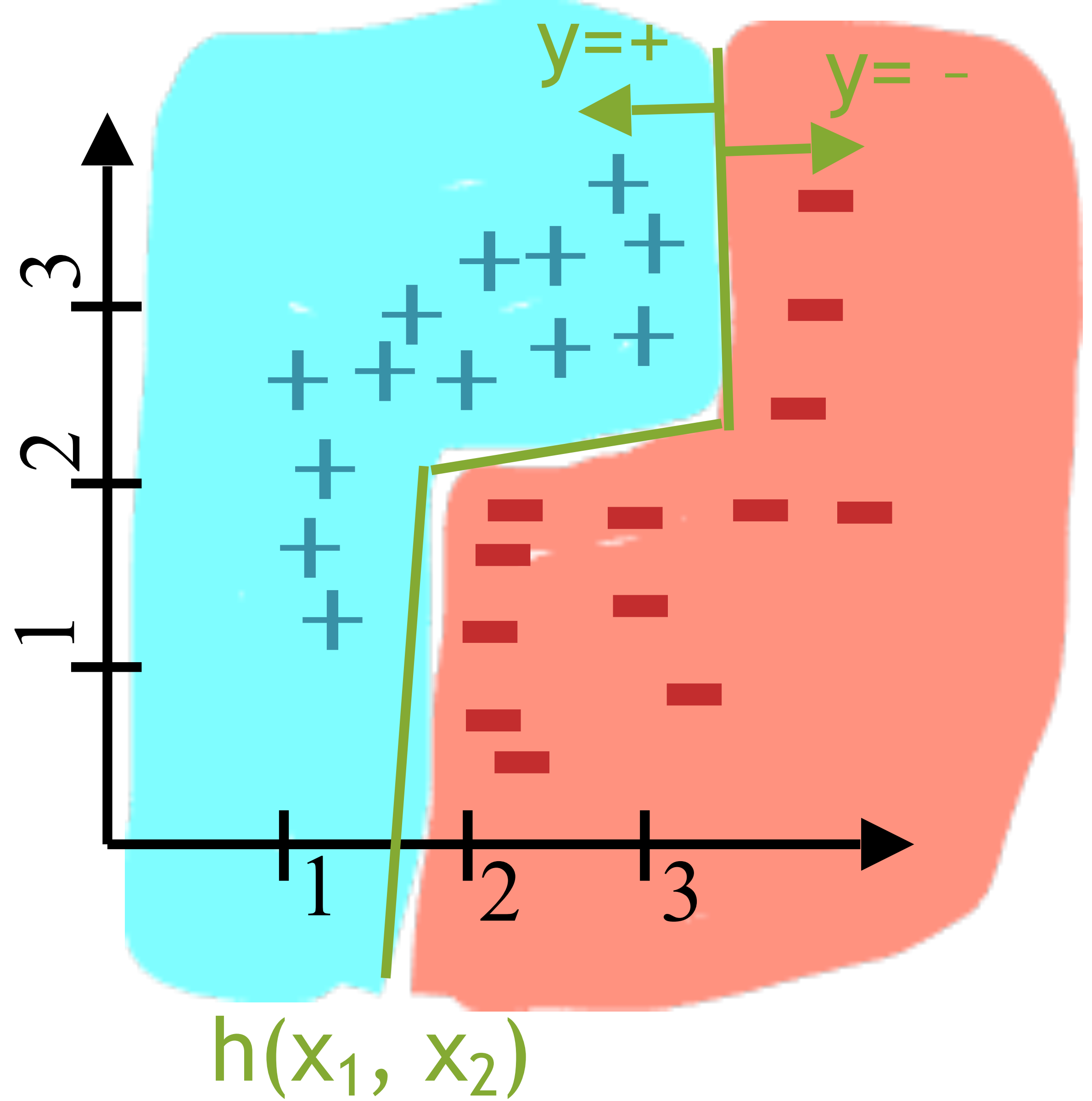
Test time: Given new features $\mathbf{x}$, predict $\hat{y} = h(\mathbf{x})$, check $\hat{y} = y$?

Decision boundary = dividing line / curve / surface / whatever between classes

linear decision boundary



nonlinear decision boundary



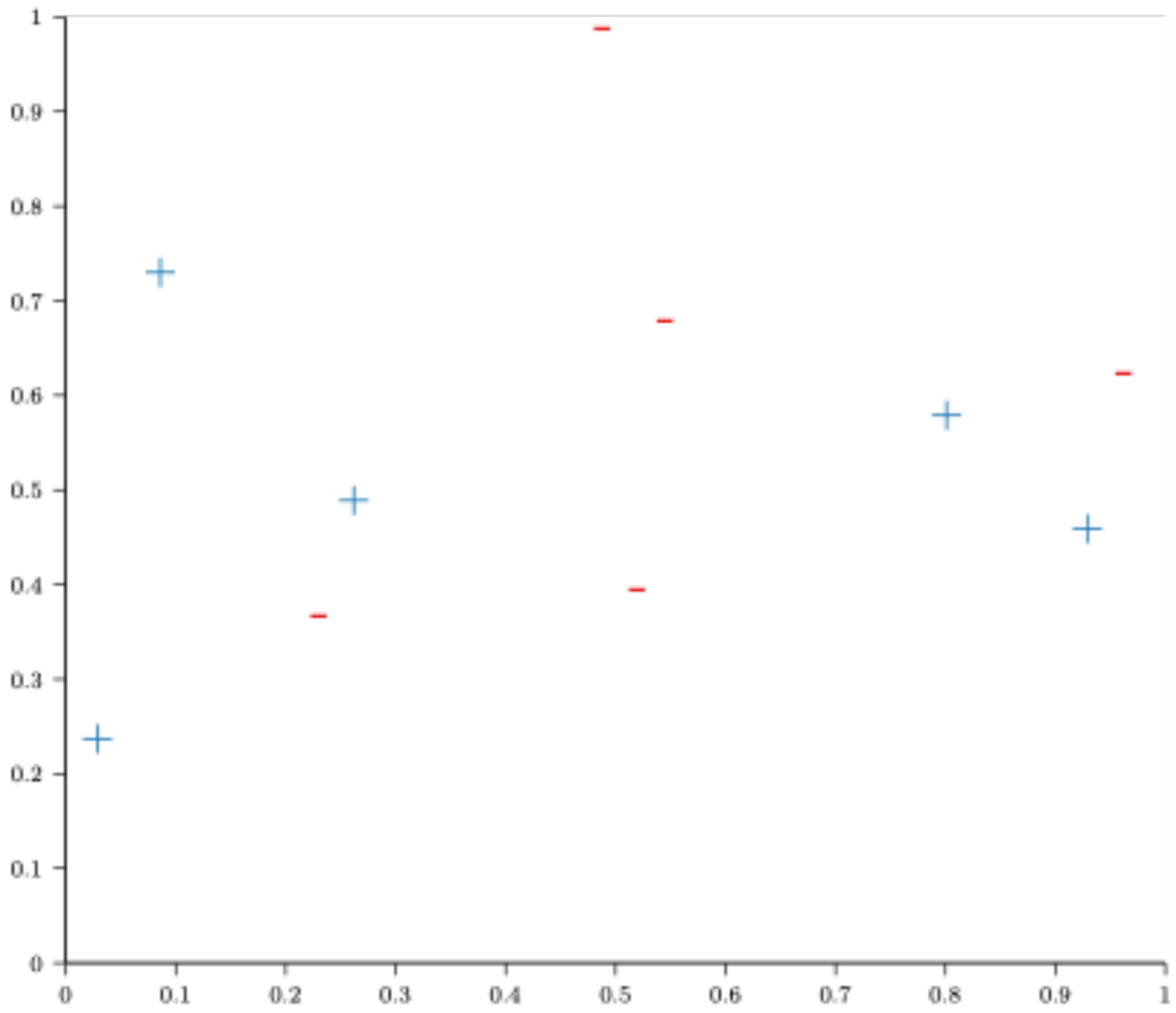
Neighbors

- Notice: predictions for nearby points tend to be similar
- Motivates ***nearest-neighbor*** methods: predict based on nearby training points
  - often not SOTA accuracy
  - but highly interpretable
- NN methods form the basis for lots of AI systems
  - e.g., search or recommendation systems
  - don't just need to predict a label, but return an item (a webpage, a product to buy) — prediction is useless without the item

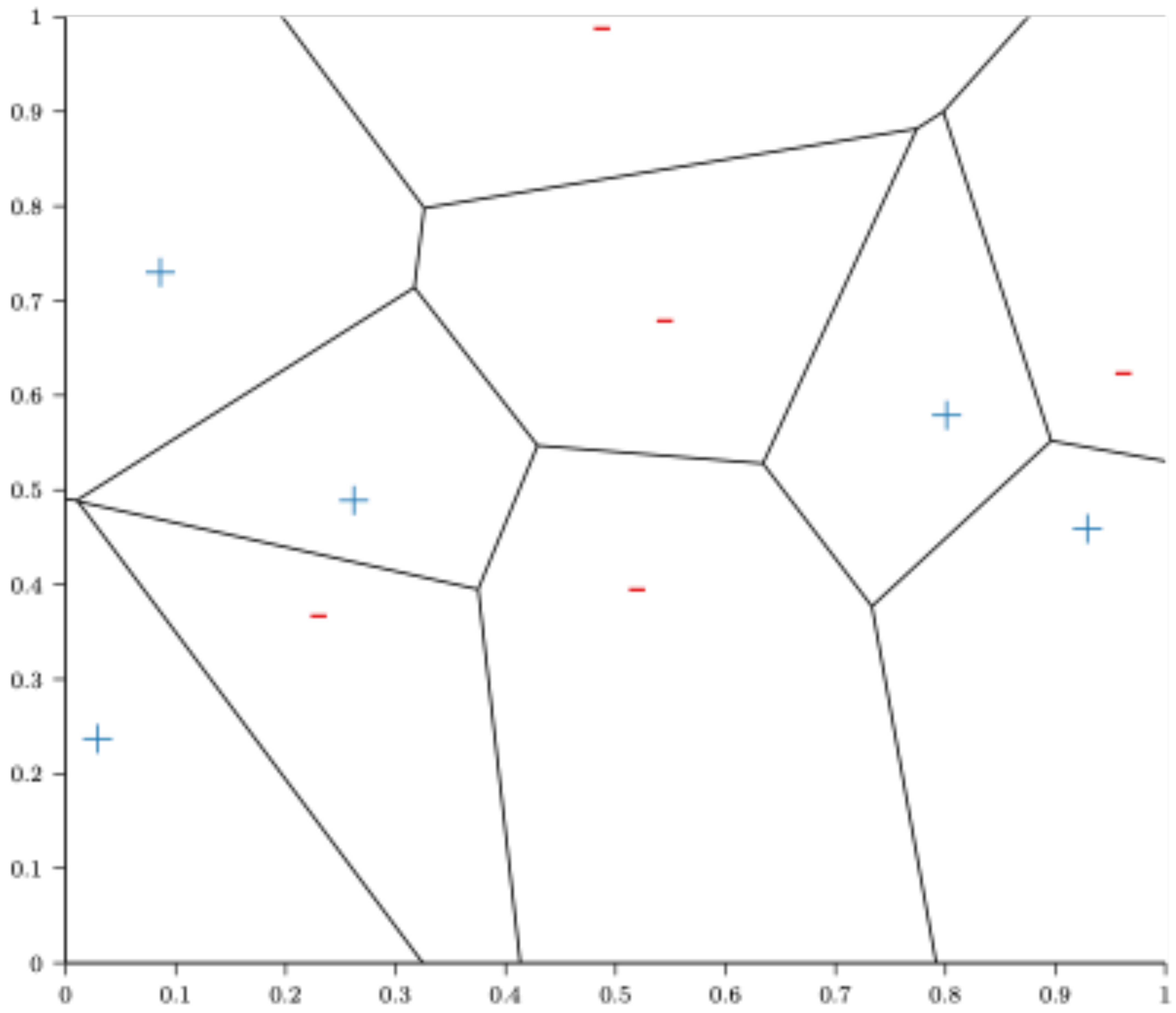
k-nearest-neighbors

- Simplest neighborhood-based method: k-NN
- Train: just store the dataset!
  - no parameters to fit: k-NN is a ***nonparametric*** method
- Test: look up the k nearest training points
  - for classification, predict the most common label (like *majority* classifier)
  - for regression, predict the average label
- 1-NN is just called ***nearest neighbor***

