Neural Network Training

Danna Gurari
University of Colorado Boulder
Spring 2022

https://home.cs.colorado.edu/~DrG/Courses/NeuralNetworksAndDeepLearning/AboutCourse.html

Review

- Last lecture:
  - Objective function: what to learn
  - Gradient descent: how to learn
  - Training a neural network: optimization
  - Gradient descent for different activation functions

- Assignments (Canvas):
  - Problem set 1 grades out
  - Lab assignment 1 due Monday

- Questions?

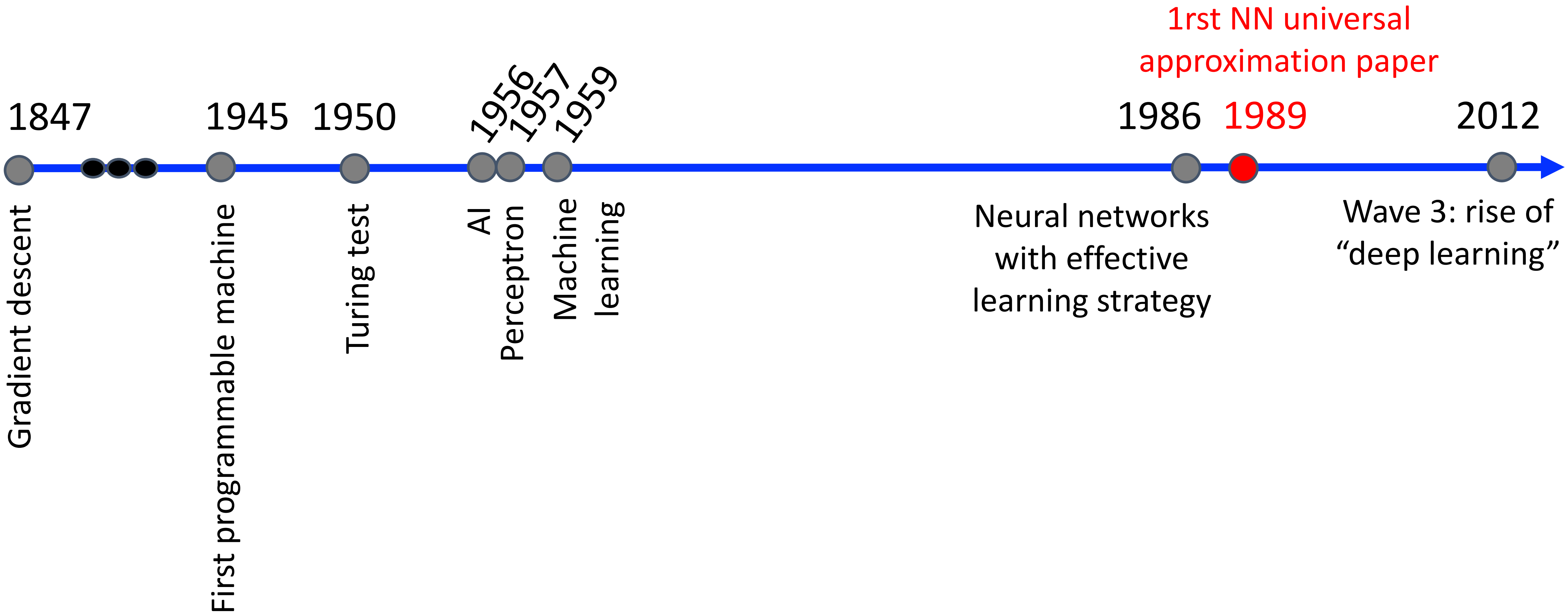
Today’s Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

Today's Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

Historical Context: Universal Approximator

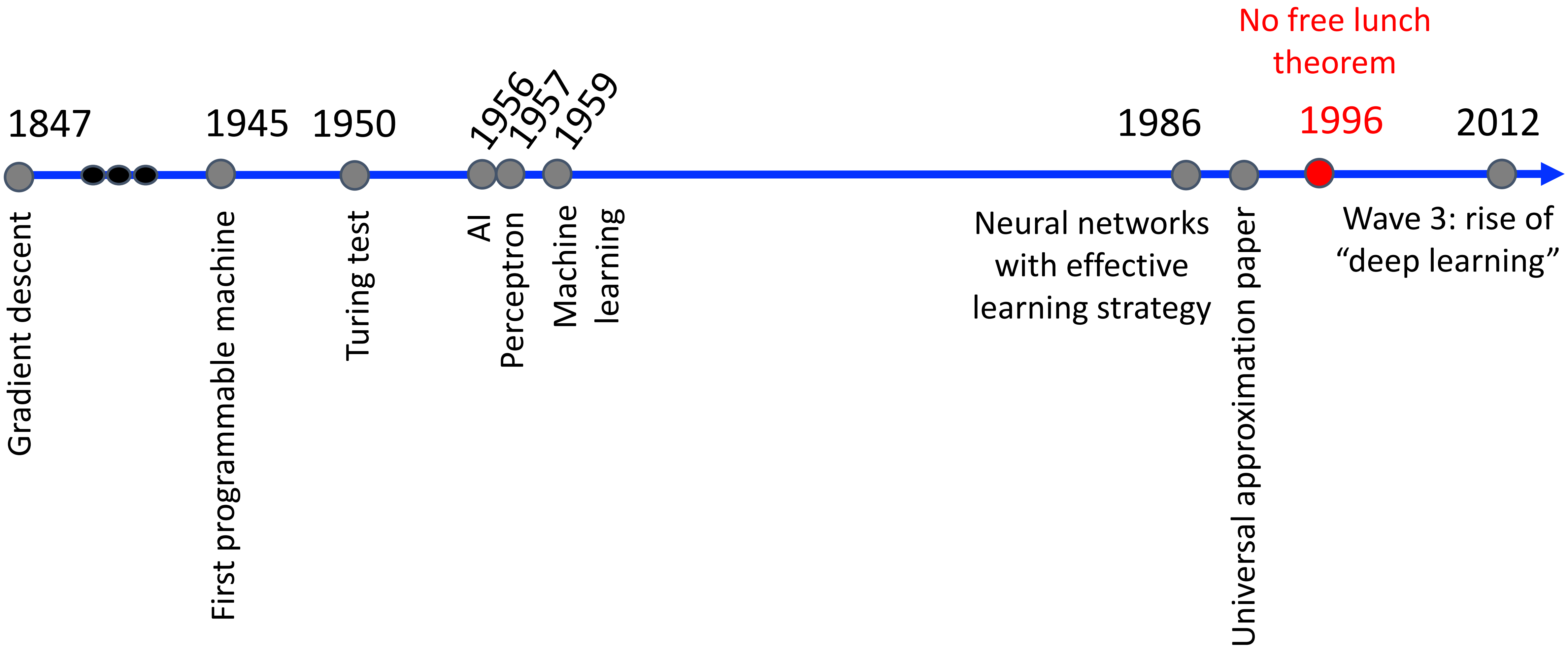


Hornik, Stinchcombe and White. Multilayer feedforward networks are universal approximators. Neural Networks, 1989

“The universal approximation theorem means that regardless of what function we are trying to learn, we know that a large MLP [multilayer perceptron] will be able to *represent* this function.”

- Ch. 6.4.1 of Goodfellow book on Deep Learning

Historical Context: Challenge



Hornik, Stinchcombe and White. Multilayer feedforward networks are universal approximators. Neural Networks, 1989 “**no free lunch theorem**… no machine learning algorithm is universally is any better than any other.”

- Ch. 5.2.1 of Goodfellow book on Deep Learning

Deep Learning Goal

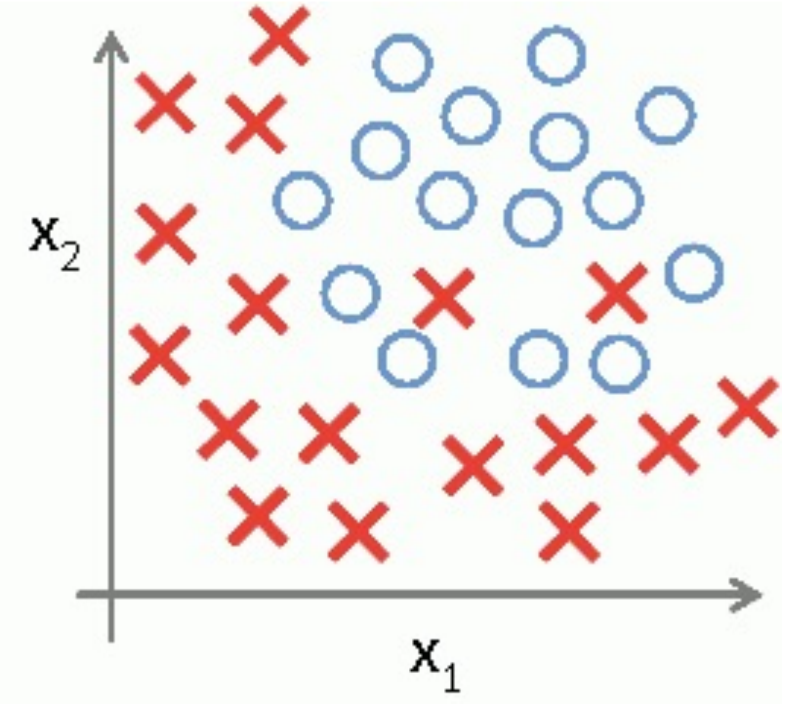
Since neural networks can in theory represent ANY function, how do we learn models that can perform well for the data generated in real world problems…

Today’s Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

Recall: Class Exercise from Lecture 1

- Model-based classification approach: separate x from o



Class volunteer:
1) Draw a straight line (linear equation)
2) Draw a parabola (quadratic equation)
3) Draw any curve

Models with increasing representational capacity

Figure source: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

Model Capacity

Which model would you choose to separate x from o?

(a) (b) (c)

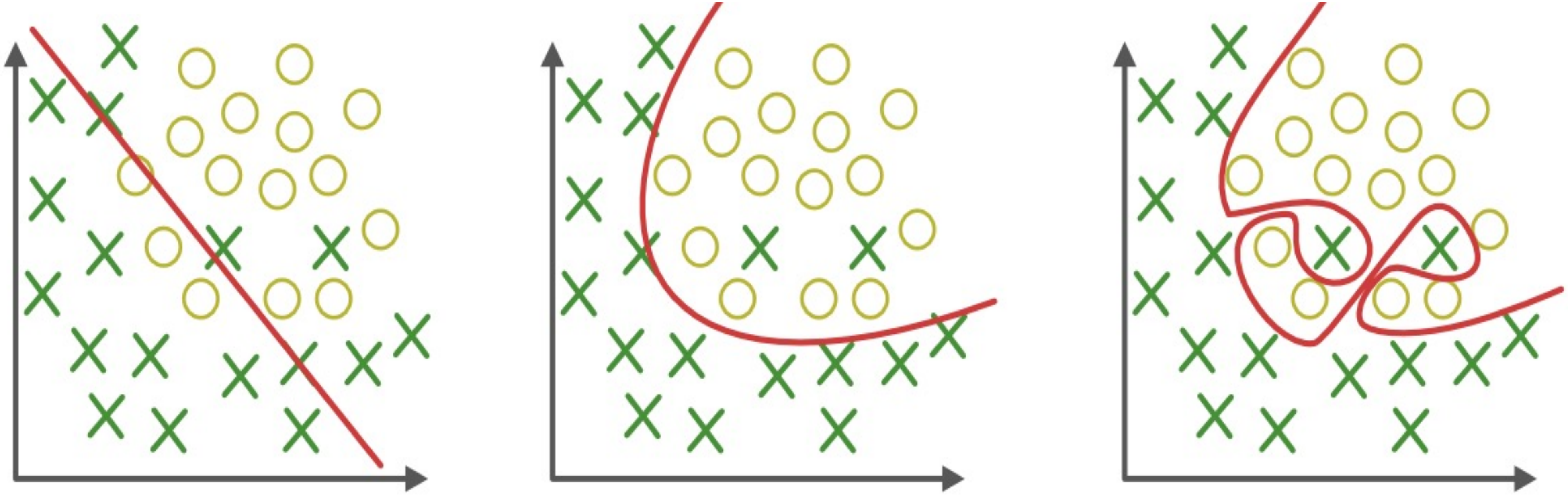


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity

Underfits: too simple to explain the data

Overfits: too complex to generalize to a test set

(a) (b) (c)

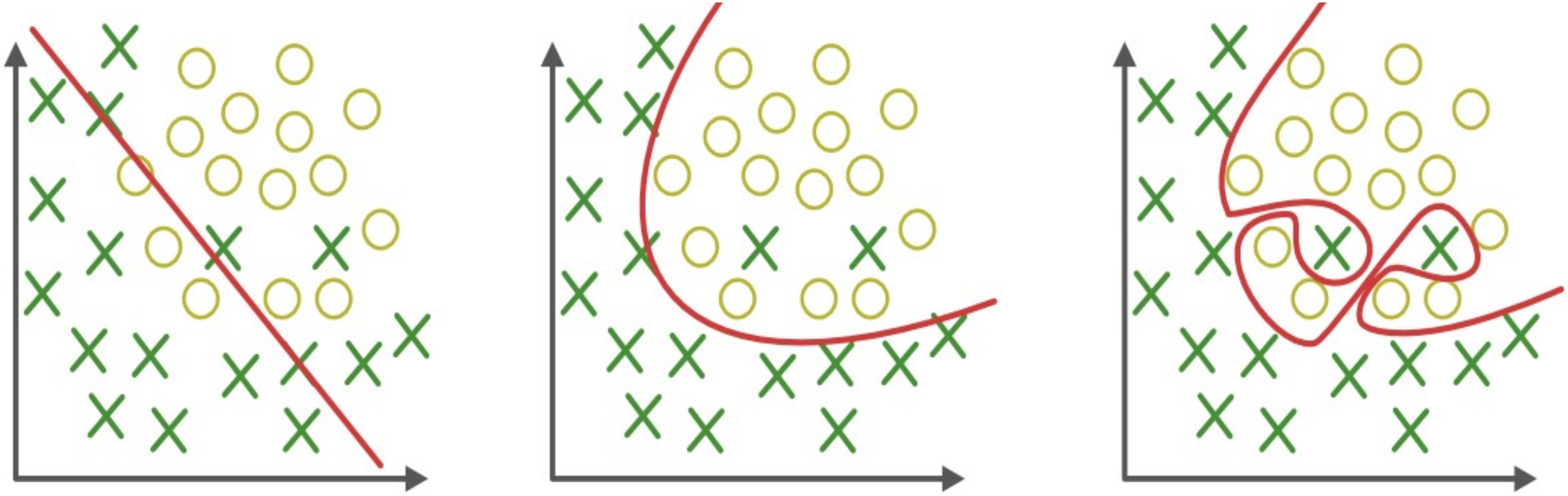


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity

Key challenge for neural networks since they have many parameters

Underfits: too simple to explain the data

Overfits: too complex to generalize to a test set

(a) (b) (c)

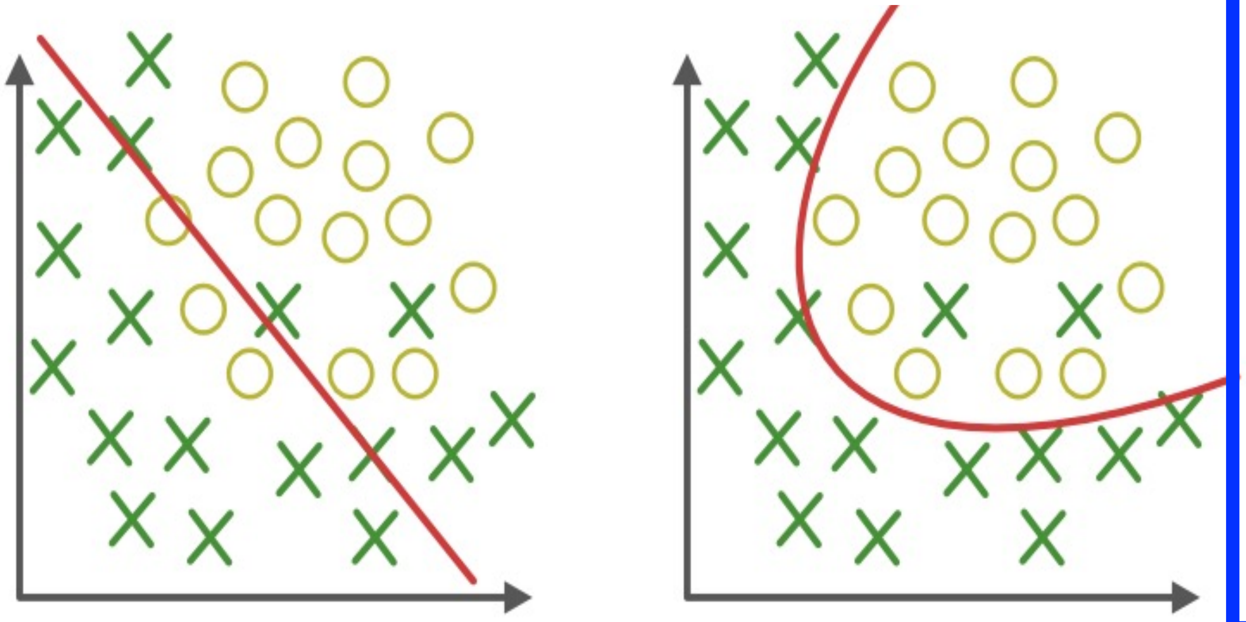


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity: Overfitting

- What is learned by models that overfit?
  - How to model **noise!**
- What would cause noise in a dataset?

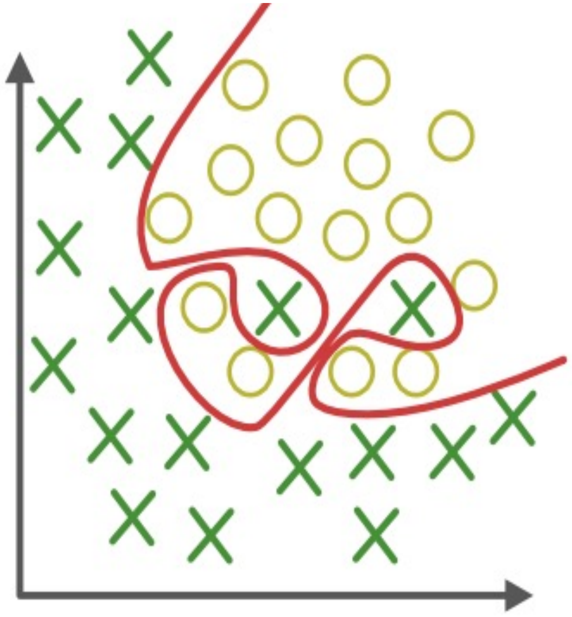
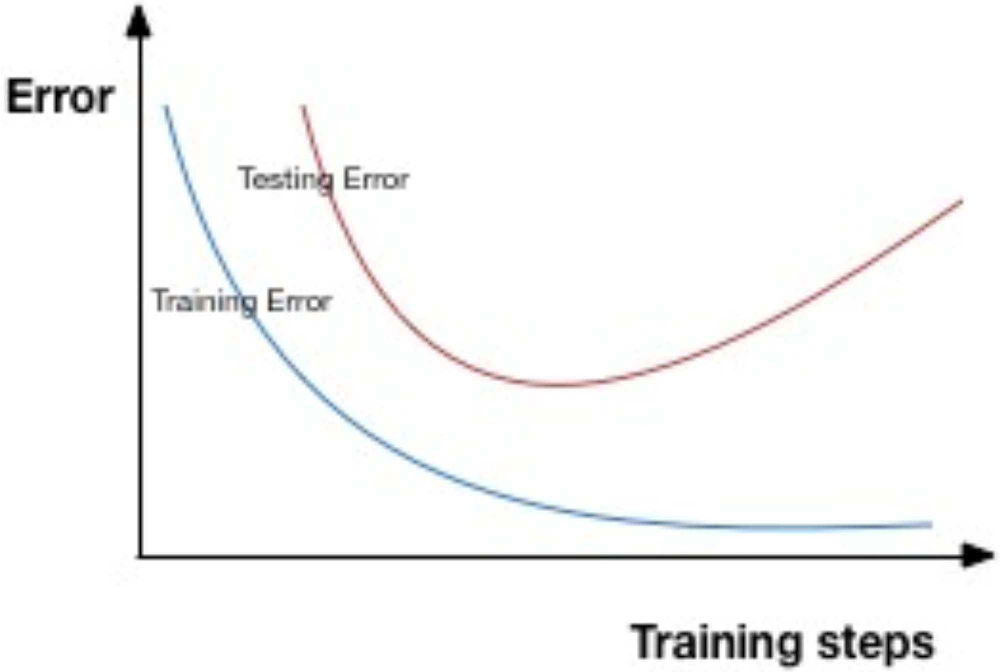


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity: Overfitting

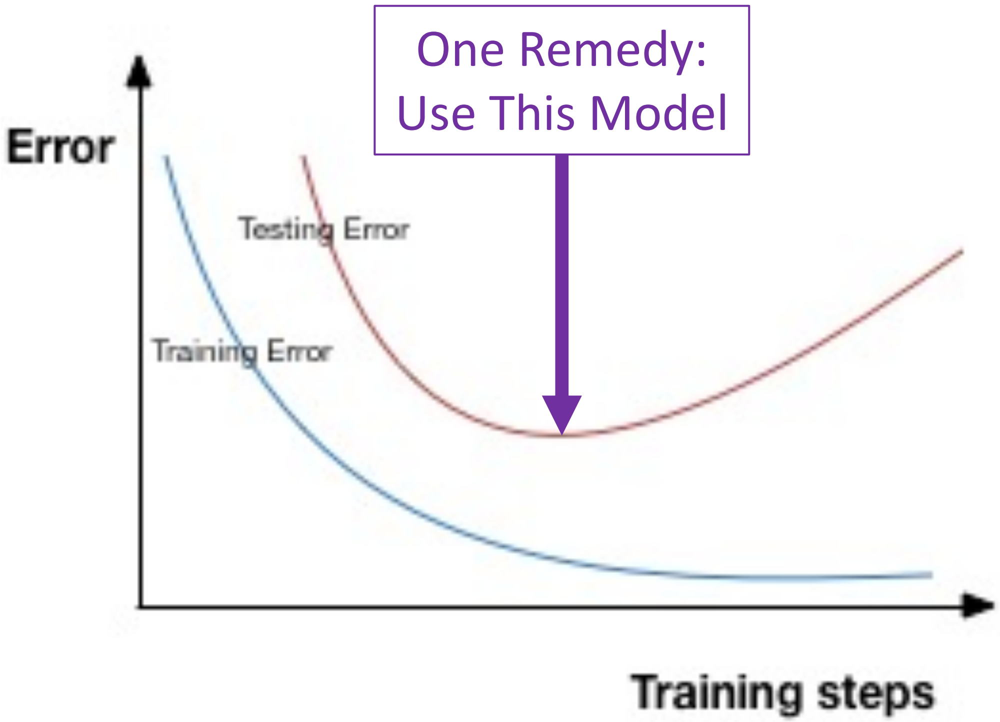


- To detect overfitting, analyze error/loss for models tested on training data and test data
  - What happens to training data error as number of training steps increases?
    - Error shrinks
  - What happens to test data error as number of training steps increases?
    - Error shrinks and then grows
  - Why does training error ***shrink*** and test error ***grow***?
    - Modeling ***noise*** in the training data (i.e., “overfitting”) reduces training error at the expense of losing knowledge that generalizes to unobserved test data

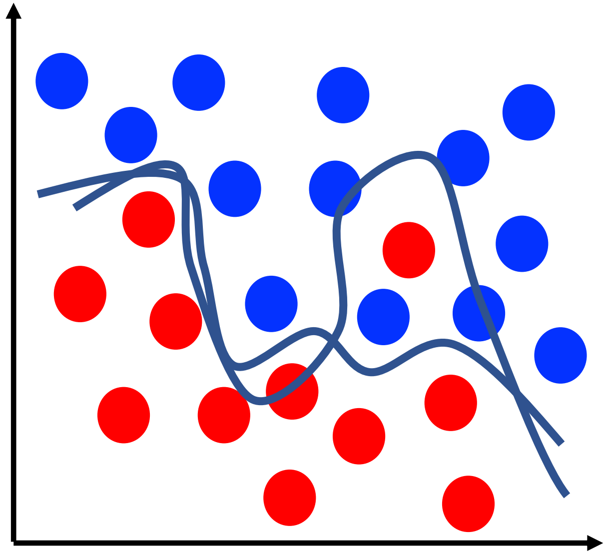
Image Source: https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Model Capacity: How to Avoid Overfitting?