Nearest Neighbor: Example

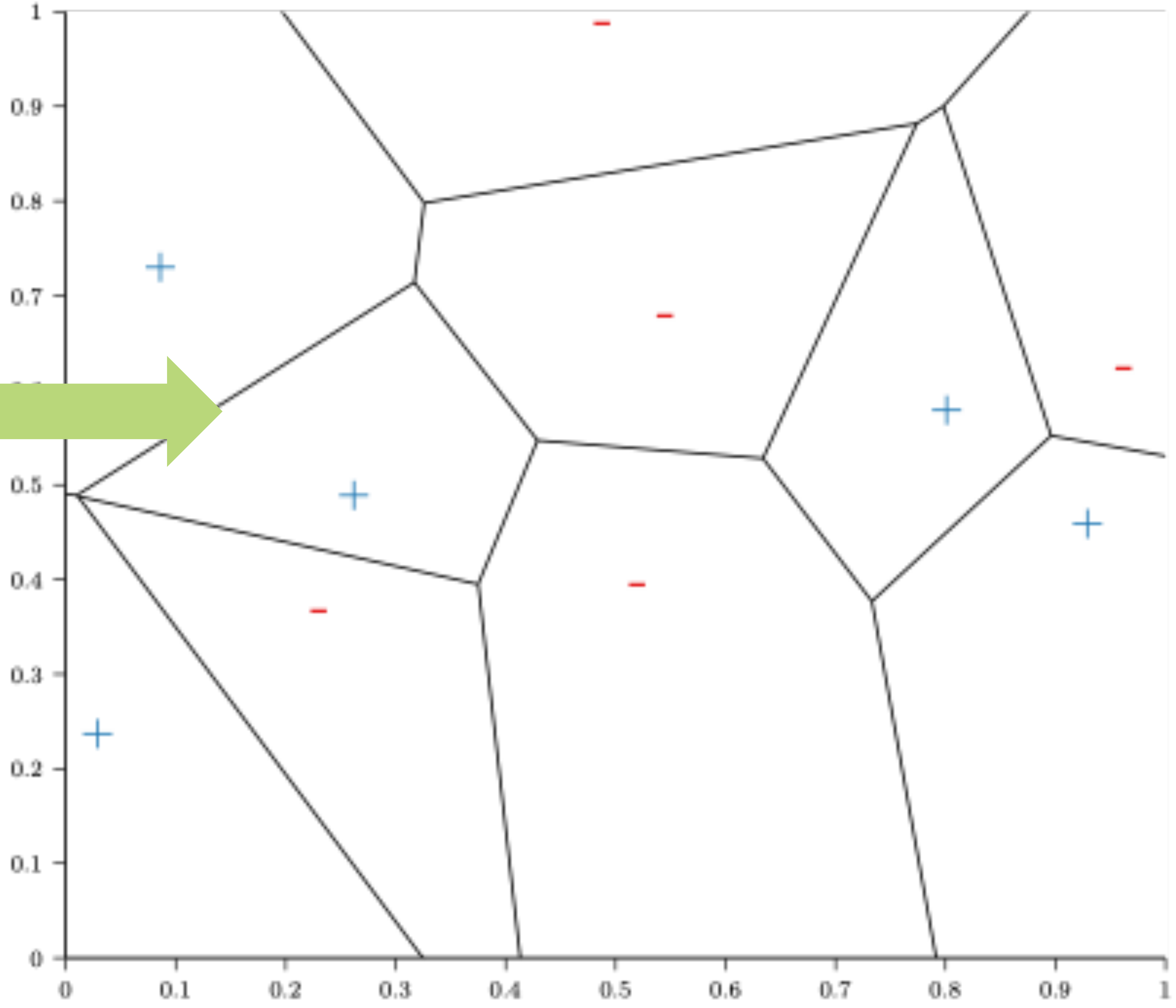


Nearest Neighbor: Example

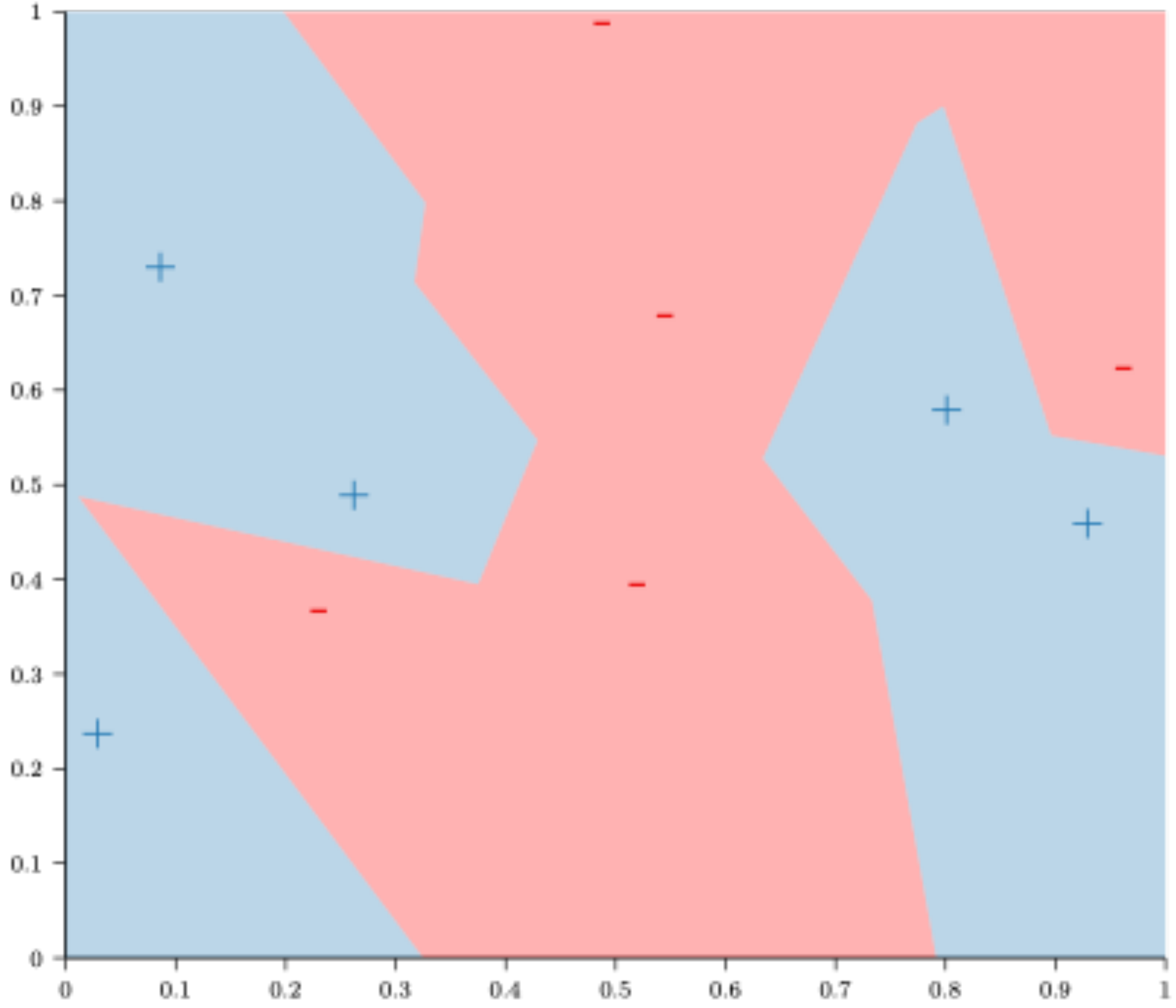


Nearest Neighbor: Example

- This is a **Voronoi diagram**
- Each **cell** contain one of our training examples
- **All points within a cell** are **closer** to that training example, than to any other training example
- **Points on the Voronoi line segments** are **equidistant** to one or more training examples



Nearest Neighbor: Example



Thought question

- What is the training error of 1-NN?

KNN: Remarks

Distance Functions:

- KNN requires a **distance function**

$$d : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- The most common choice is **Euclidean distance**

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sqrt{\sum_{m=1}^{M} \left(u_m - v_m\right)^2}$$

- But there are other choices (e.g. **Manhattan distance**)

$$d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{m=1}^{M} \left|u_m - v_m\right|$$

KNN: Computational Efficiency

- N training examples, each one w/ M features
- Computational complexity when k=1:

Task	Naive	Smart data structure: ball tree, kd-tree, neighborhood graph *(exact NN)*	Smart data structure *(approximate NN)*
Train	O(MN) or O(1)	O(N log N) if M=1,2; in general O(MN^2) worst case	O(MN log N)
Predict (one test example)	O(MN)	O(log N) if M=1,2; in general O(MN) worst case	O(M log N)

Problem: Exact can be fast for small M, but slow for large M

If approximate is good enough: can still be very fast!

KNN: Theoretical Guarantees

Cover & Hart (1967)

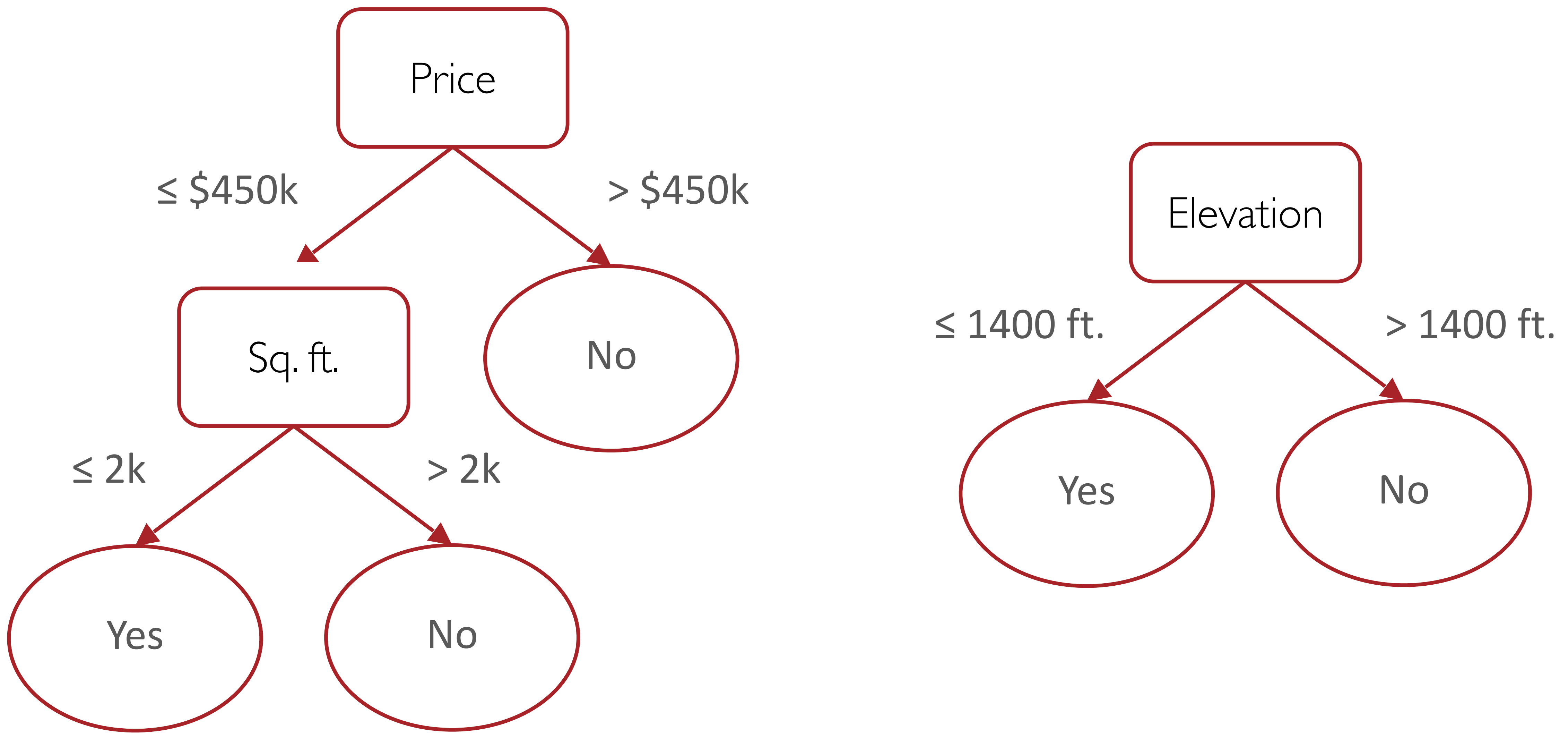
Let h(x) be a Nearest Neighbor (k=1) binary classifier. As the number of training examples N goes to infinity…

$$\text{error}_{\text{true}}(h) < 2 \times \text{Bayes Error Rate}$$

"In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor."

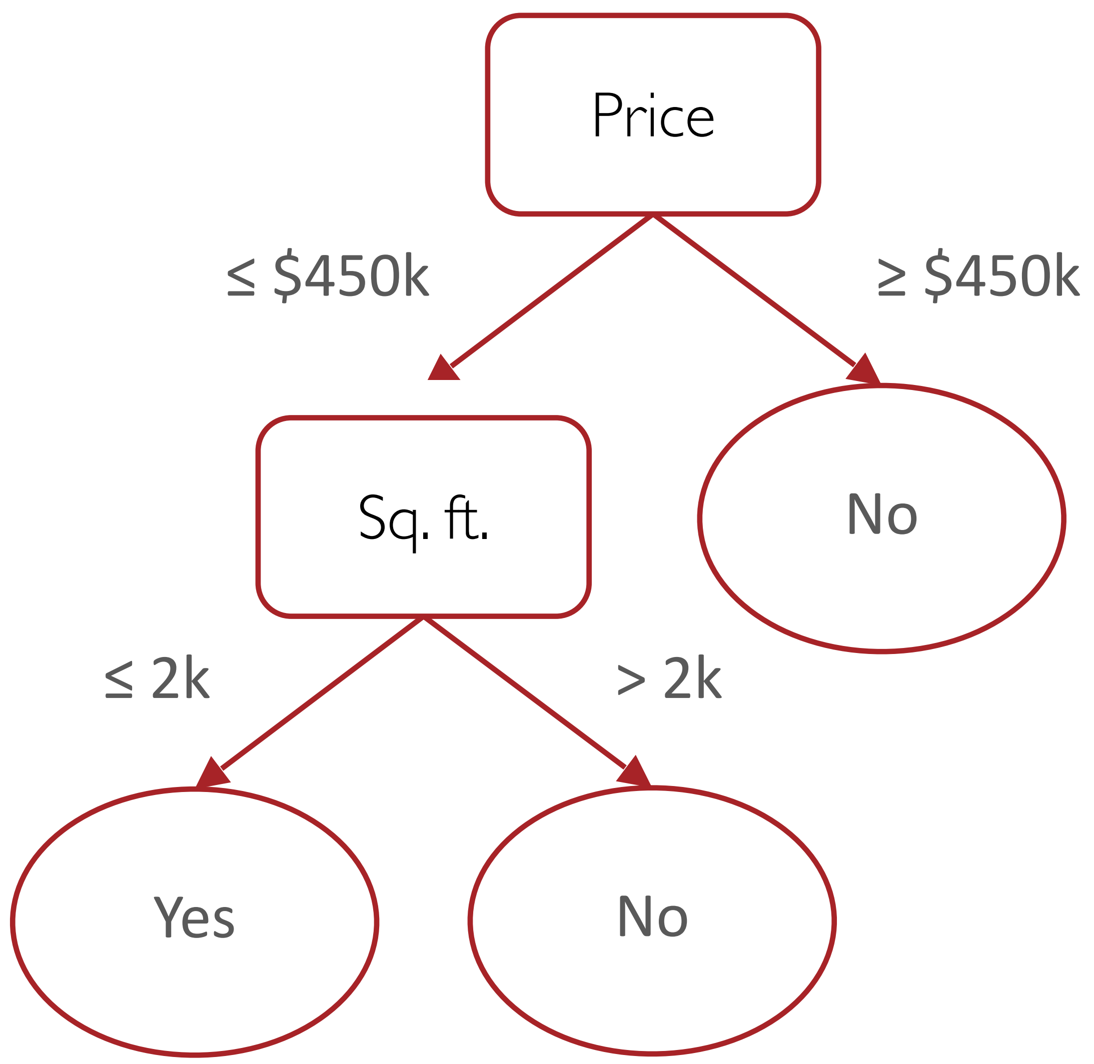
very informally,
Bayes Error Rate =
'the best you could possibly do'

Model selection



- A ***hypothesis space*** is a set of hypotheses (classifiers, regression functions, …)
  - the learning algorithm searches through the hypothesis space to try to find a good hypothesis
- E.g., all decision trees of depth ≤ 3 with splits on attributes { square feet, price, elevation, rooms }

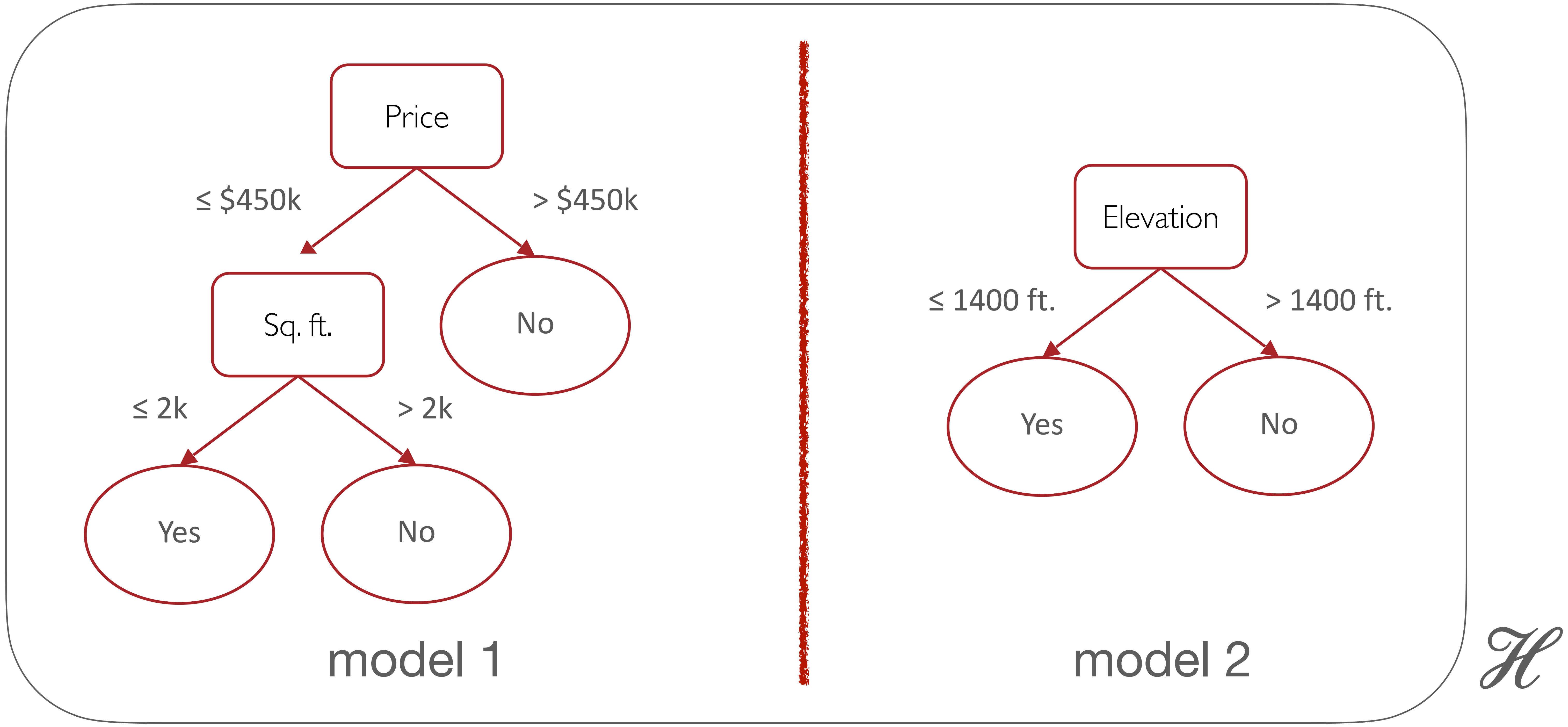
Model selection



- If the hypotheses all share a similar form (e.g., decision trees of the same structure), the hypothesis space is often called a ***model***
  - different models are distinguished by (often real-valued) ***parameters****;* so it is the learner's job to fit parameters
  - here, parameters are split features & split thresholds
- Some flexibility: could say different split features → different model, parameters are just thresholds

Geoff Gordon

Model selection



- More generally, may want to search over several models
  - hypothesis space is the *union* of the models
  - and *model selection* is the process of choosing
- Often, two nested loops
  - inner (learning) loop fits parameters of one model
  - outer (model selection) loop chooses which one

Hyper-parameters

- Often, we call the distinguishing factors among models ***hyperparameters***
  - e.g., depth of decision tree, k in k-NN, or how many attention heads in transformer
  - in this case, outer loop is ***hyperparameter tuning***
- Not just model structure: learning algorithm can have hyperparameters
  - e.g., mutual information vs. Gini split criterion for decision tree learner
  - e.g., momentum parameter for deep net learning
  - e.g., early stopping for either decision trees or deep nets

Why model selection?

max of k dice

1	2	3	4	5
3.5	4.47	4.96	5.24	5.43

amount of selection bias

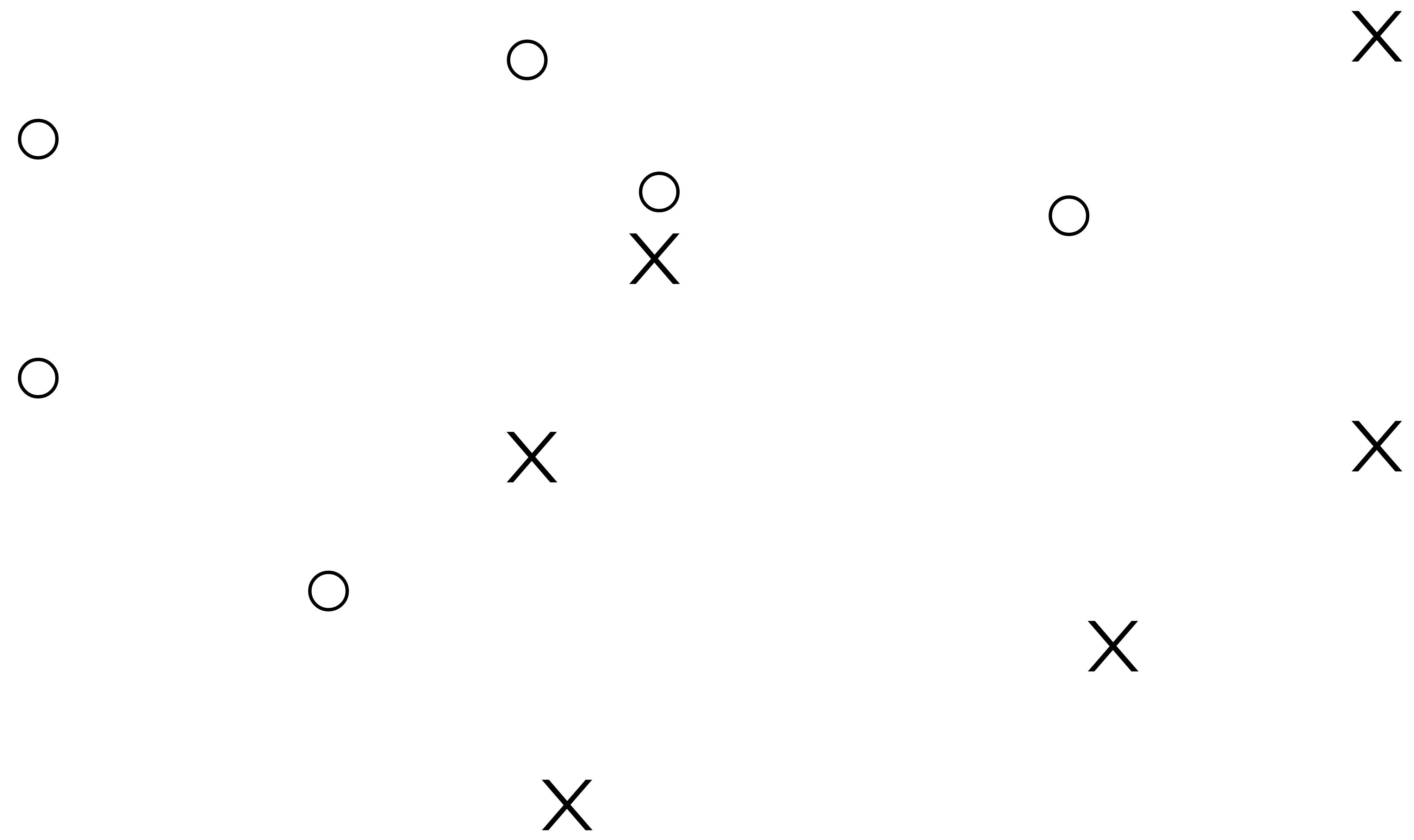
1	2	3	4	5
0	0.97	1.46	1.74	1.93

- Sometimes we just don't know model structure
- Often, a tradeoff between simplicity and expressivity
  - i.e., underfitting vs. overfitting
- Complex model class might contain a model that's closer to the truth — but overfitting becomes more of a concern

Geoff Gordon

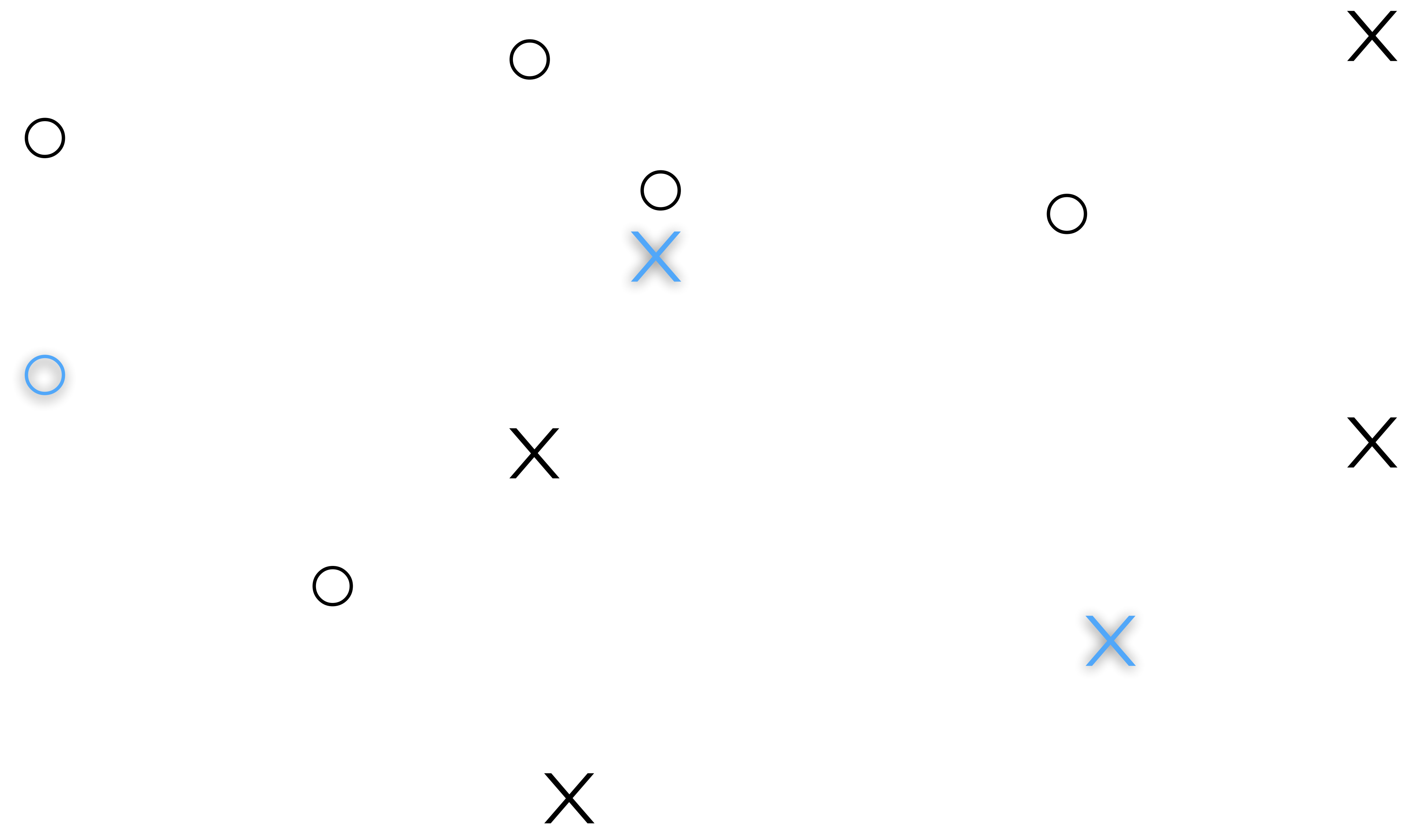
How can we evaluate a single model?