Early stopping



Add training data



Many more techniques to be discussed in this course…

Image Source: https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Model Capacity

Underfits: too simple to explain the data

Overfits: too complex to generalize to a test set

(a) (b) (c)

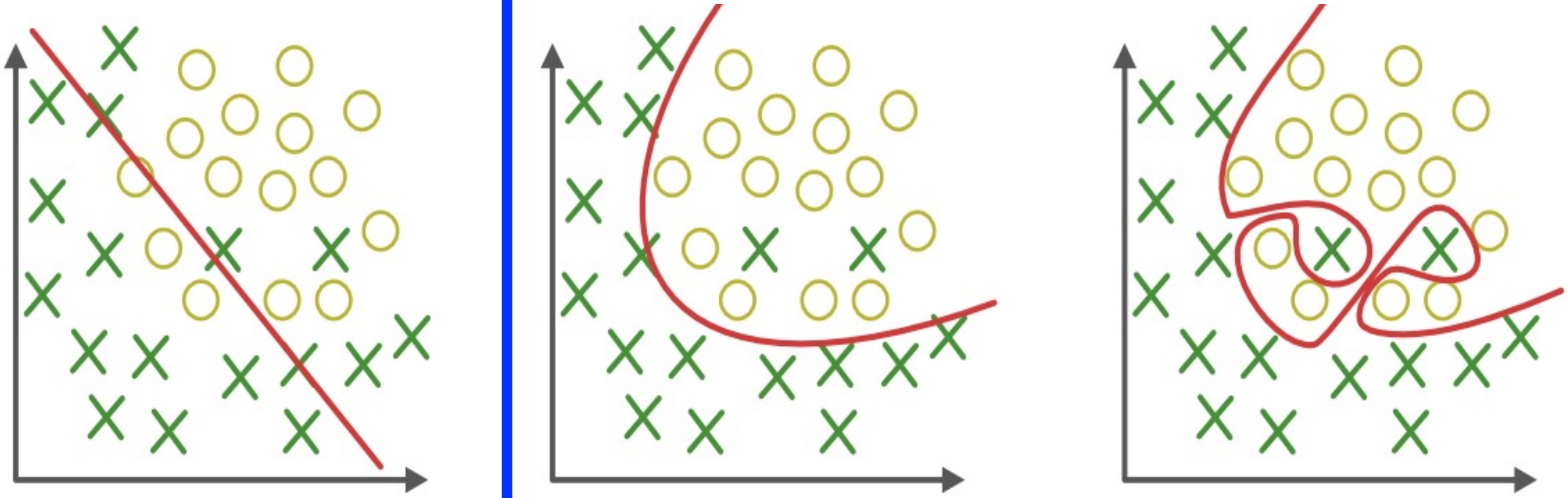
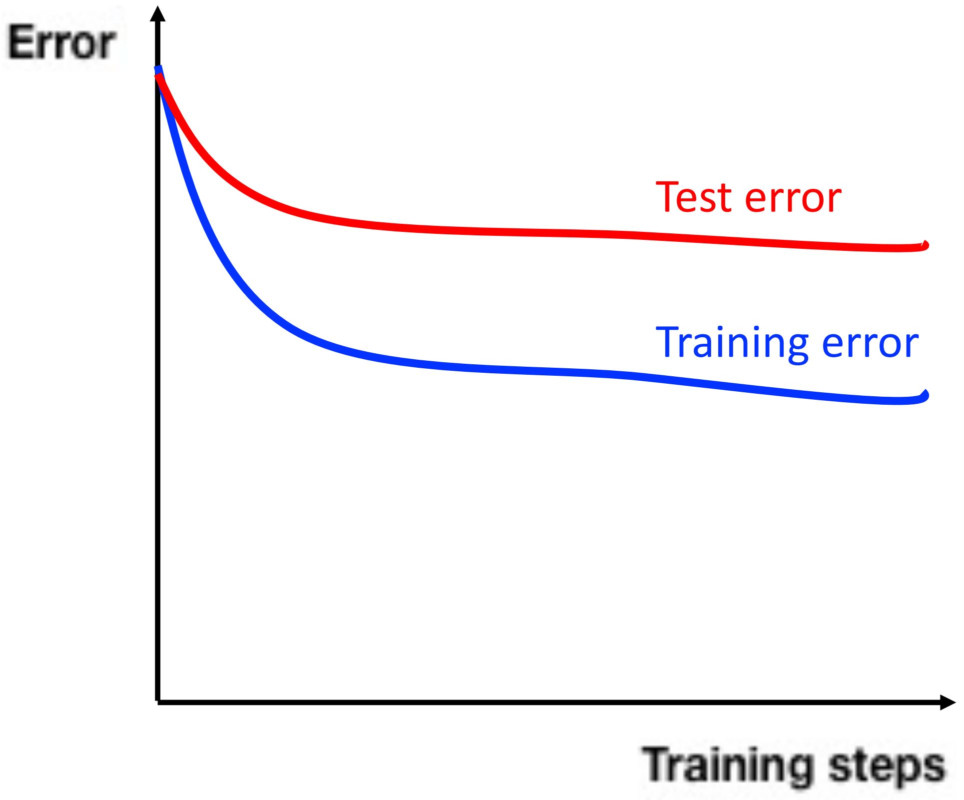


Figure source: https://towardsdatascience.com/underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6fe4a8a49dbf

Model Capacity: Underfitting



- To detect overfitting, analyze error/loss for models tested on training data (and optionally test data)
  - What happens to training data error as number of training steps increases?
    - Error remains high
  - What happens to test data error as number of training steps increases?
    - Error remains high

Model Capacity: How to Avoid Underfitting?

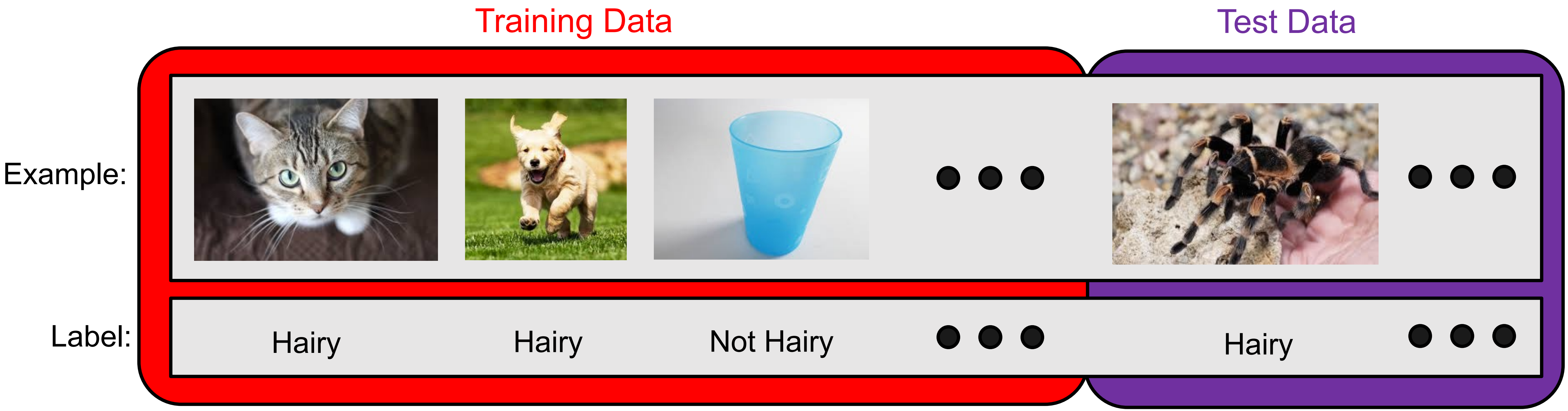
Increase representational complexity, for example add the number of layers and/or units in a neural network

Goal: learn a model with a capacity that is neither too small nor too large so it can generalize well when predicting on previously unseen test data

Today’s Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

Recall: Our Goal is to Design Models that **Generalize** Well to New, Previously Unseen Examples (Test Data)



Key Challenge: how to select a model without repeatedly observing the test data (which leads to overfitting)?

Model Design Decisions

Model hyperparameters (selected); e.g.,
- Number of layers
- Number of units in each layer
- Activation function
- Batch size
- Learning rate
- …

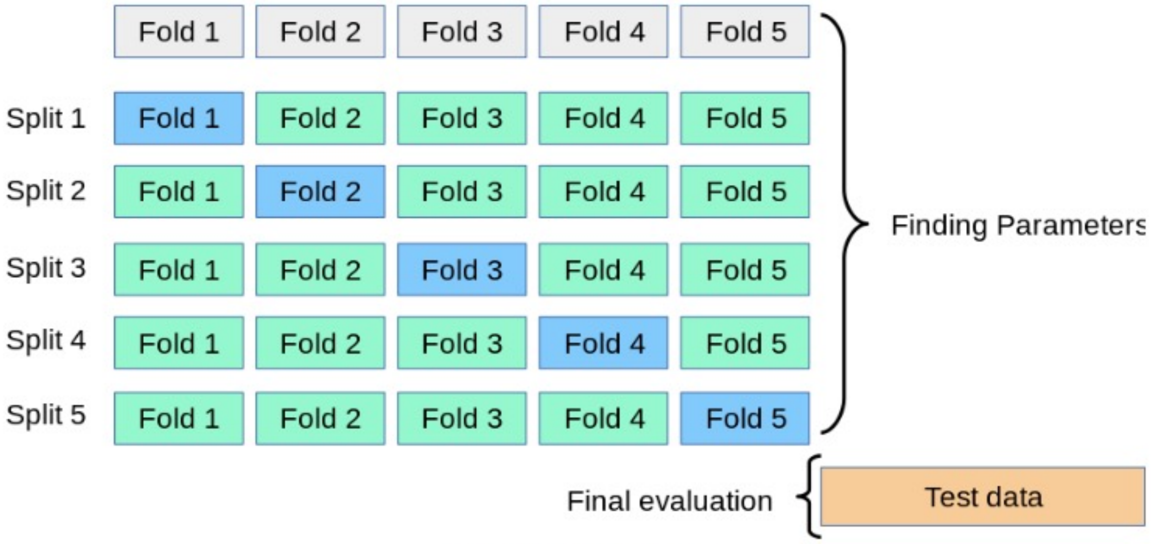
Model parameters (learned)
- Weights
- Biases

Key Challenge: how to select a model without repeatedly observing the test data (which leads to overfitting)?

Hyperparameter Tuning: Split Training Set So It Can Be Used to Test Different Hyperparameters

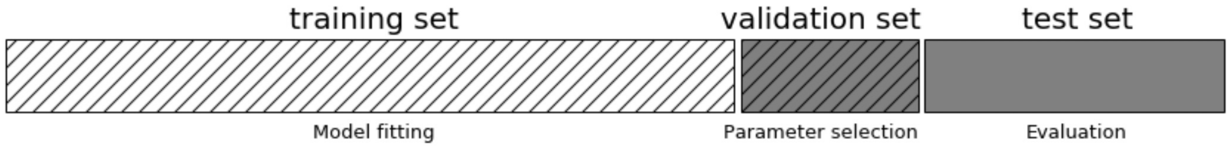
For statistically strong results:

Small training dataset: cross validation



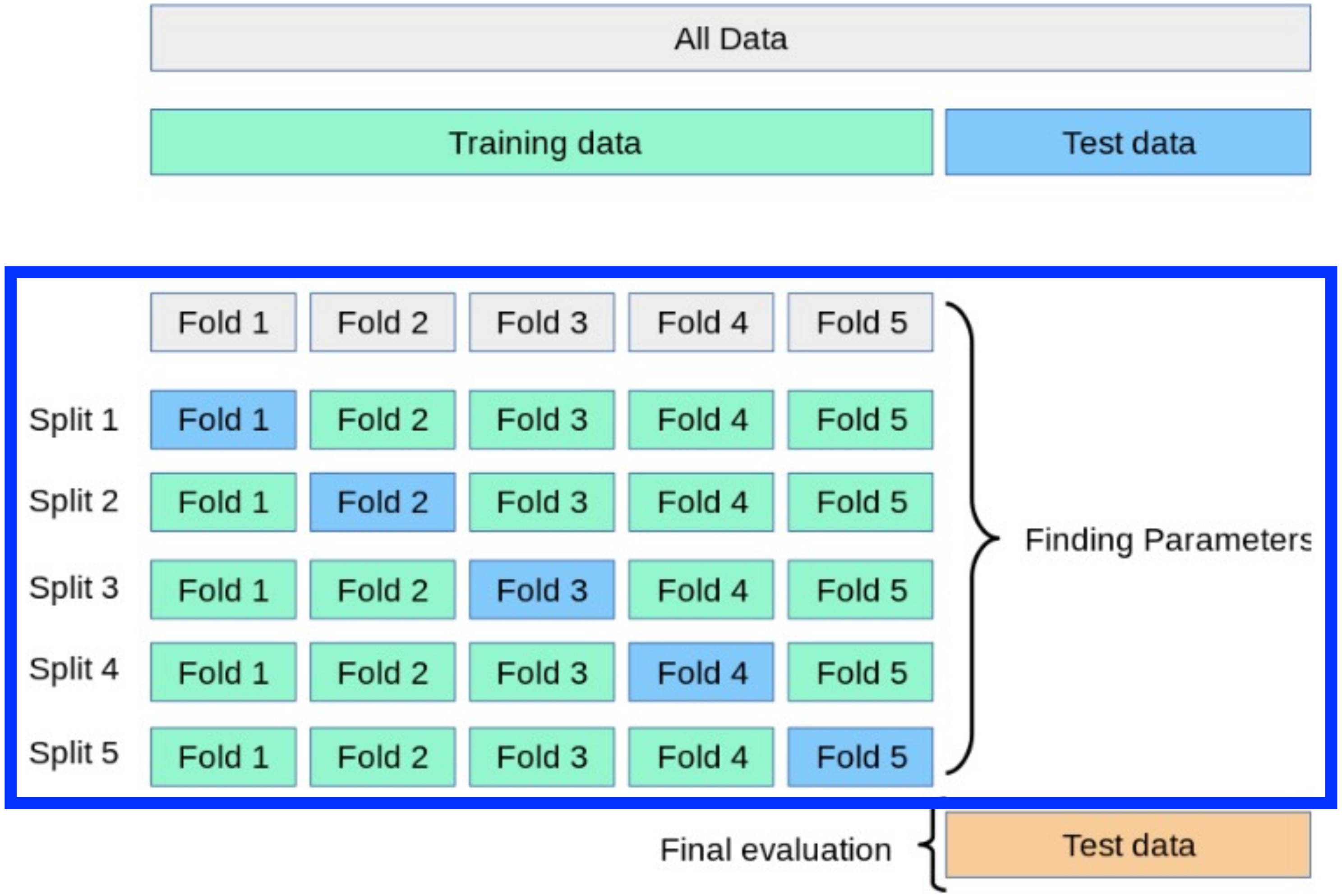
https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Else: train/validation split



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

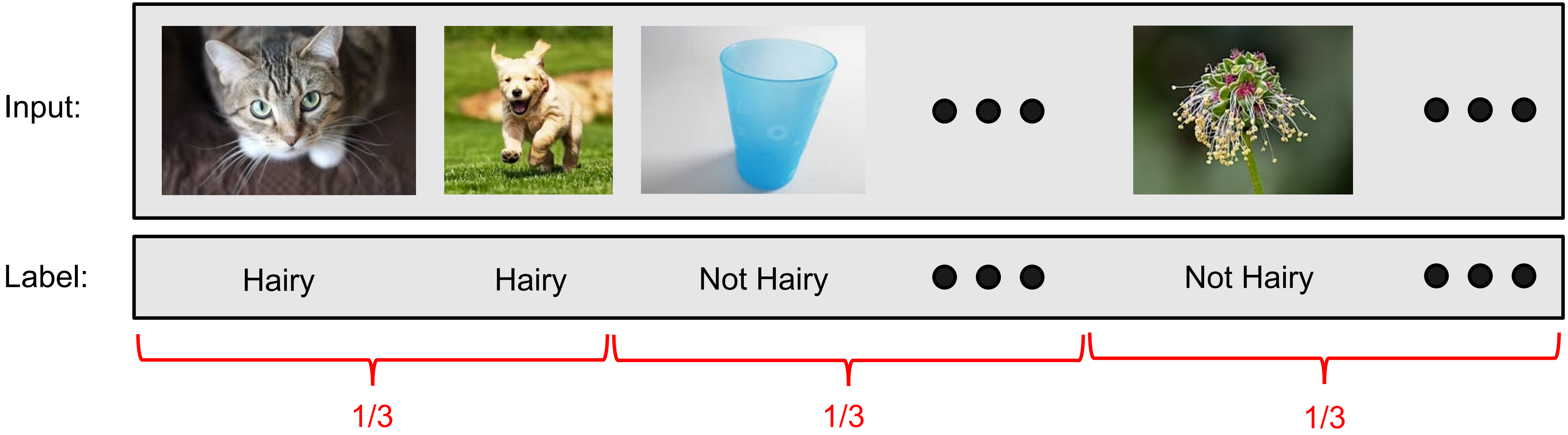
Cross-Validation: Limit Influence of Dataset Split



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Cross-Validation: Limit Influence of Dataset Split

e.g., 3-fold cross-validation on training data



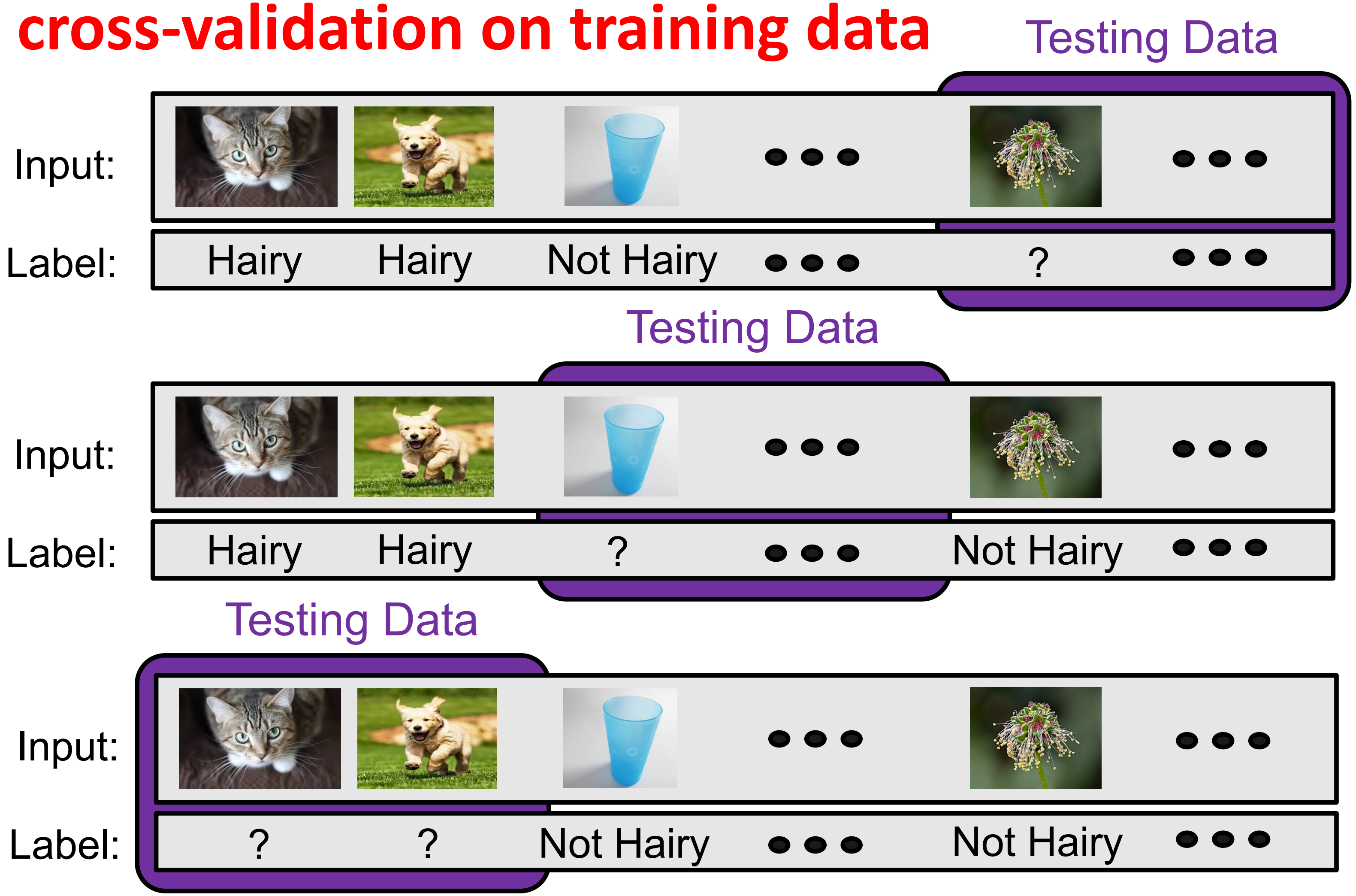
Cross-Validation: Limit Influence of Dataset Split

e.g., 3-fold cross-validation on training data

Fold 1:
– train on k–1 partitions
– test on k partitions

Fold 2:
– train on k–1 partitions
– test on k partitions

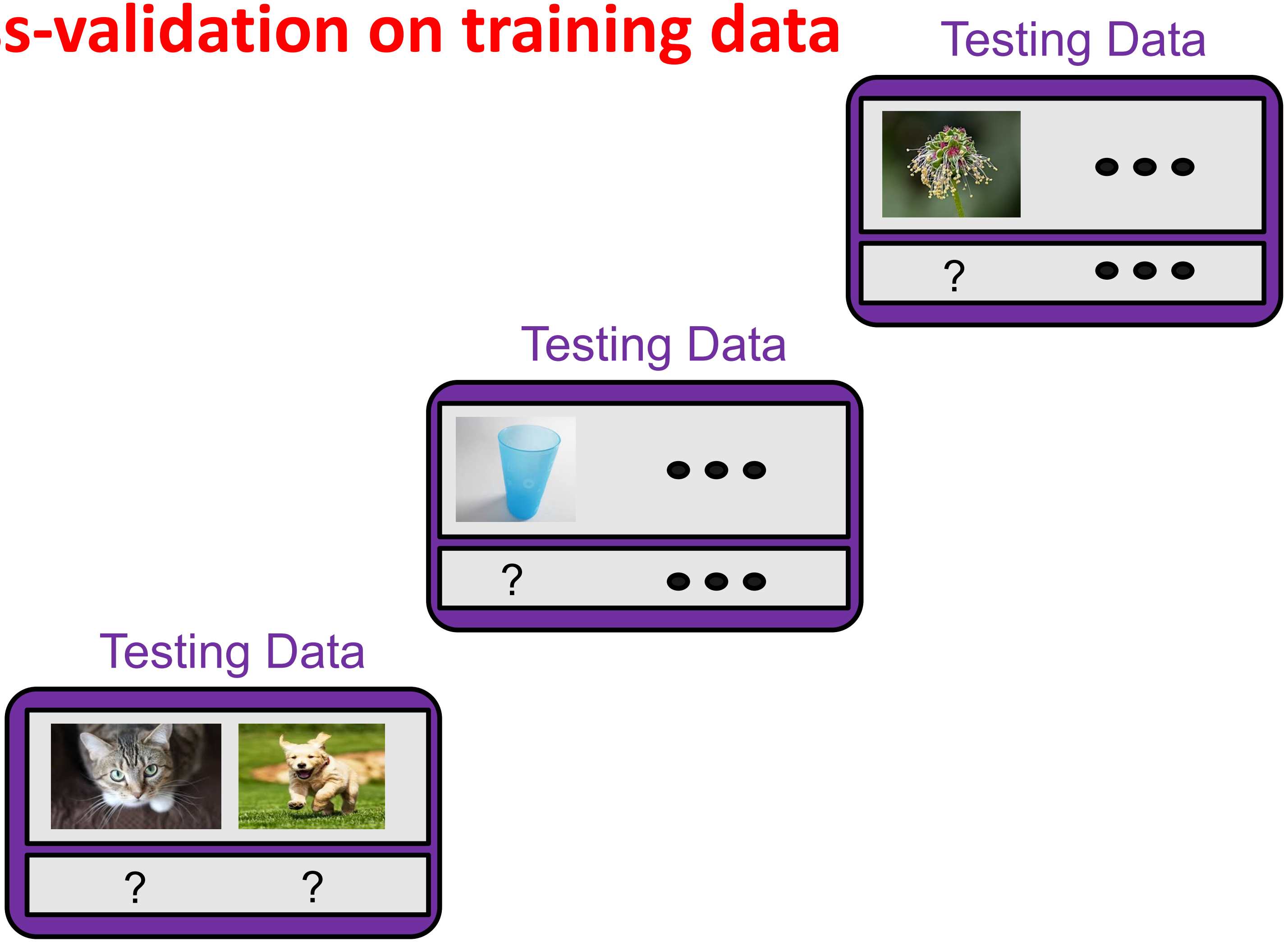
Fold 3:
– train on k–1 partitions
– test on k partitions



Cross-Validation: Limit Influence of Dataset Split

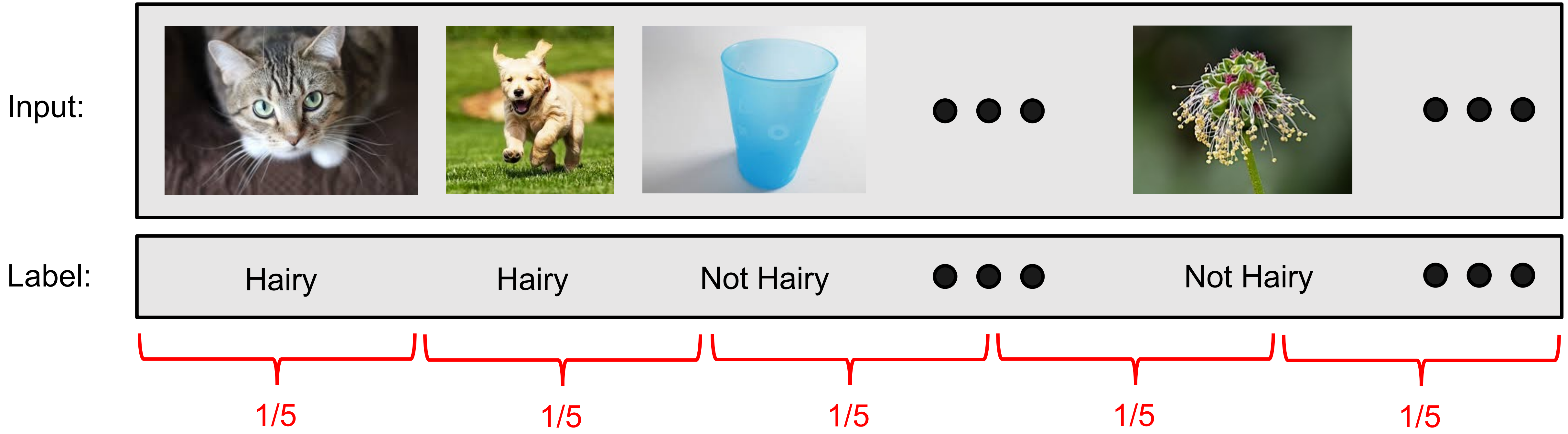
e.g., 3-fold cross-validation on training data

Model performance:
performance across all folds of “test” data



Cross-Validation: Limit Influence of Dataset Split

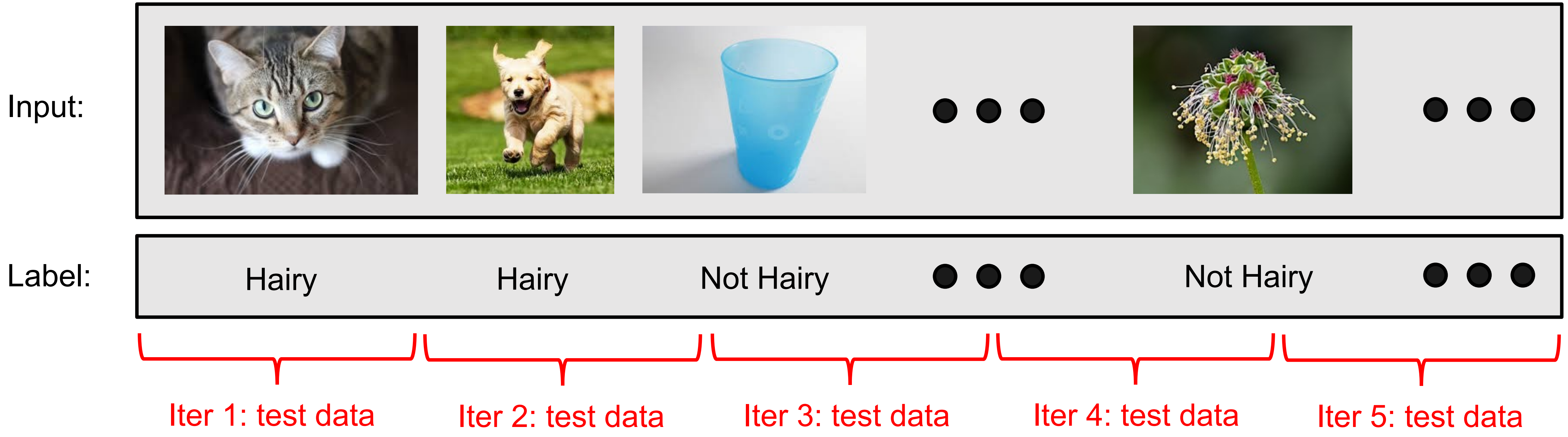
e.g., 5-fold cross-validation on training data



How many partitions of the data to create?

Cross-Validation: Limit Influence of Dataset Split

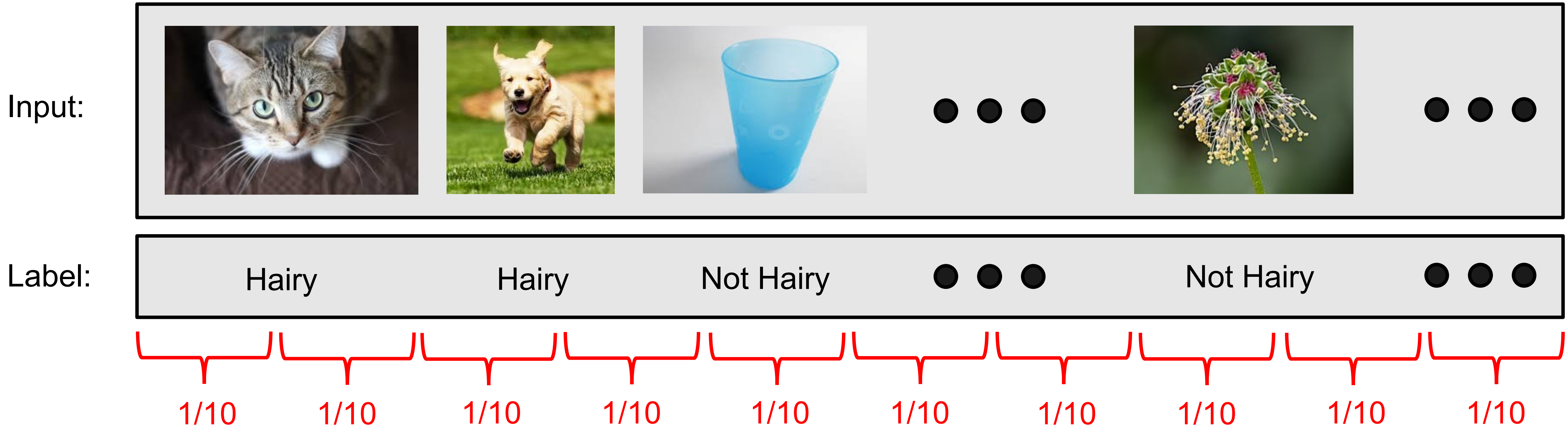
e.g., 5-fold cross-validation on training data



How many iterations of train & test to run?

Cross-Validation: Limit Influence of Dataset Split

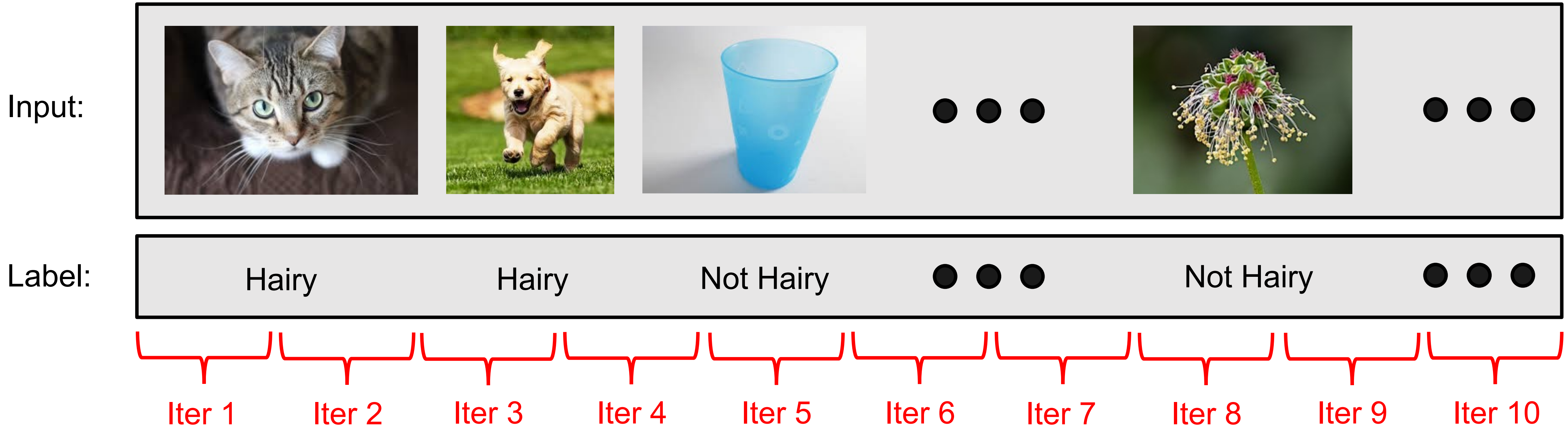
e.g., 10-fold cross-validation on training data



How many partitions of the data to create?

Cross-Validation: Limit Influence of Dataset Split

e.g., 10-fold cross-validation on training data



How many iterations of train & test to run?

Cross-Validation: Limit Influence of Dataset Split

e.g., k-fold cross-validation on training data

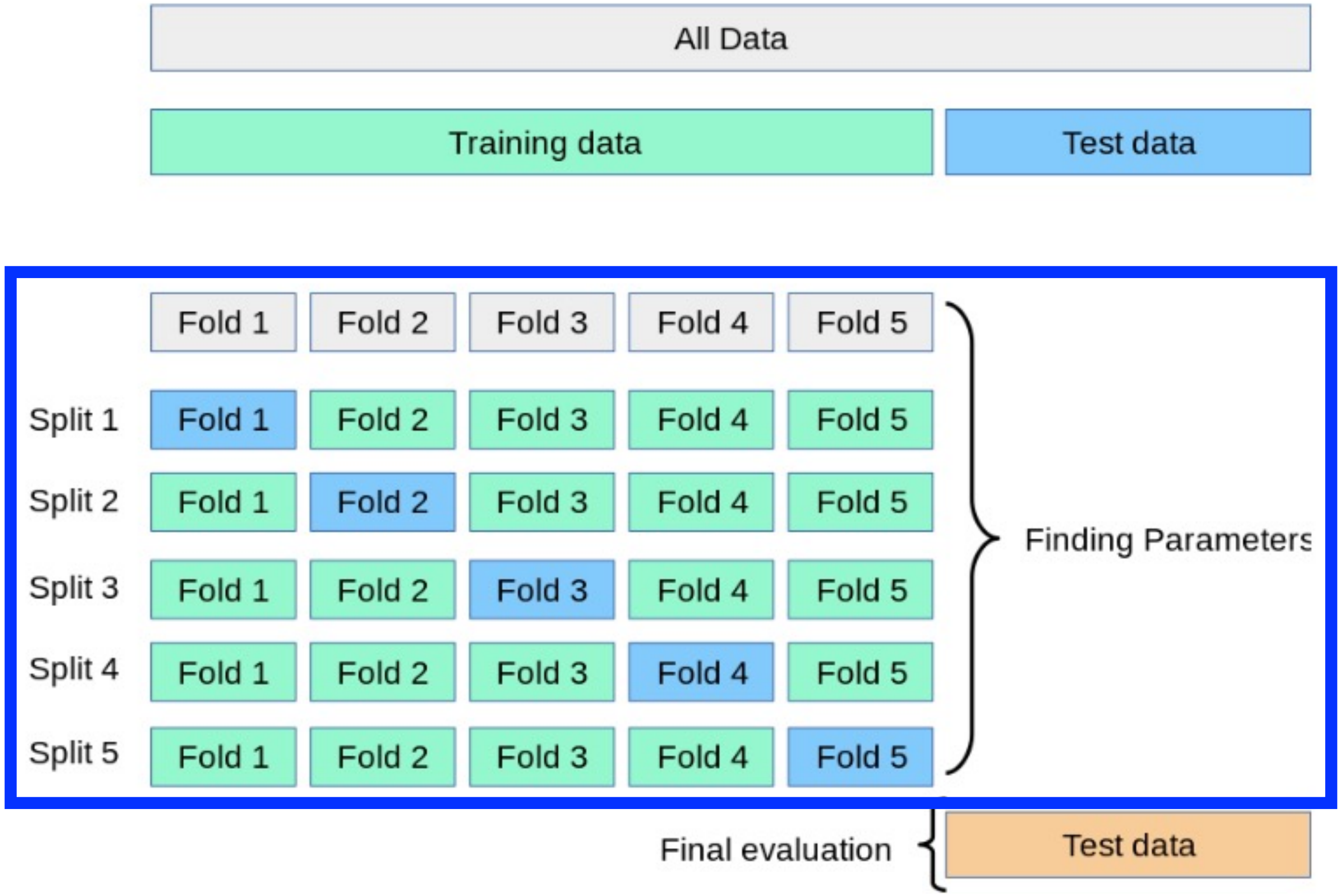
Input:

Label: Hairy Hairy Not Hairy ● ● ● Not Hairy ● ● ●

What are the (dis)advantages of using larger values for “k”?