- Hold-out set (aka validation set)



How can we evaluate a single model?

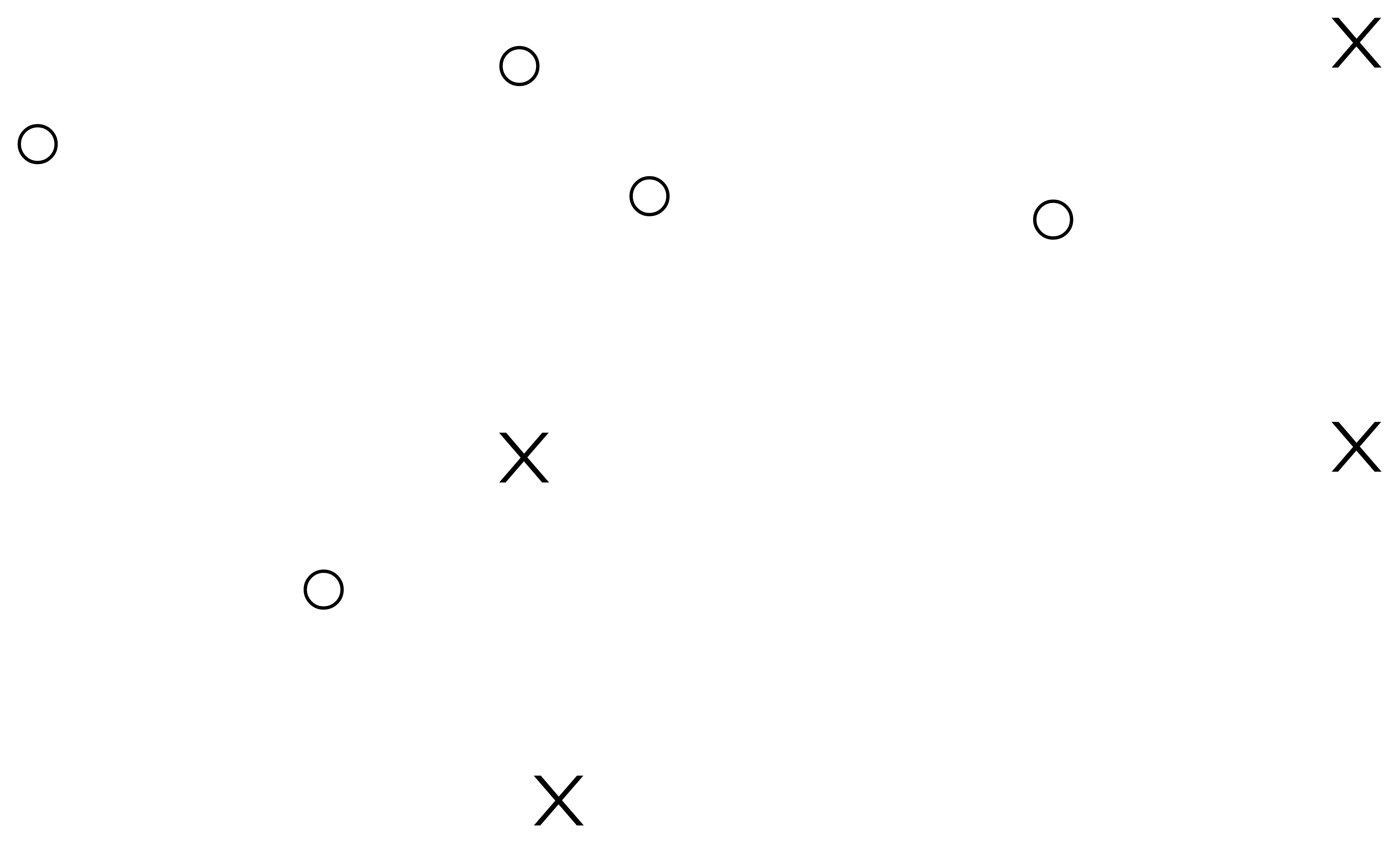
- Hold-out set (aka validation set)



remove hold-out group, fit on rest

How can we evaluate a single model?

- Hold-out set (aka validation set)

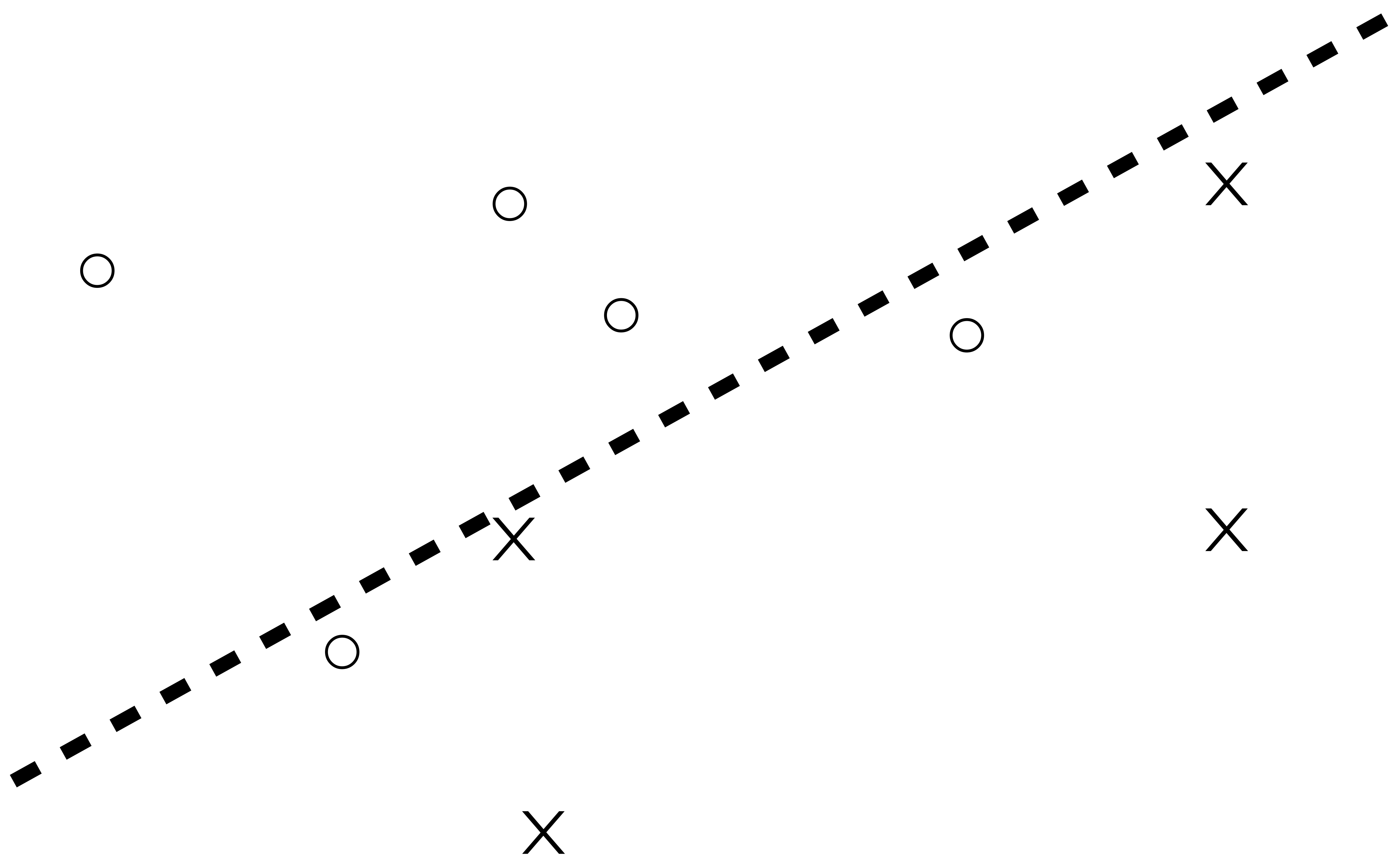


remove hold-out group, fit on rest

Geoff Gordon

How can we evaluate a single model?

- Hold-out set (aka validation set)

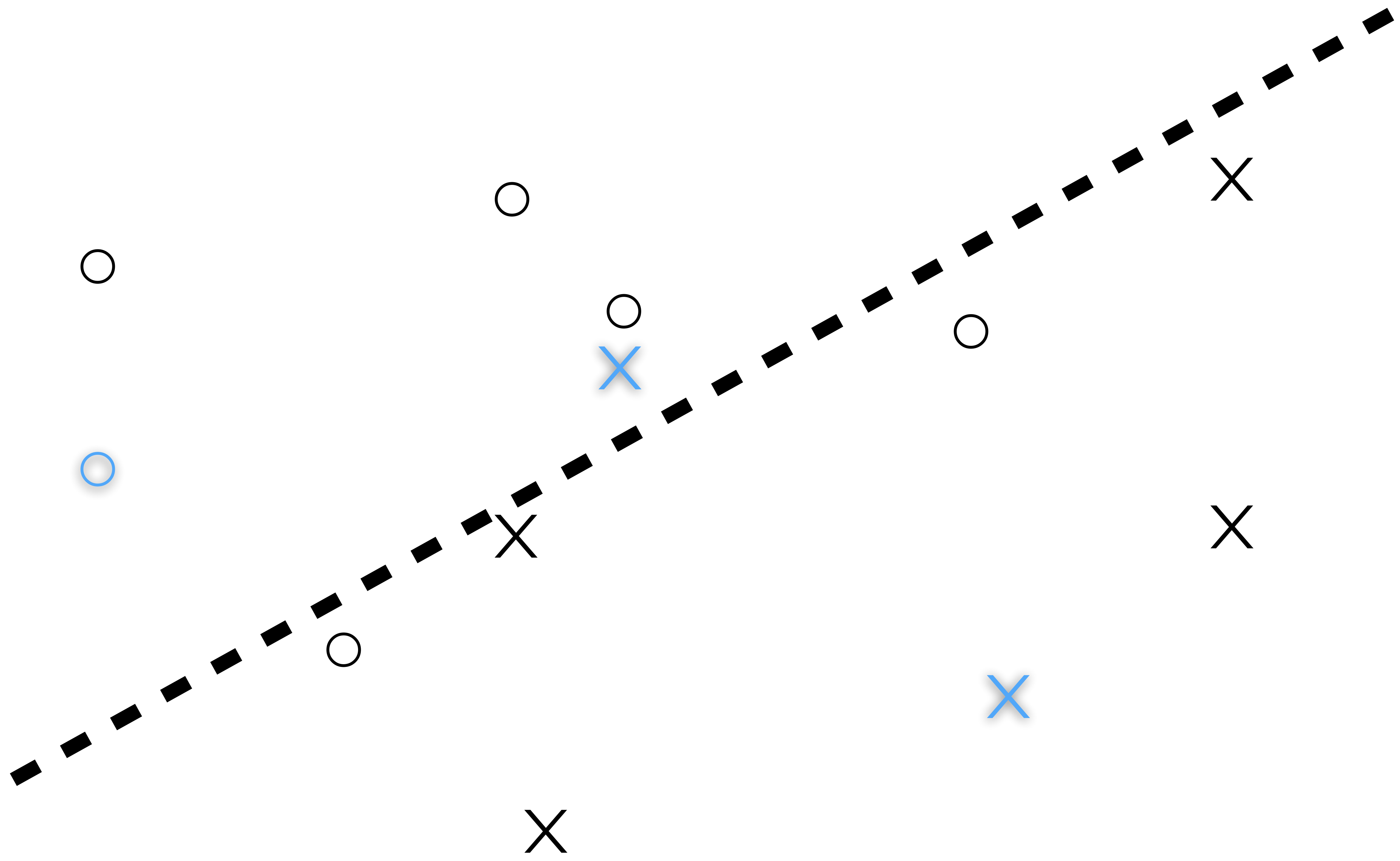


remove hold-out group, fit on rest

Geoff Gordon

How can we evaluate a single model?