Cross-Validation: Limit Influence of Dataset Split

Typically, select the hyperparameters that lead to the best results overall across all the folds

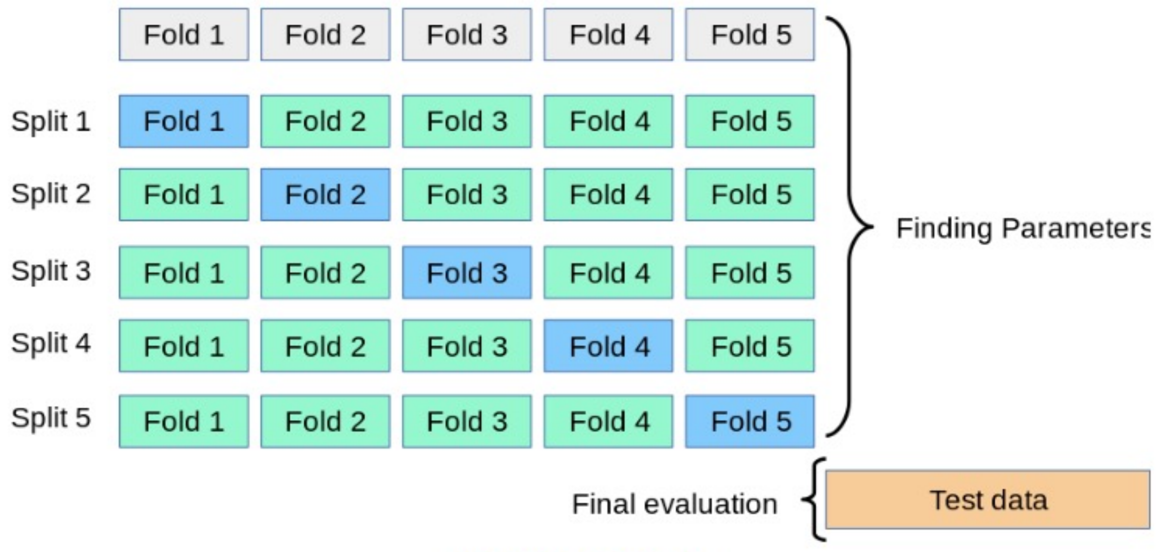


https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

Hyperparameter Tuning: Split Training Set So It Can Be Used to Test Different Hyperparameters

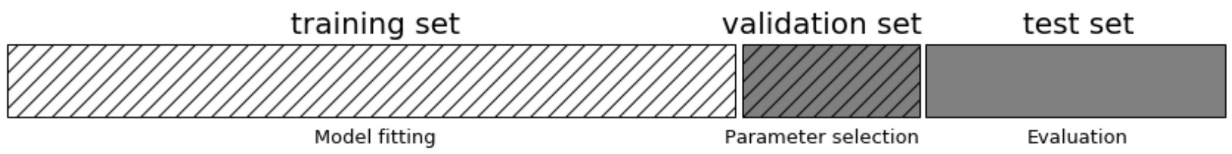
For statistically strong results:

Small training dataset: cross validation



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

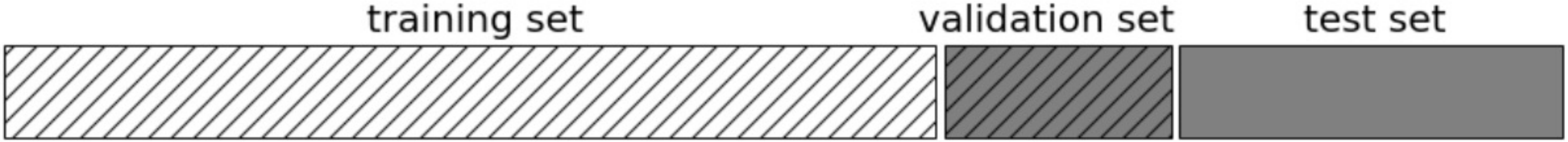
Else: train/validation split



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Validation Split

- Split training data into “train” and “validation” datasets



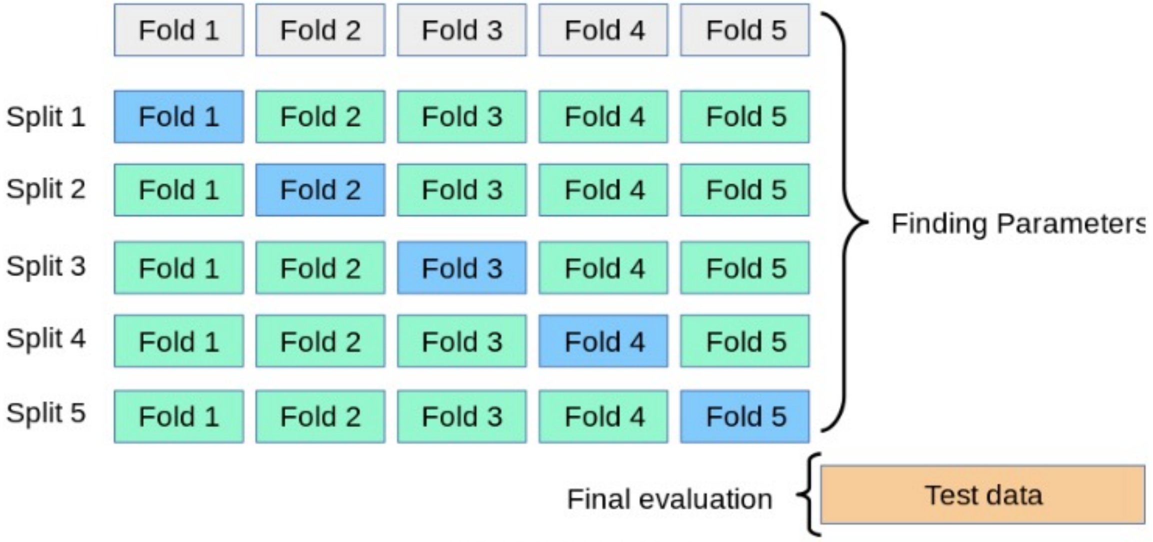
- Hyperparameter selection: test models trained with different hyperparameter values on the validation set to find the best one
- Final model: retrain using the model hyperparameters selected from validation set testing using the data in the training AND validation splits

https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Hyperparameter Tuning: Split Training Set So It Can Be Used to Test Different Hyperparameters

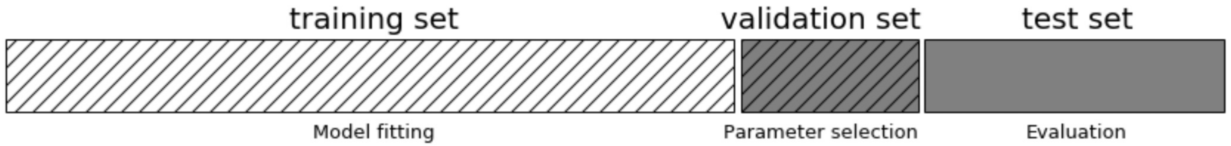
For statistically strong results:

Small training dataset: cross validation



https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/

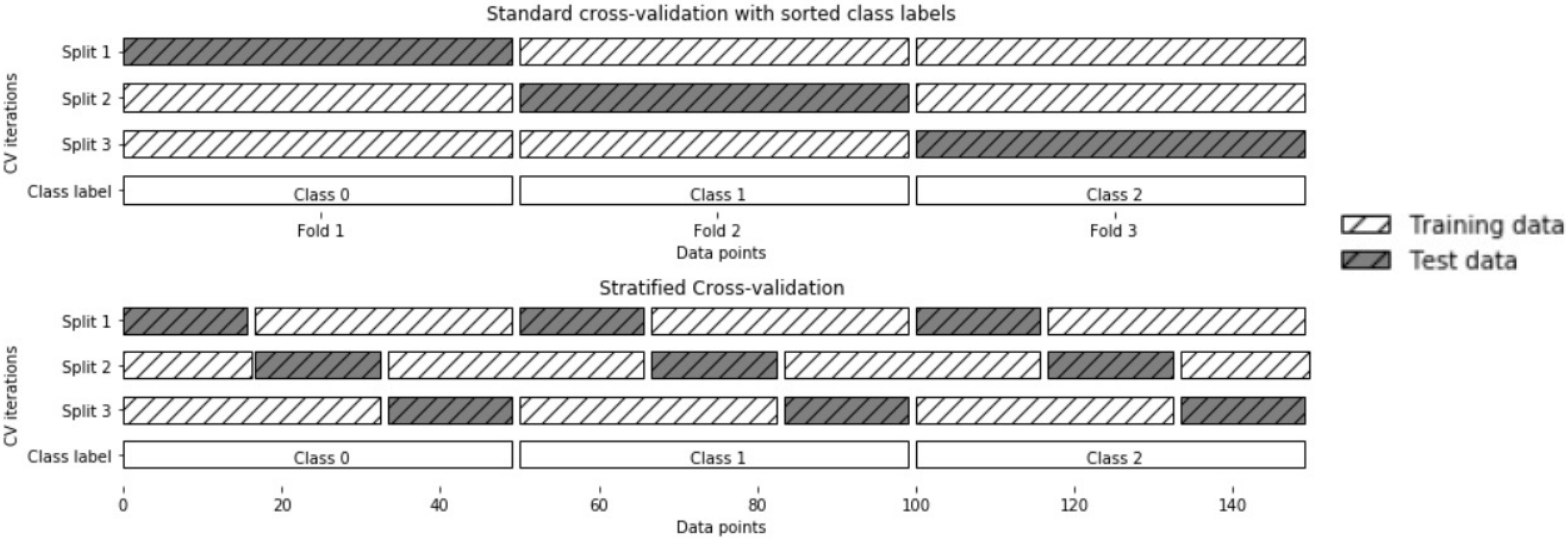
Else: train/validation split



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

Stratified Dataset Splits

- Preserve frequencies of each category in each dataset split; e.g.,



https://github.com/amueller/introduction_to_ml_with_python/blob/master/05-model-evaluation-and-improvement.ipynb