- Hold-out set (aka validation set)

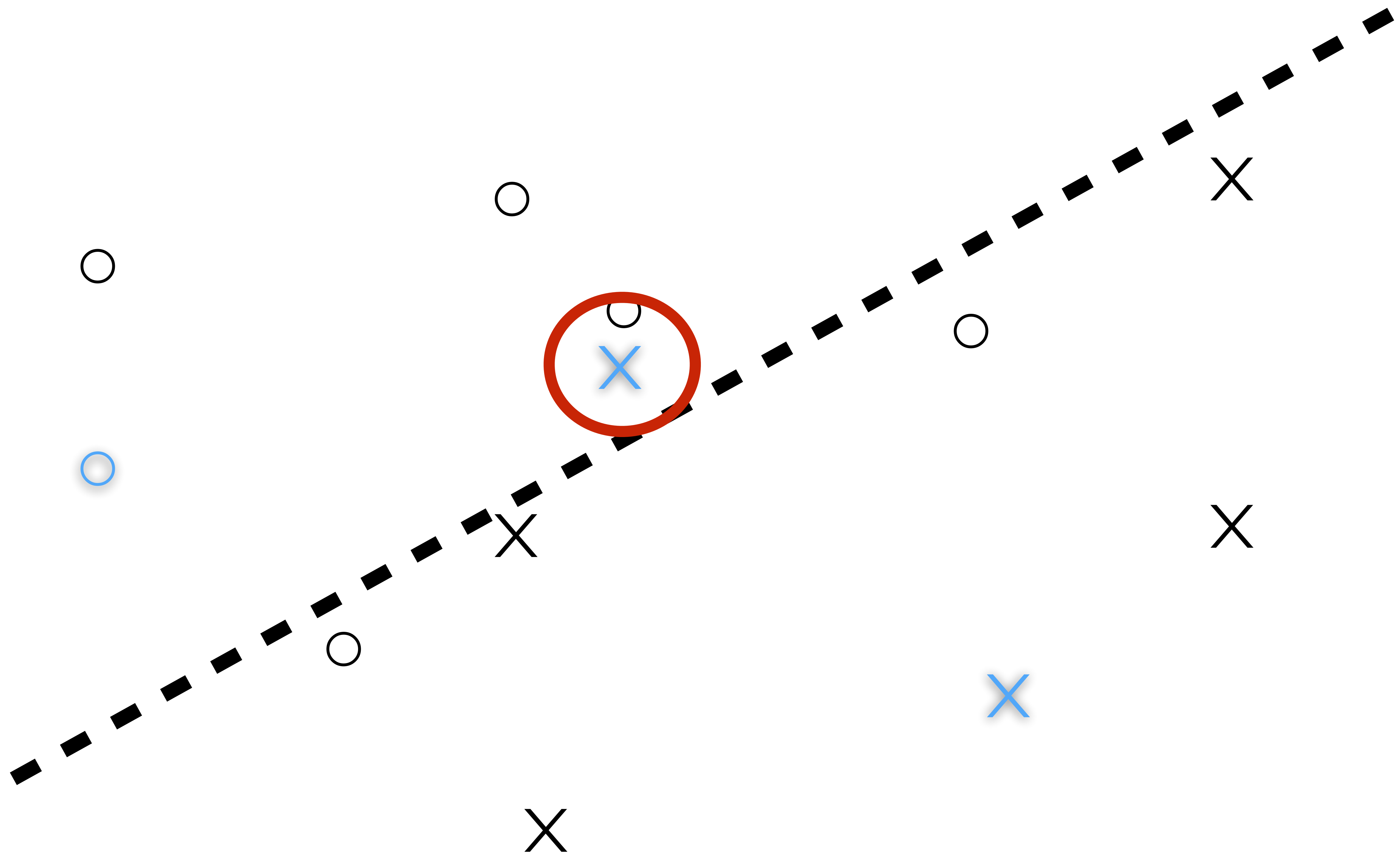


add back in hold-out group, compute error

Geoff Gordon

How can we evaluate a single model?

- Hold-out set (aka validation set)

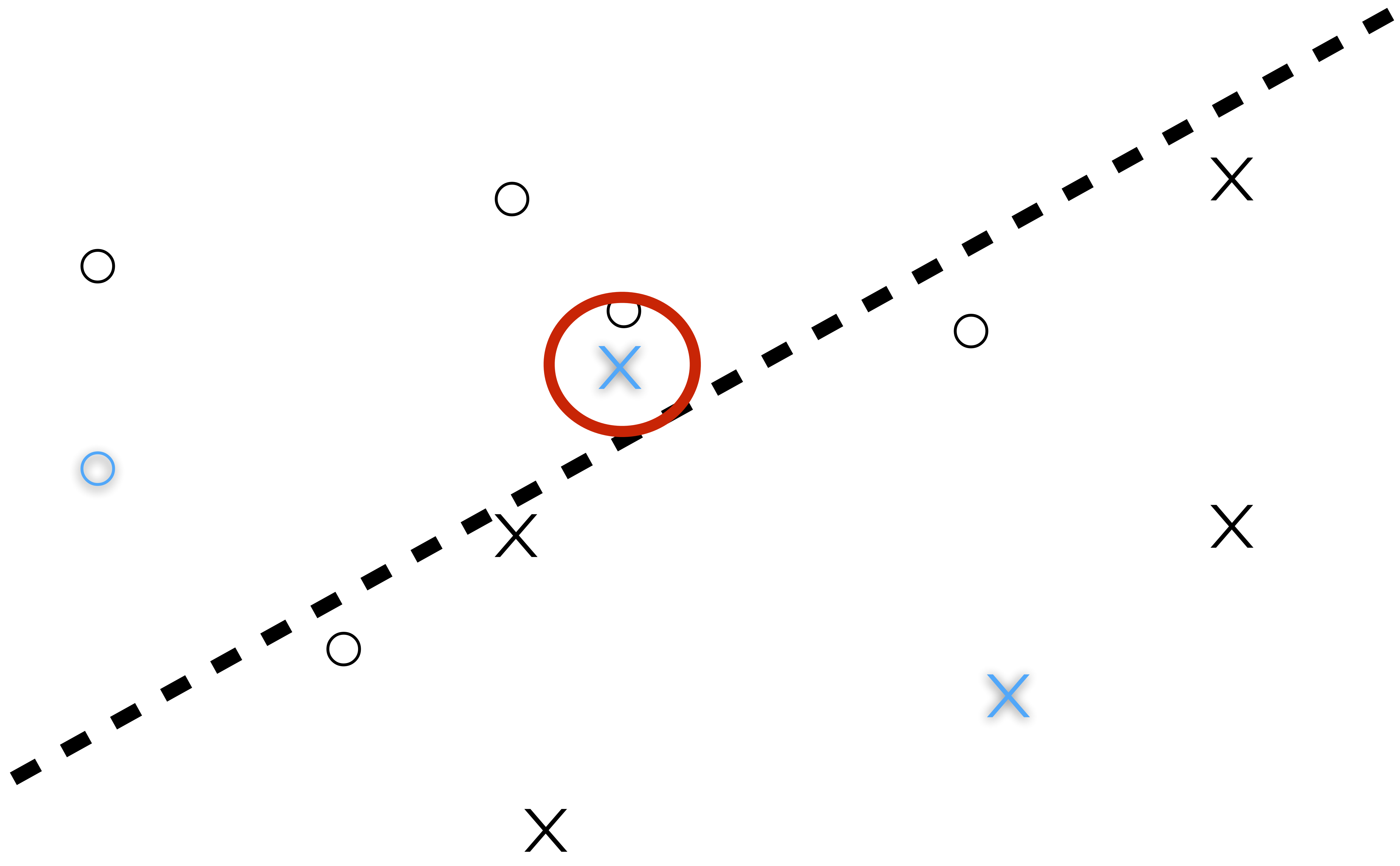


add back in hold-out group, compute error

Geoff Gordon

How can we evaluate a single model?

- Hold-out set (aka validation set)



estimated error rate: 1/3

add back in hold-out group, compute error

Geoff Gordon

Test set

- But now if we use this evaluation to pick a model, we get overfitting again
- So we need to evaluate on yet another set of independent examples — called ***test set***

Model selection with train/ validation/ test split

- Start with available data $\mathscr{D}$
- Split into three parts:
  - $\mathscr{D}_{\text{train}}$: learning within a model class
  - $\mathscr{D}_{\text{val}}$: model / hyperparameter selection
  - $\mathscr{D}_{\text{test}}$: final evaluation
- The three partitions need to be ***independent***
  - i.e., no information leakage
- With ideal data, this just means disjoint
  - but in real world, need to be more careful
  - e.g., data collected at same site might be correlated
  - or nearby in time, or by same survey taker, or …

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	Low	Normal	No
No	Medium	Normal	No
Yes	Medium	Abnormal	Yes

Model selection with train/ validation/ test split

- Start with available data $\mathscr{D}$
- Split into three parts:
  - $\mathscr{D}_{\text{train}}$: learning within a model class
  - $\mathscr{D}_{\text{val}}$: model / hyperparameter selection
  - $\mathscr{D}_{\text{test}}$: final evaluation
- The three partitions need to be ***independent***
  - i.e., no information leakage
- With ideal data, this just means disjoint
  - but in real world, need to be more careful
  - e.g., data collected at same site might be correlated
  - or nearby in time, or by same survey taker, or …

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	Low	Normal	No
No	Medium	Normal	No
Yes	Medium	Abnormal	Yes

Geoff Gordon

Model selection with train/ validation/ test split

- Start with available data $\mathscr{D}$
- Split into three parts:
  - $\mathscr{D}_{train}$:
  - $\mathscr{D}_{val}$: n
  - $\mathscr{D}_{test}$: f

tension: bigger $\mathscr{D}_{train}$ for better model learning vs. bigger $\mathscr{D}_{val}$ for better model selection vs. bigger $\mathscr{D}_{test}$ for more accurate final evaluation

- The three partitions need to be ***independent***
  - i.e., no information leakage
- With ideal data, this just means disjoint
  - but in real world, need to be more careful
  - e.g., data collected at same site might be correlated
  - or nearby in time, or by same survey taker, or …

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
	Medium	Normal	No
	Low	Abnormal	Yes
	Medium	Normal	Yes
	High	Abnormal	Yes
No	Low	Normal	No
No	Medium	Normal	No
Yes	Medium	Abnormal	Yes

Model selection with train/validation/test split

- Now, for each model:
  - train on $\mathscr{D}_{\text{train}}$
  - evaluate on $\mathscr{D}_{\text{val}}$ — track best
- Finally, evaluate single best model on $\mathscr{D}_{\text{test}}$

good parameters, maybe not best, and optimistic performance estimate

each individual evaluation is unbiased, but max is again optimistic

an unbiased estimate of the selected model's performance

Model selection with train/validation/test split

- Now, for each model:
  - train on $\mathscr{D}_{\text{train}}$
  - evaluate on $\mathscr{D}_{\text{val}}$ — track best
- [optionally] retrain best model on $\mathscr{D}_{\text{train}} \cup \mathscr{D}_{\text{val}}$
- Finally, evaluate single best model on $\mathscr{D}_{\text{test}}$

good parameters, maybe not best, and optimistic performance estimate

each individual evaluation is unbiased, but max is again optimistic

more training data

an unbiased estimate of the selected model's performance

Greedy search

- If there are lots of models, might not be able to train and evaluate all of them w/ available compute
- In this case, might do a greedy search
  - e.g., to get a good decision tree with 18 nodes, start from our 17-node tree and pick a single node to split
  - we don't even look at all available models (tree structures), instead focusing on neighbors of ones we already think are good
  - within a model, we might not find optimal parameters (e.g., don't optimize all thresholds at once, just the latest)

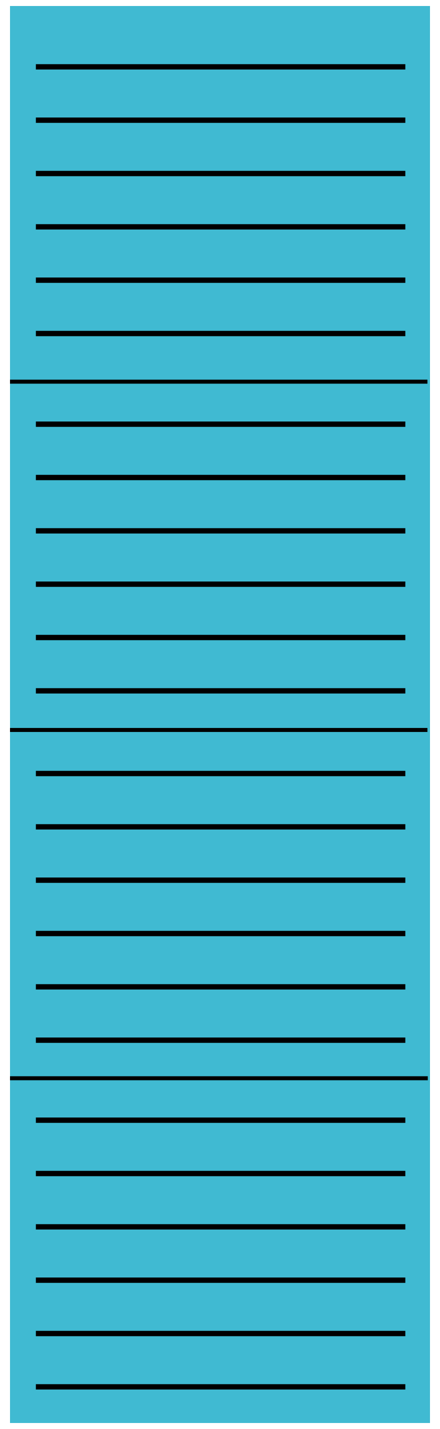
Cross-validation

- Tension between size of $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{val}}$
- What if we could put each training example in both?
- Split data into k parts (called ***folds***), repeatedly train on some folds and validate on others
  - each fold serves as validation for each other fold, hence ***cross***-validation

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	Low	Normal	No
No	Medium	Normal	No
Yes	Medium	Abnormal	Yes

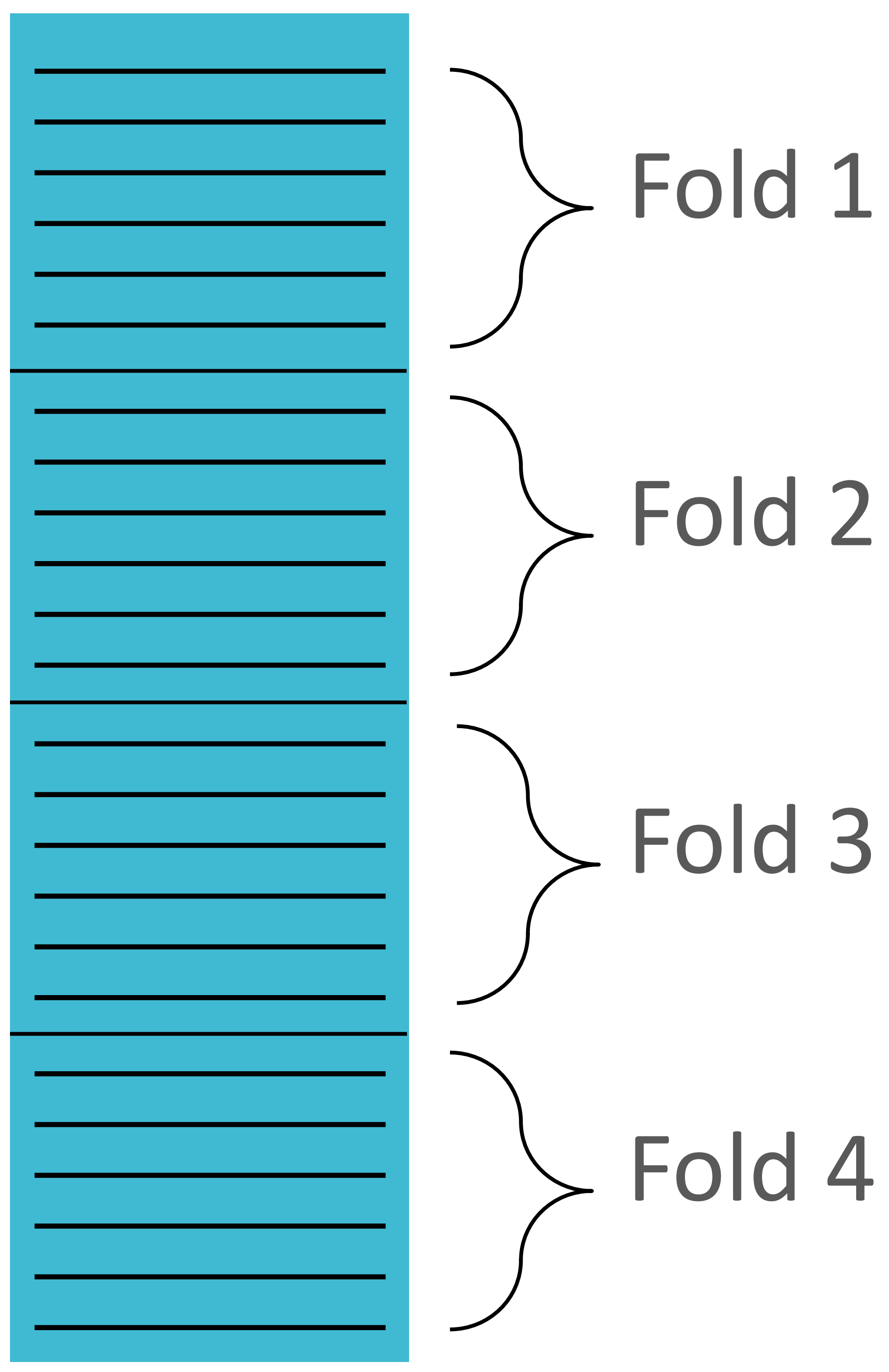
Cross-validation

- Tension between size of $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{val}}$
- What if we could put each training example in both?
- Split data into k parts (called ***folds***), repeatedly train on some folds and validate on others
  - each fold serves as validation for each other fold, hence ***cross***-validation



Cross-validation

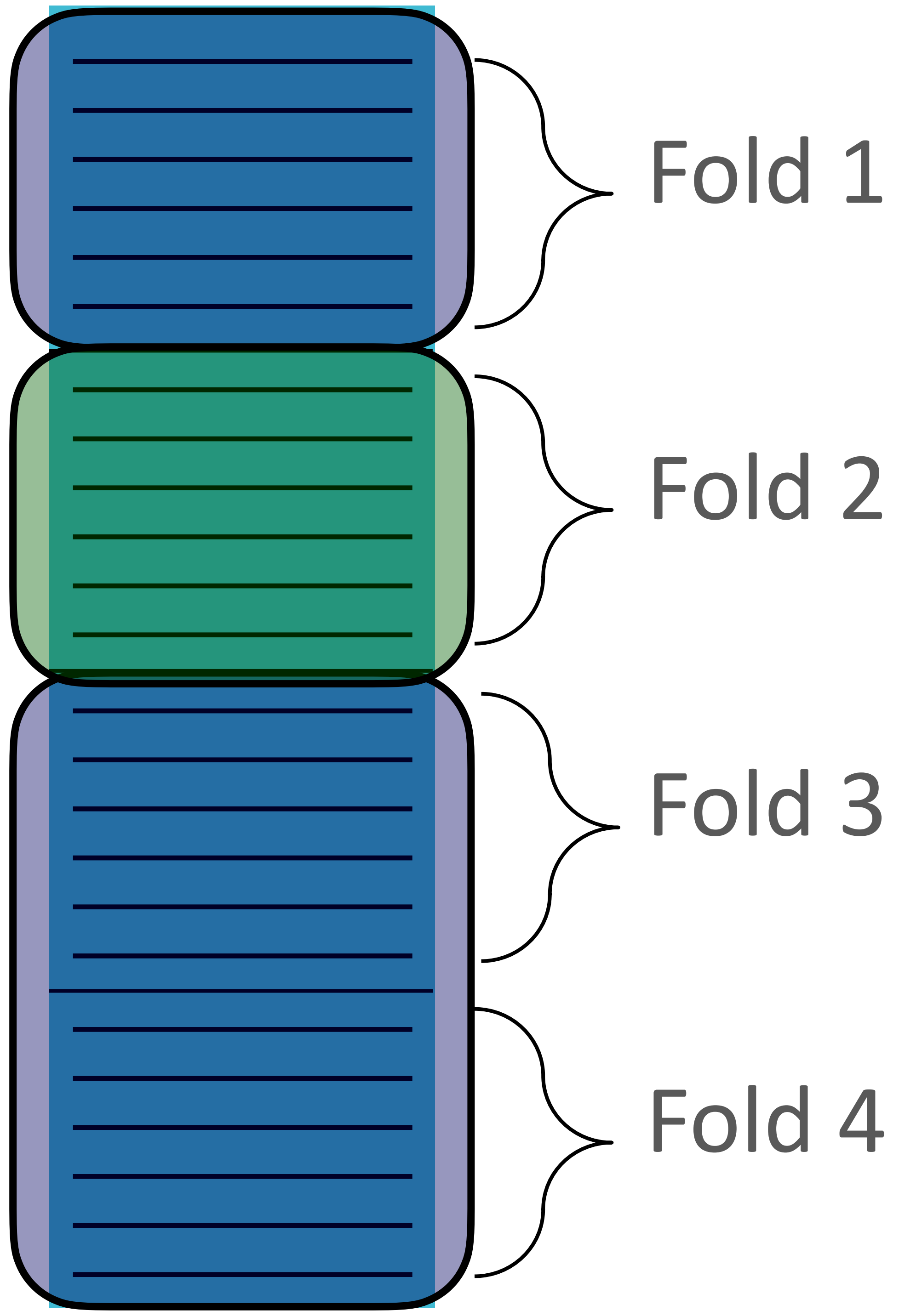
- Tension between size of $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{val}}$
- What if we could put each training example in both?
- Split data into k parts (called ***folds***), repeatedly train on some folds and validate on others
  - each fold serves as validation for each other fold, hence ***cross***-validation



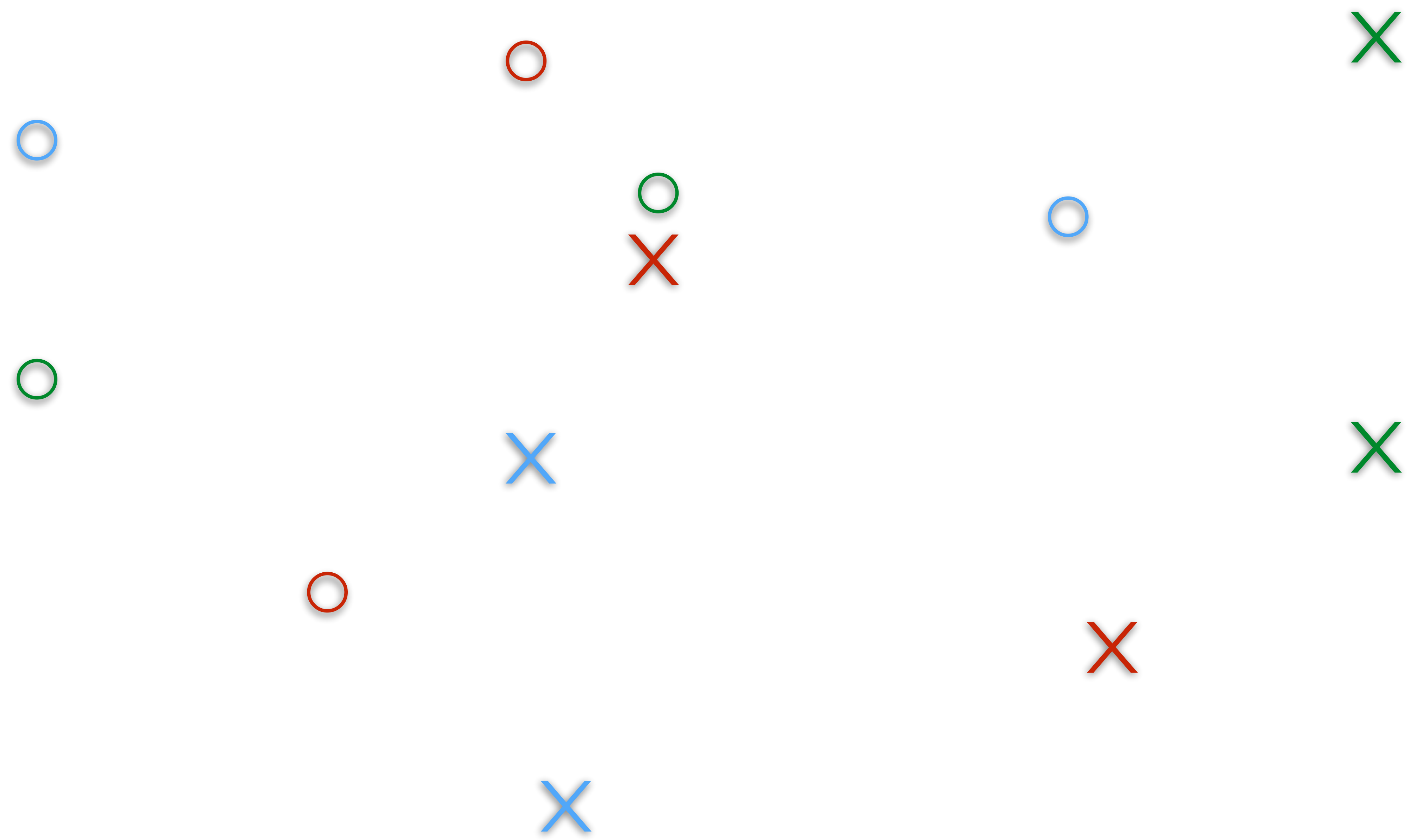
Geoff Gordon

Cross-validation

- Tension between size of $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{val}}$
- What if we could put each training example in both?
- Split data into k parts (called ***folds***), repeatedly train on some folds and validate on others
  - each fold serves as validation for each other fold, hence ***cross***-validation