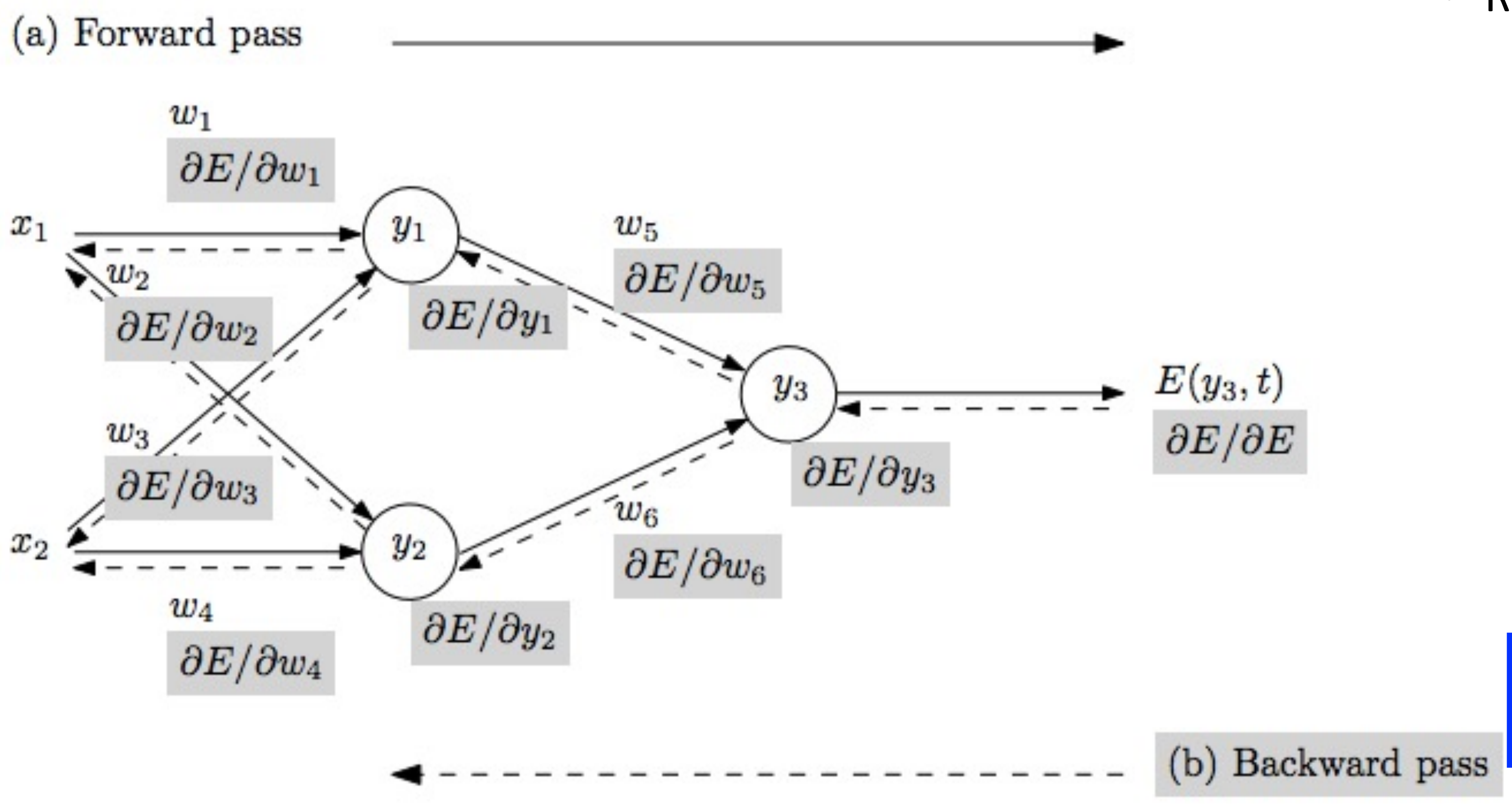
Today's Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

Challenge: Train Faster!!!

Algorithm training can take hours, days, weeks, months, or more with big data and so many parameters…

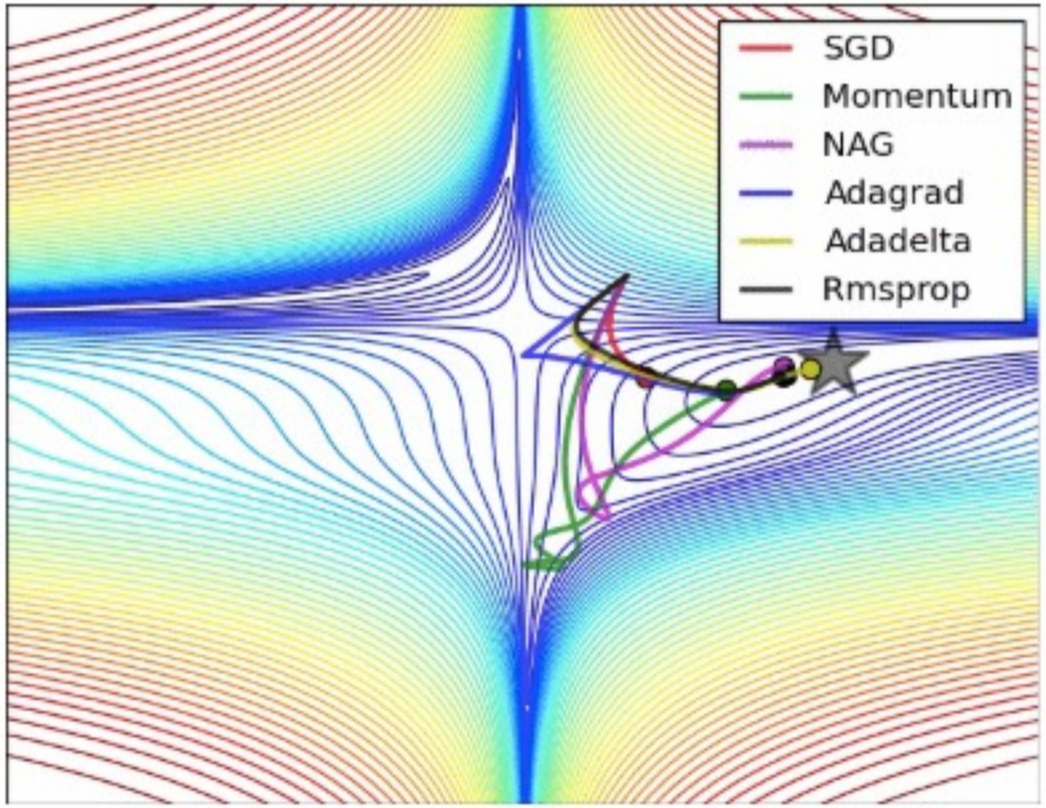
Recall: How Neural Networks Learn

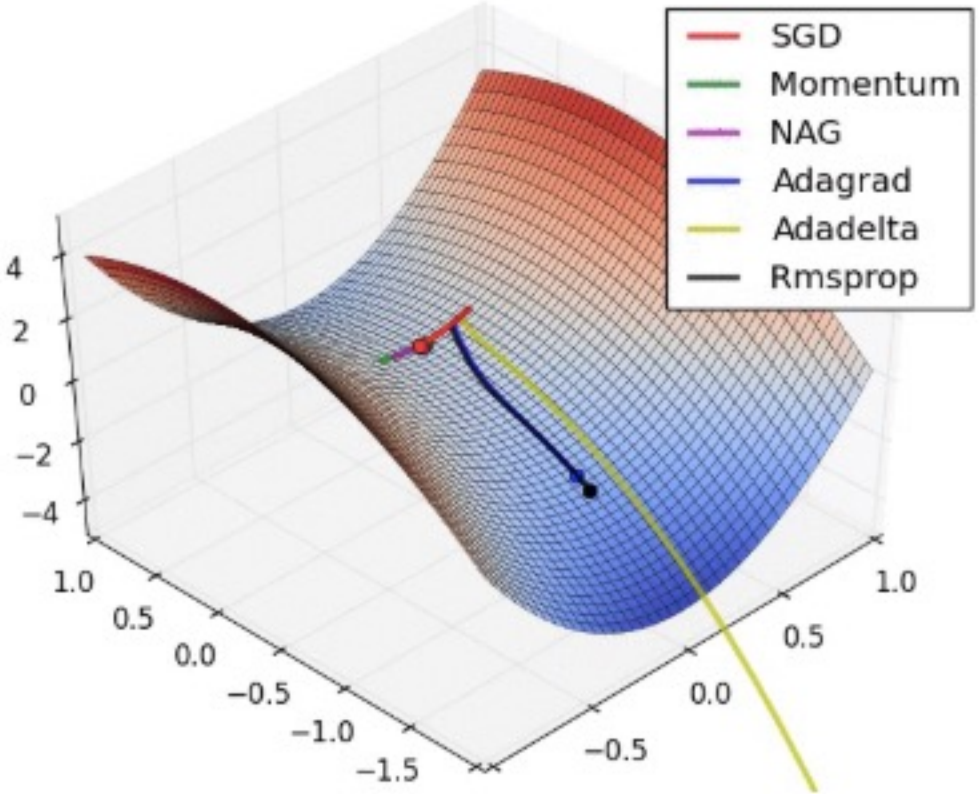


- Repeat until stopping criterion met:
  1. **Forward pass**: propagate training data through model to make prediction
  2. Quantify the dissatisfaction with a model's results on the training data
  3. **Backward pass**: using predicted output, calculate gradients backward to assign blame to each model parameter
  4. Update each parameter using calculated gradients

Figure from: Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, Jeffrey Mark Siskind; Automatic Differentiation in Machine Learning: a Survey; 2018

Train Faster: How to Update Using Gradient?



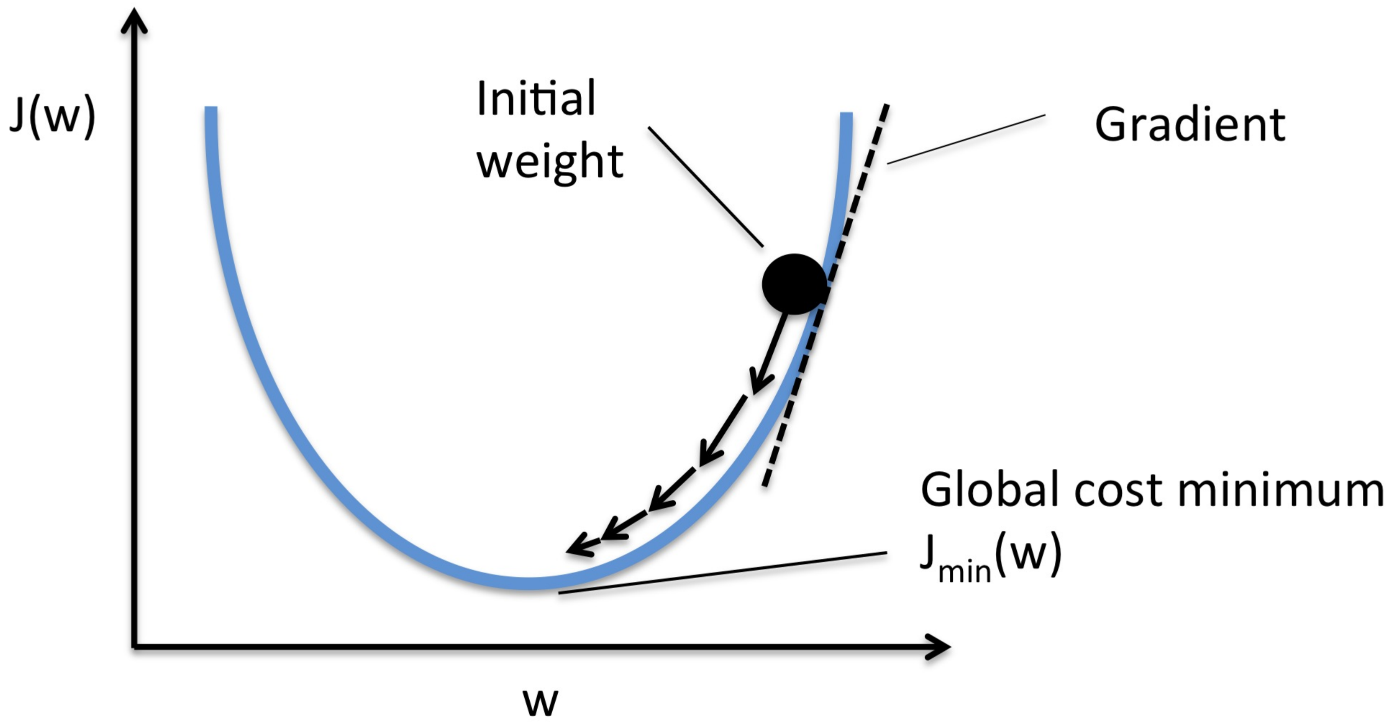


- Demo at http://cs231n.github.io/neural-networks-3/#update

Train Faster: How to Update Using Gradient?

- Vanilla Approach: `x += - learning_rate * dx` (Parameters: `x`; Gradient: `dx`)

Inefficient since steps get smaller as gradient gets smaller



http://cs231n.github.io/neural-networks-3/#update

Figure from: https://rasbt.github.io/mlxtend/user_guide/general_concepts/gradient-optimization/

Train Faster: How to Update Using Gradient?