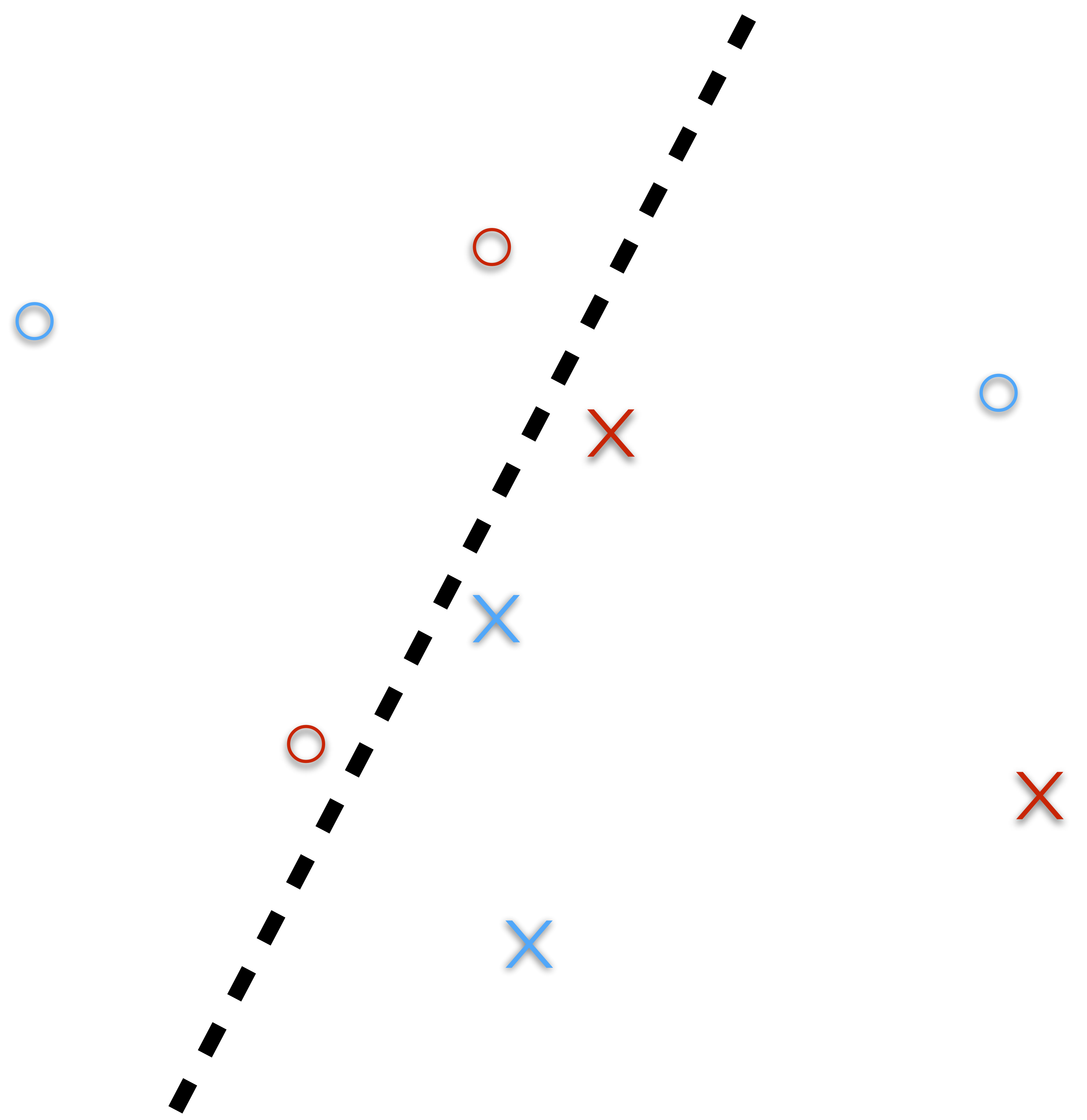
Cross-validation



split data evenly into groups ("folds")

Geoff Gordon

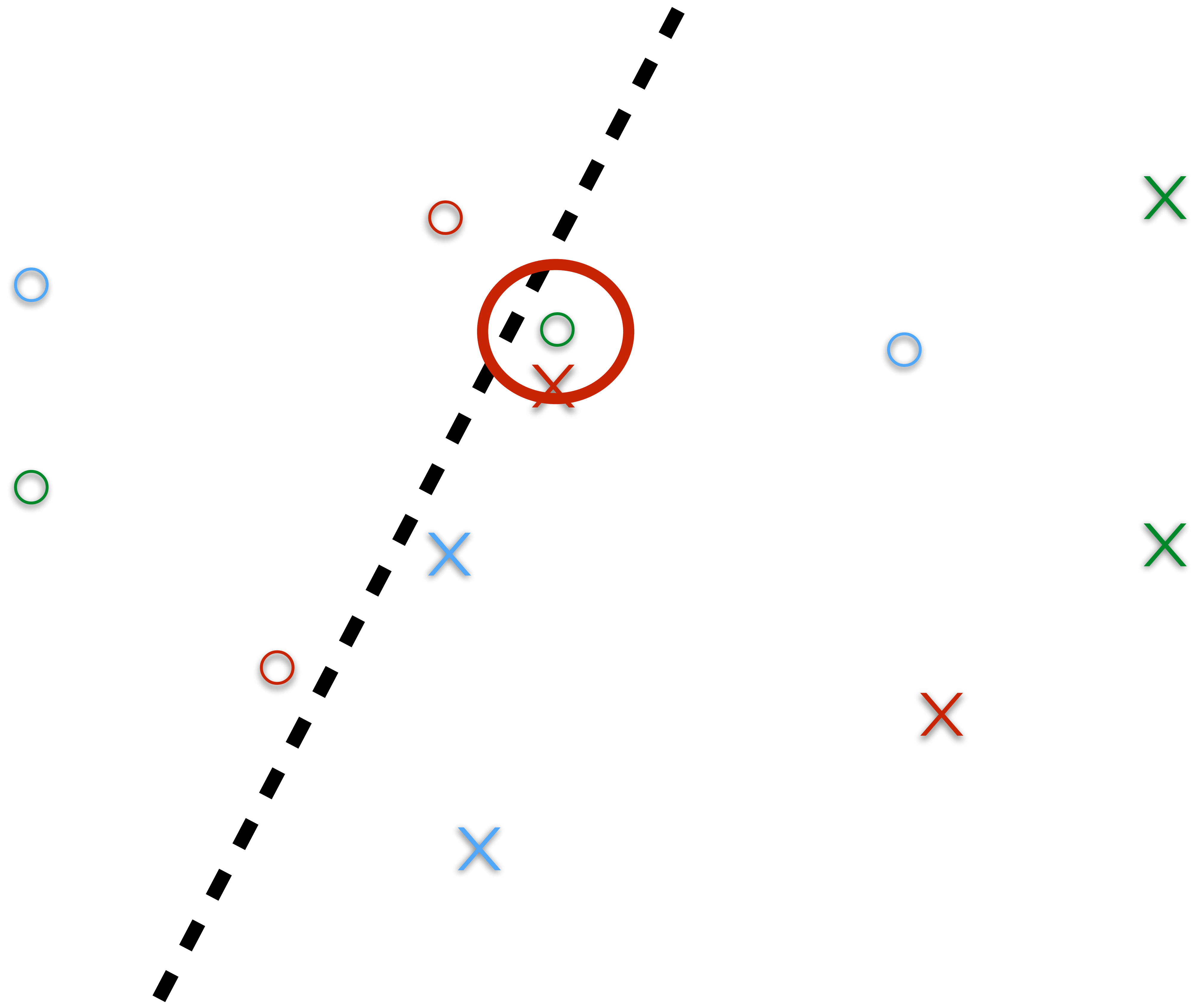
Cross-validation



remove green group, fit on rest

Geoff Gordon

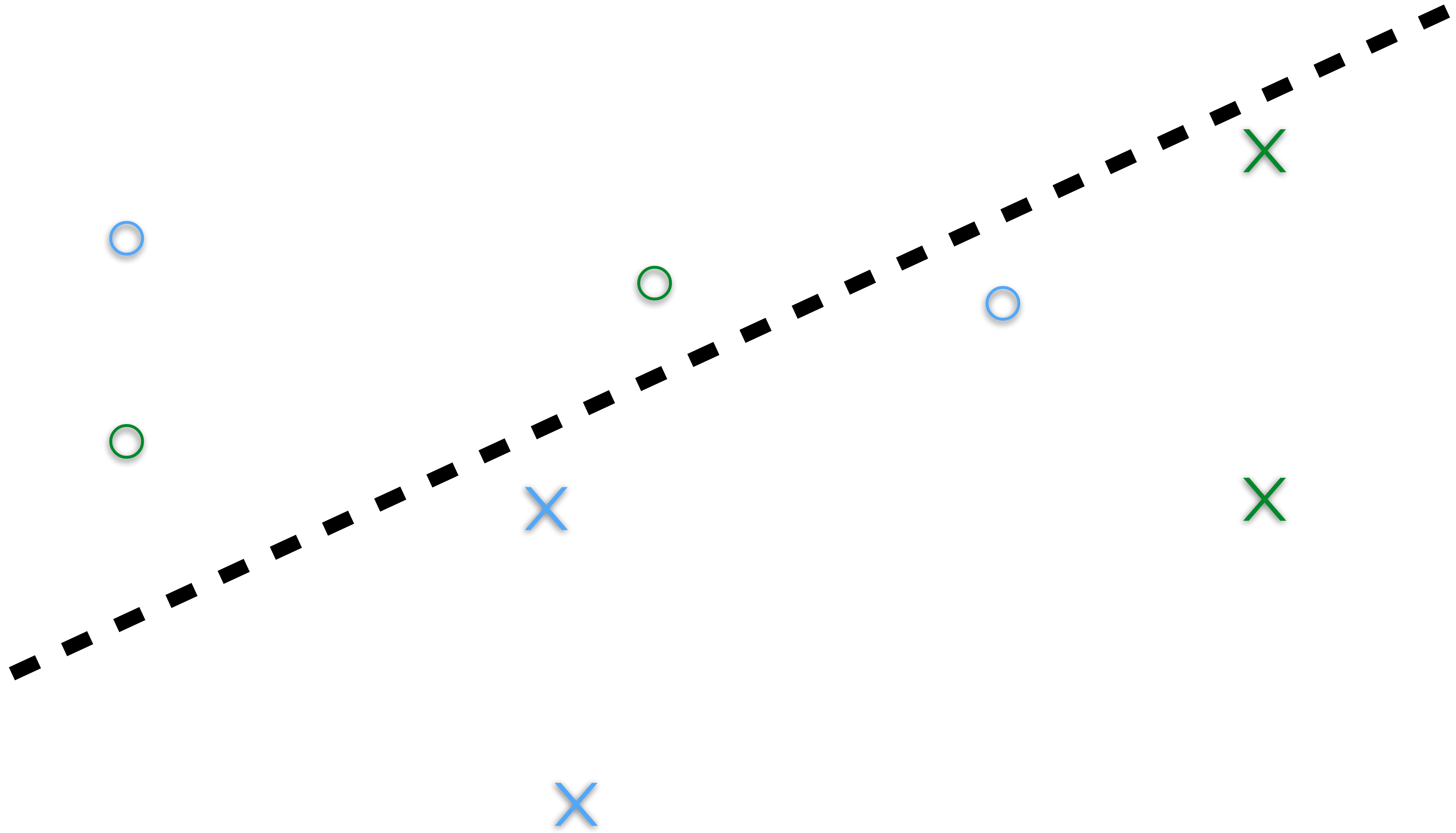
Cross-validation



add back green group: error 1/4

Geoff Gordon

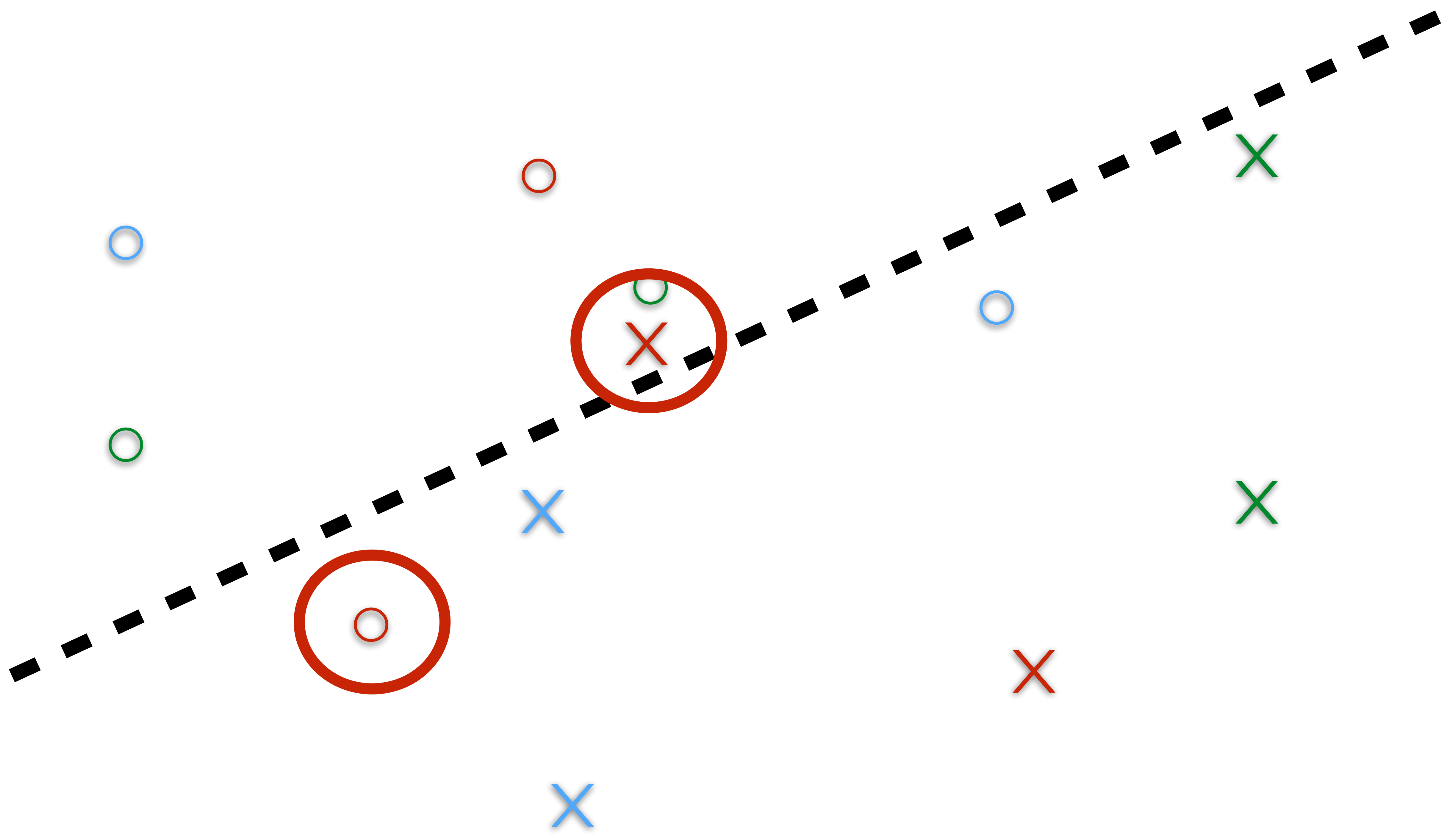
Cross-validation



remove red group, fit on rest

Geoff Gordon

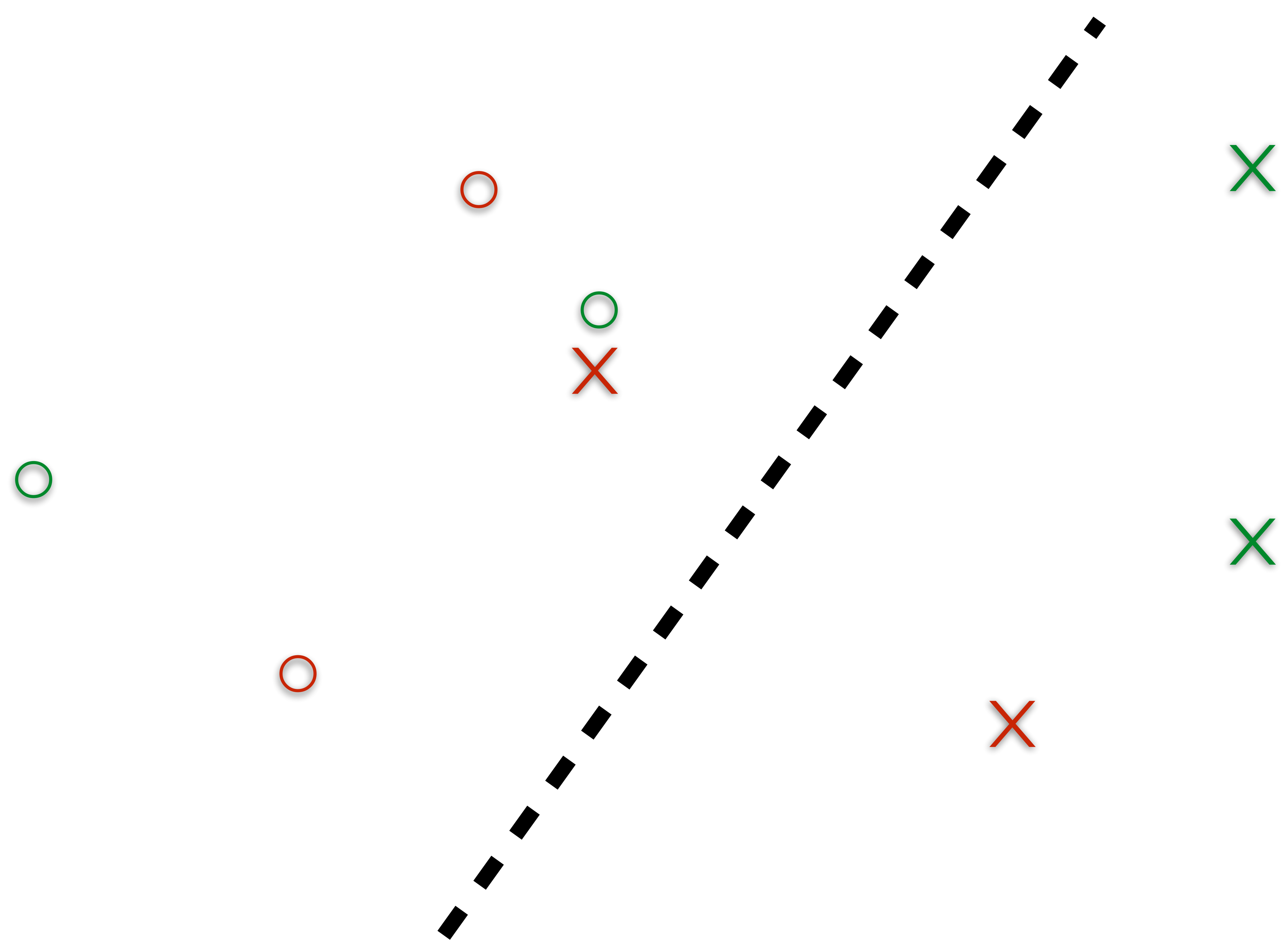
Cross-validation



add back red group: error 2/4

Geoff Gordon

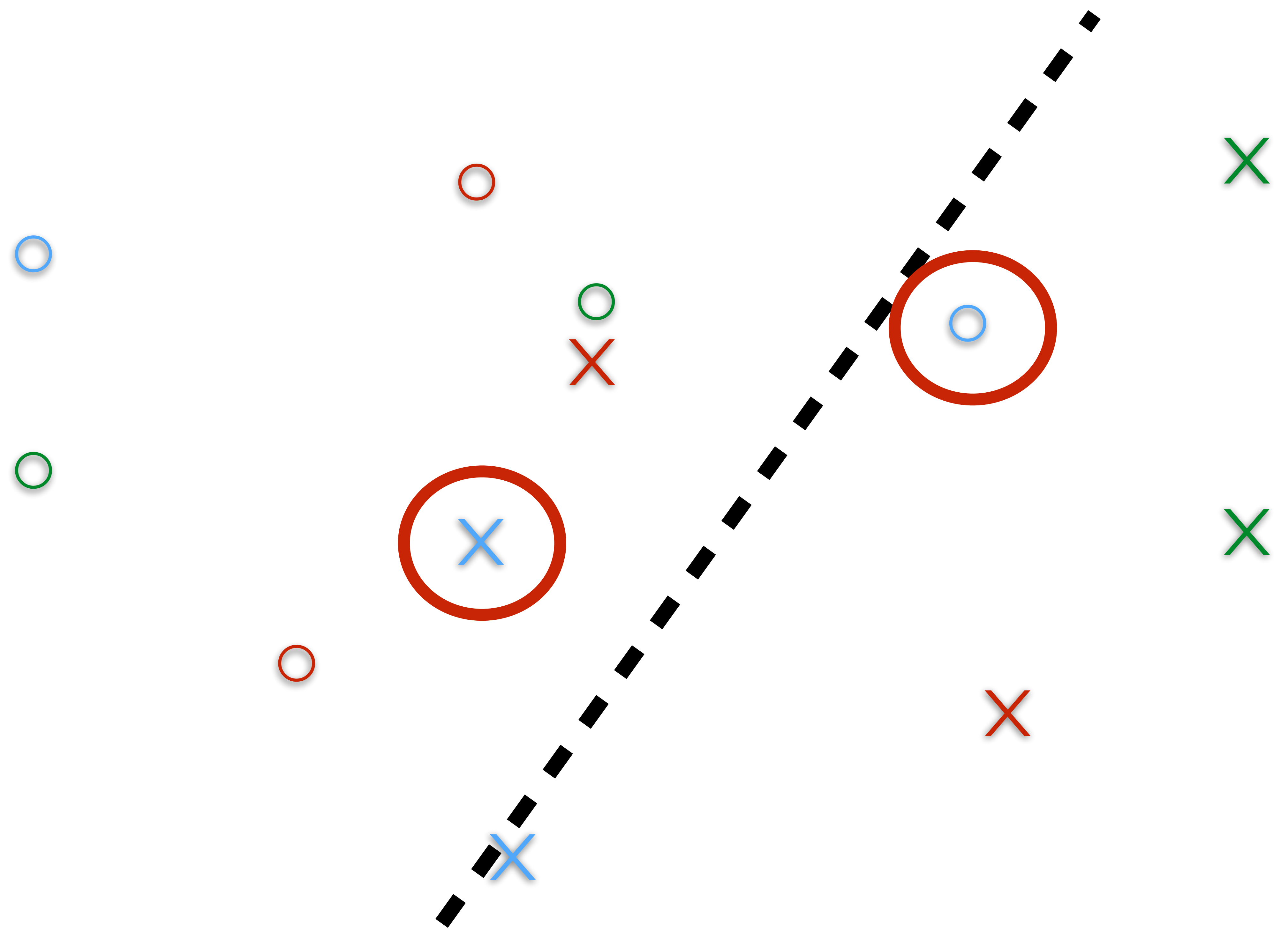
Cross-validation



remove blue group, fit on rest

Geoff Gordon

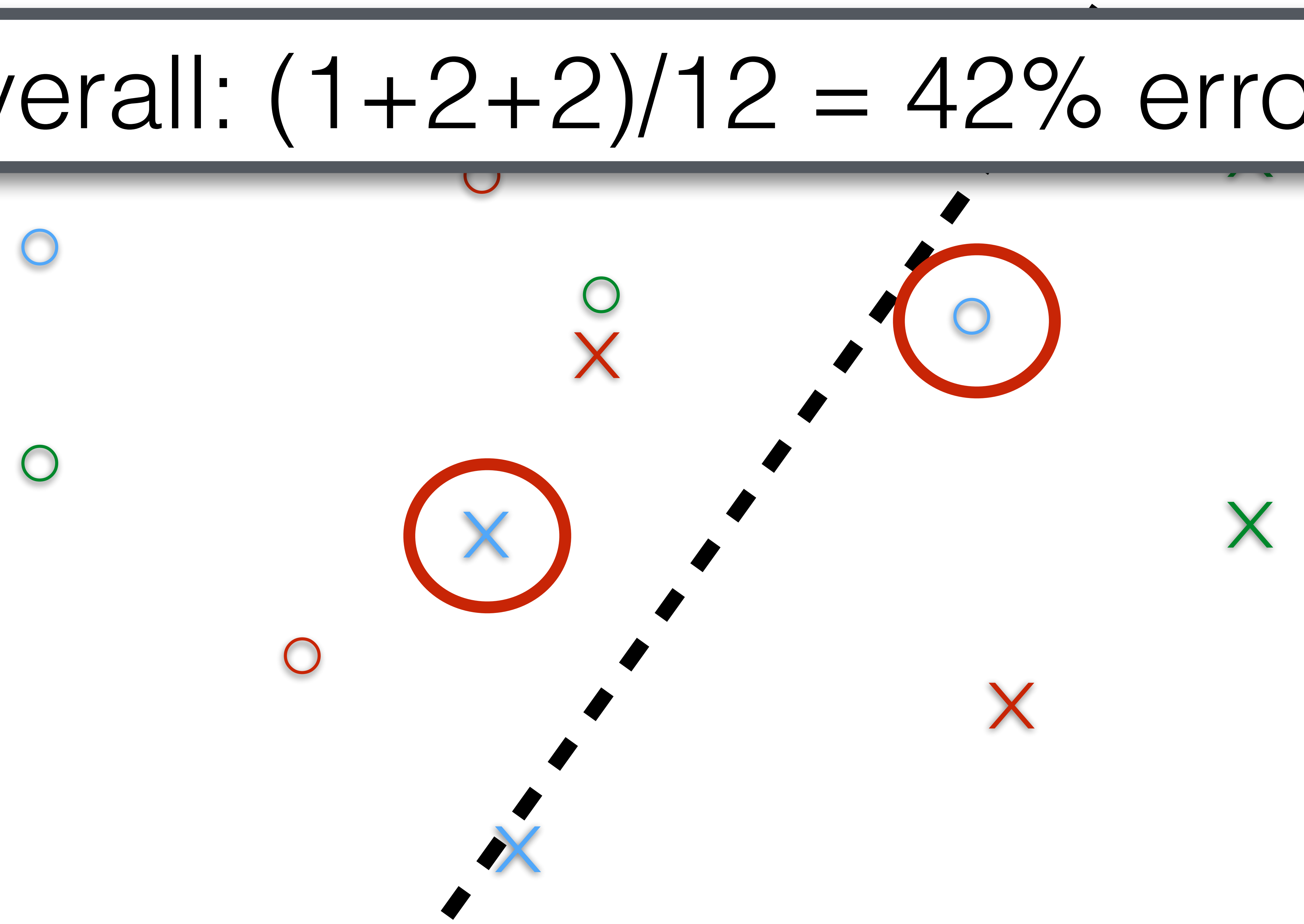
Cross-validation



add back blue group: error 2/4

Cross-validation

Overall: (1+2+2)/12 = 42% error rate



add back blue group: error 2/4

Geoff Gordon

Model selection with cross-validation

- Split into $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$
- Split $\mathscr{D}_{\text{train}}$ into folds
- For each model:
  - for each fold F
    - train on $\mathscr{D}_{\text{train}} \backslash F$
    - evaluate on F
  - average performance over folds — track best model
- [optionally] retrain best model on all folds
- Finally, evaluate single best model on $\mathscr{D}_{\text{test}}$

Model selection with cross-validation

- Split into $\mathscr{D}_{\text{train}}$ and $\mathscr{D}_{\text{test}}$
- Split $\mathscr{D}_{\text{train}}$ into folds
- For each model:
  - for each fold F
    - train on $\mathscr{D}_{\text{train}} \backslash F$
    - evaluate on F
  - average performance over folds — track best model
- [optionally] retrain best model on all folds
- Finally, evaluate single best model on $\mathscr{D}_{\text{test}}$

could also return ***all*** trained models, use them in an ***ensemble*** (e.g., by averaging their predictions)

Geoff Gordon