- Momentum optimization:
  - Analogy: roll a ball down a hill and it will pick up momentum

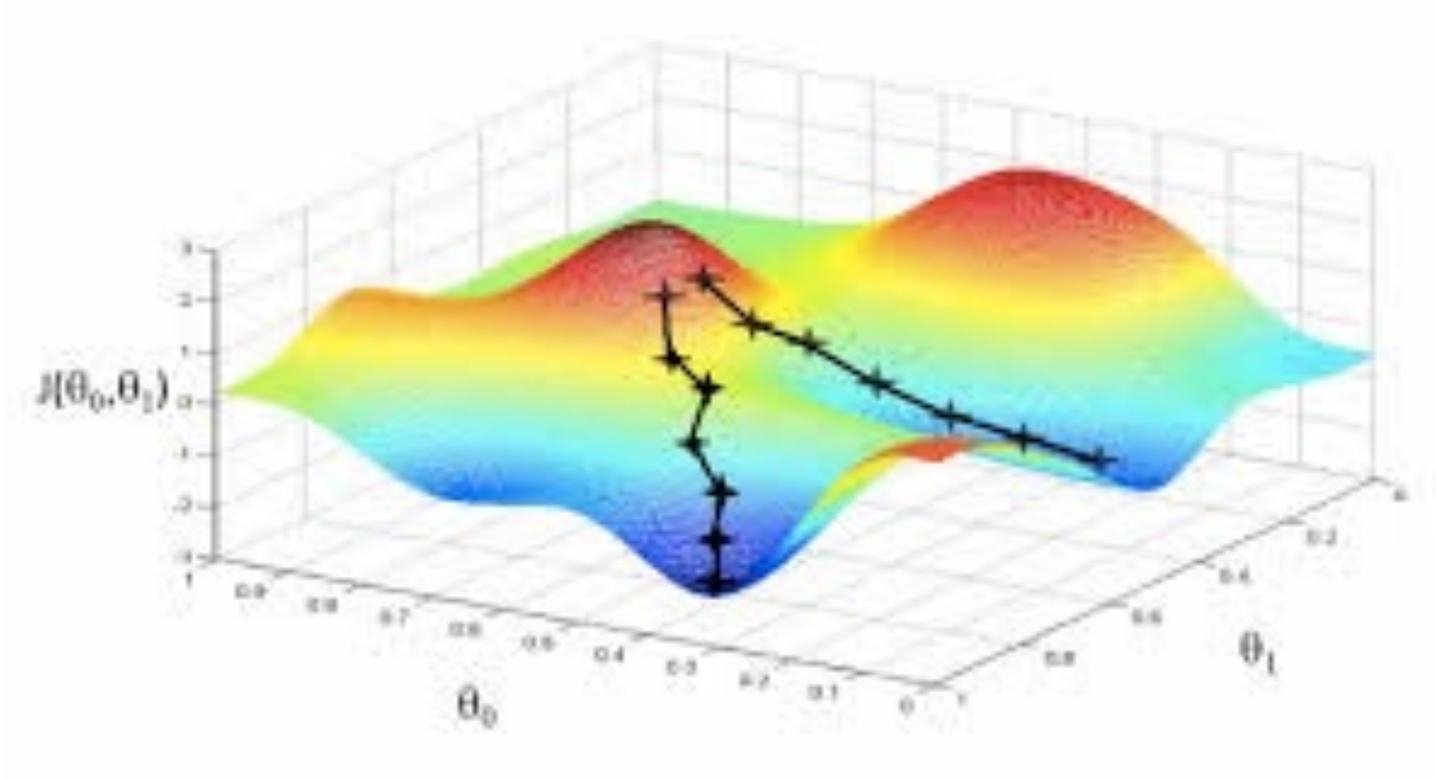


Figure from: https://medium.com/ai-society/hello-gradient-descent-ef74434bdfa5

Train Faster: How to Update Using Gradient?

- Momentum optimization:
  - Analogy: roll a ball down a hill and it will pick up momentum

Like friction; values range from 0 to 1 with larger being greater friction

Velocity vector captures cumulative direction of previous gradients; initialized to 0

Gradient not used for speed but instead acceleration

```
v = mu * v - learning_rate * dx # integrate velocity
x += v # integrate position
```

- What are advantages and disadvantages?
  - Can roll past local minima ☺
  - It may roll past optimum and oscillate around it ☹
  - Another hyperparameter to tune: mu ☹

http://cs231n.github.io/neural-networks-3/#update

Train Faster: How to Update Using Gradient?

- Step decay:
  - Reduce the learning rate by some factor every few epochs

- Exponential decay

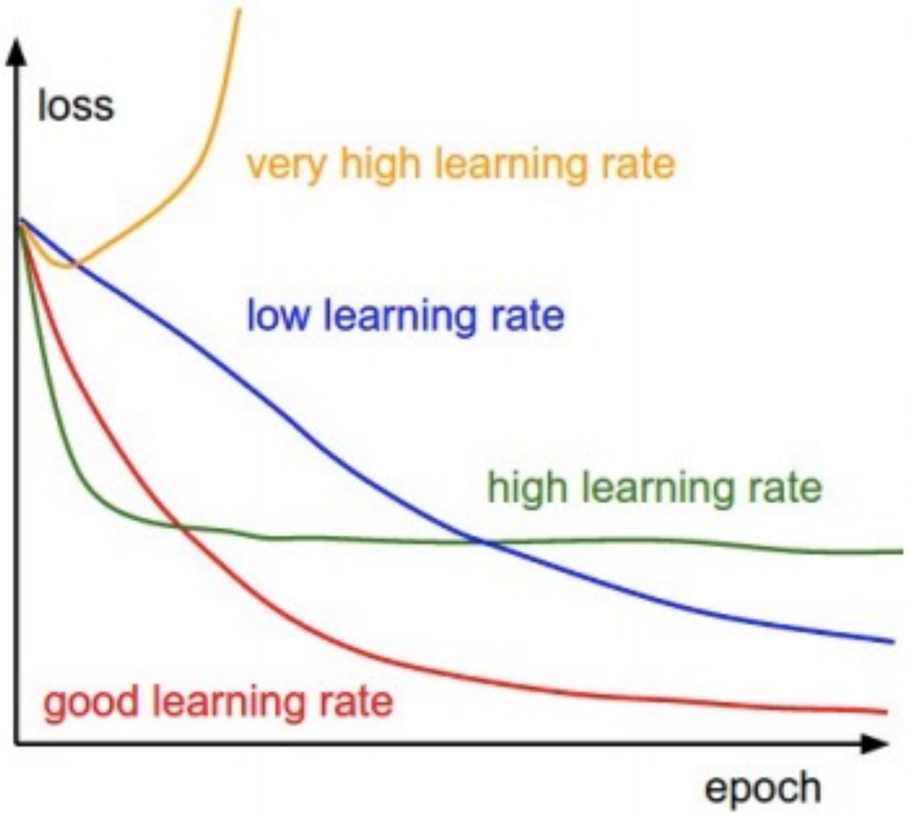
- 1/t decay

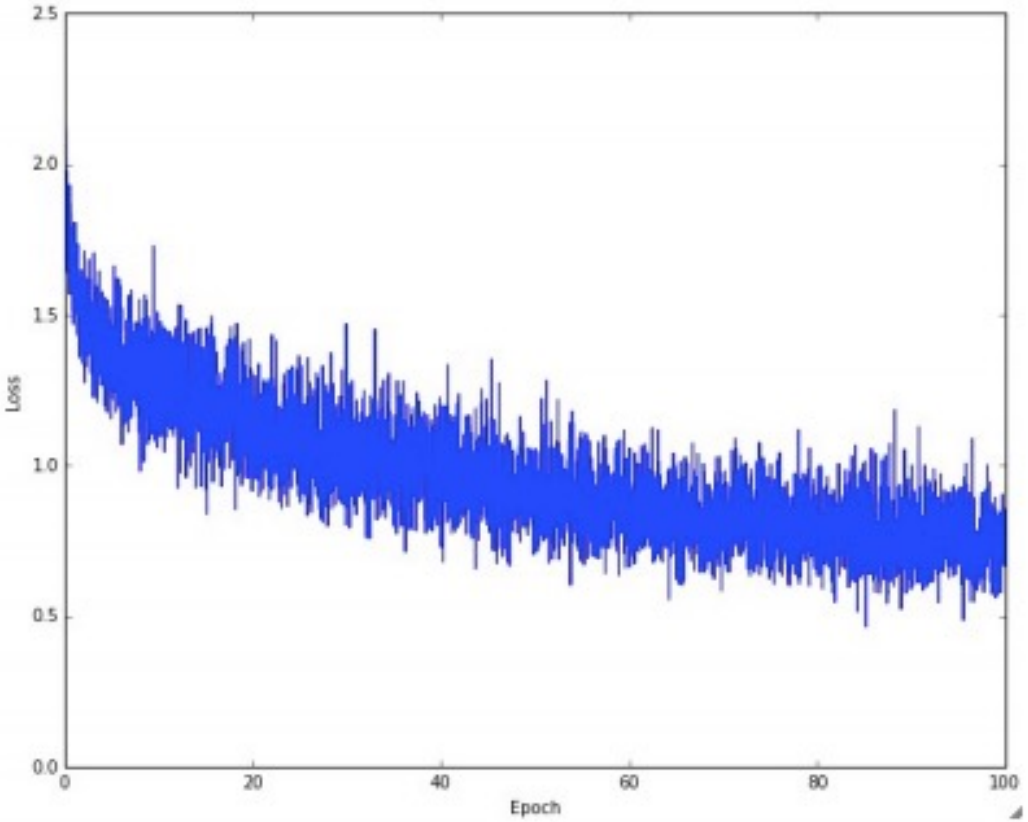
- Adapt learning rate per-parameter
- e.g., AdaGrad, RMSprop, and Adam (i.e., adaptive momentum – very popular in practice)

http://cs231n.github.io/neural-networks-3/#update

Monitor Loss/Error During Training

- What should happen to the loss function value during training?





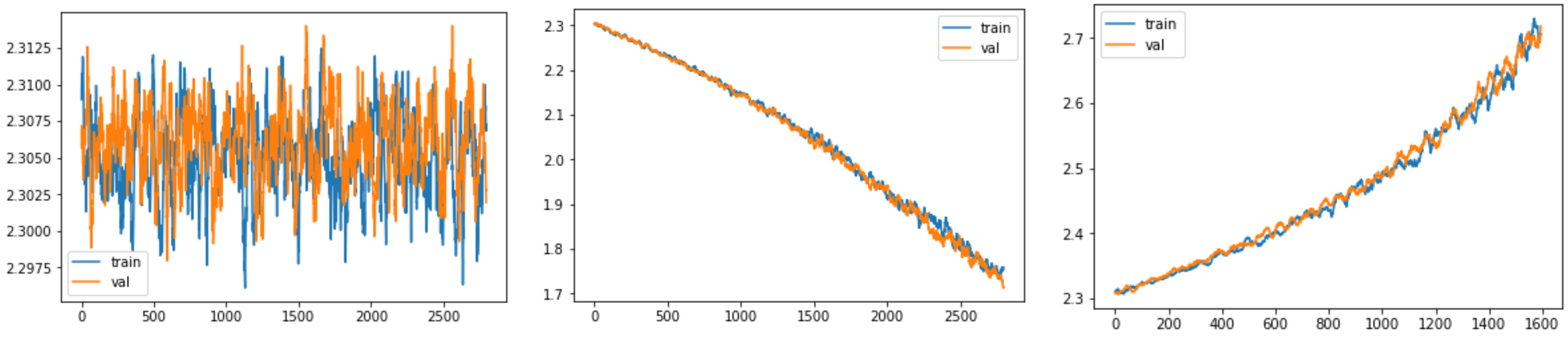
https://cs231n.github.io/neural-networks-3/#update

Monitor Loss/Error: It Should Shrink

And have fun along the way: https://lossfunctions.tumblr.com/

Discussion: Given These Loss Curves, What Do You Think Is Happening With Learning?

(a) (b) (c)



Today’s Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

Today’s Topics

- Universal approximation theorem
- Selecting model capacity: avoid overfitting and underfitting
- Selecting model hyperparameters
- Learning efficiently: optimization methods
- Programming tutorial

The End