# Lecture 13: Evaluation

COMP 343, Spring 2022
Victoria Manfredi

WESLEYAN
UNIVERSITY

**Acknowledgements:** These slides are based primarily on material from the book Machine Learning by Tom Mitchell (and associated slides), the book Machine Learning, An Applied Mathematics Introduction by Paul Wilmott, slides created by Vivek Srikumar (Utah), Dan Roth (Penn), Shuiwang Ji (Texas A&M), Mark Schmidt (UBC), Julia Hockenmaier (Illinois Urbana-Champaign), Jessica Wu (Harvey Mudd) and C. David Page (U of Wisconsin-Madison)

# Today's Topics

## Midterm

- Wednesday March 30

## Loss minimization

- Overview
- How to minimize?

## Evaluation

- Bias vs. variance
- Other metrics besides accuracy

vumanfredi@wesleyan.edu

Masks are still required in this class and my office hours

# Learning as Loss Minimization

## LOSS FUNCTIONS

# Learning as loss minimization

## The setup

- Examples $\mathbf{x}$ with labels $y$ drawn from a fixed, unknown distribution $P(X, Y)$
- Hidden oracle classifier $f$ labels examples
- We wish to find a hypothesis $h$ that mimics $f$

# Learning as loss minimization

## The setup

- Examples $\mathbf{x}$ with labels $y$ drawn from a fixed, unknown distribution $P(X, Y)$
- Hidden oracle classifier $f$ labels examples
- We wish to find a hypothesis $h$ that mimics $f$

## The ideal situation

- Define a function $L$ that penalizes bad hypotheses
- **Learning**: pick a function $h \in H$ to minimize expected loss

$$\min_{h \in H} \sum_{(\mathbf{x},y) \in D} L(f(\mathbf{x}), h(\mathbf{x}))P(X = \mathbf{x}, Y = y)$$

$L(f(\mathbf{x}), h(\mathbf{x}))$ is the loss of function $h$ on example $\mathbf{x}$
when the true label or prediction for $\mathbf{x}$ is given by $f(\mathbf{x})$

$D$ is set of all possible $(\mathbf{x}, y)$ pairs

# Learning as loss minimization

## The setup

- Examples $\mathbf{x}$ with labels $y$ drawn from a fixed, unknown distribution $P(X, Y)$
- Hidden oracle classifier $f$ labels examples
- We wish to find a hypothesis $h$ that mimics $f$

## The ideal situation

- Define a function $L$ that penalizes bad hypotheses
- **Learning**: pick a function $h \in H$ to minimize expected loss

$$\min_{h \in H} \sum_{(\mathbf{x},y) \in D} L(f(\mathbf{x}), h(\mathbf{x}))P(X = \mathbf{x}, Y = y)$$

$$\min_{\mathbf{w}} \sum_{(\mathbf{x},y) \in D} L(y, \mathbf{w}^T\mathbf{x})P(X = \mathbf{x}, Y = y)$$

$D$ is set of all possible $(\mathbf{x}, y)$ pairs

# Learning as loss minimization

## The setup

- Examples $\mathbf{x}$ with labels $y$ drawn from a fixed, unknown distribution $P(X, Y)$
- Hidden oracle classifier $f$ labels examples
- We wish to find a hypothesis $h$ that mimics $f$

## The ideal situation

- Define a function $L$ that penalizes bad hypotheses
- **Learning**: pick a function $h \in H$ to minimize expected loss

$$\min_{h \in H} \sum_{(\mathbf{x}, y) \in D} L(f(\mathbf{x}), h(\mathbf{x})) P(X = \mathbf{x}, Y = y)$$

We cannot minimize this since we don't know $P(X, Y)$!

$D$ is set of all possible $(\mathbf{x}, y)$ pairs

# Empirical loss minimization

Learning = minimize empirical loss on the training set

$$\min_{h \in H} \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

where $m$ is # of training examples $\{(\mathbf{x}_i y_i)\}_{i=1,m}$

# Empirical loss minimization

Learning = minimize empirical loss on the training set

$$\min_{h \in H} \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

where $m$ is # of training examples $\{(\mathbf{x}_i y_i)\}_{i=1,m}$

Is there a problem here?

# Empirical loss minimization

Learning = minimize empirical loss on the training set

$$\min_{h \in H} \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

where $m$ is # of training examples $\{(\mathbf{x}_i y_i)\}_{i=1,m}$

Is there a problem here? Overfitting!

# Regularized empirical loss minimization

We need something that biases the learner towards simpler hypotheses

- Achieved using a function called a regularizer, which penalizes complex hypotheses

Learning:

Trade-off parameter

$$\min_{h \in H} \left( \text{regularizer}(h) + C \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i)) \right)$$

# Regularized empirical loss minimization

We need something that biases the learner towards simpler hypotheses

- Achieved using a function called a regularizer, which penalizes complex hypotheses

Learning:

Trade-off parameter

$$\min_{h \in H} \left( \text{regularizer}(h) + C \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i)) \right)$$

With linear classifiers:

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \frac{1}{m} \sum_{i=1:m} L(y_i, \mathbf{w}^T \mathbf{x}_i)$$

If this gets too large, outweighs gains in minimizing loss.

Helps with generalization: don't give large weight to feature unless good evidence it is useful

# Regularized empirical loss minimization

We need something that biases the learner towards simpler hypotheses

- Achieved using a function called a regularizer, which penalizes complex hypotheses

Learning:

$$\min_{h \in H} \left( \text{regularizer}(h) + C \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i)) \right)$$

With linear classifiers:

$$\min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \frac{1}{m} \sum_{i=1:m} L(y_i, \mathbf{w}^T \mathbf{x}_i)$$

We'll talk about regularization again when we talk about linear regression

# What is a loss function?

Loss functions should penalize mistakes

- We are minimizing average loss over the training data

$$\min_{h \in H} \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

# What is a loss function?

Loss functions should penalize mistakes

- We are minimizing average loss over the training data

$$\min_{h \in H} \frac{1}{m} \sum_{i=1:m} L(h(\mathbf{x}_i), f(\mathbf{x}_i))$$

What is the ideal loss function for classification?

- What loss function would you minimize if computation were free?

# 0-1 loss

Simplest loss function counts # of mistakes, aka 0-1 loss:

- Penalizes classification mistakes between true label $y$ and prediction $y'$

$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y \neq y \\ 0 & \text{if } y = y \end{cases}$$

# 0-1 loss

Simplest loss function counts # of mistakes, aka 0-1 loss:

- Penalizes classification mistakes between true label $y$ and prediction $y'$

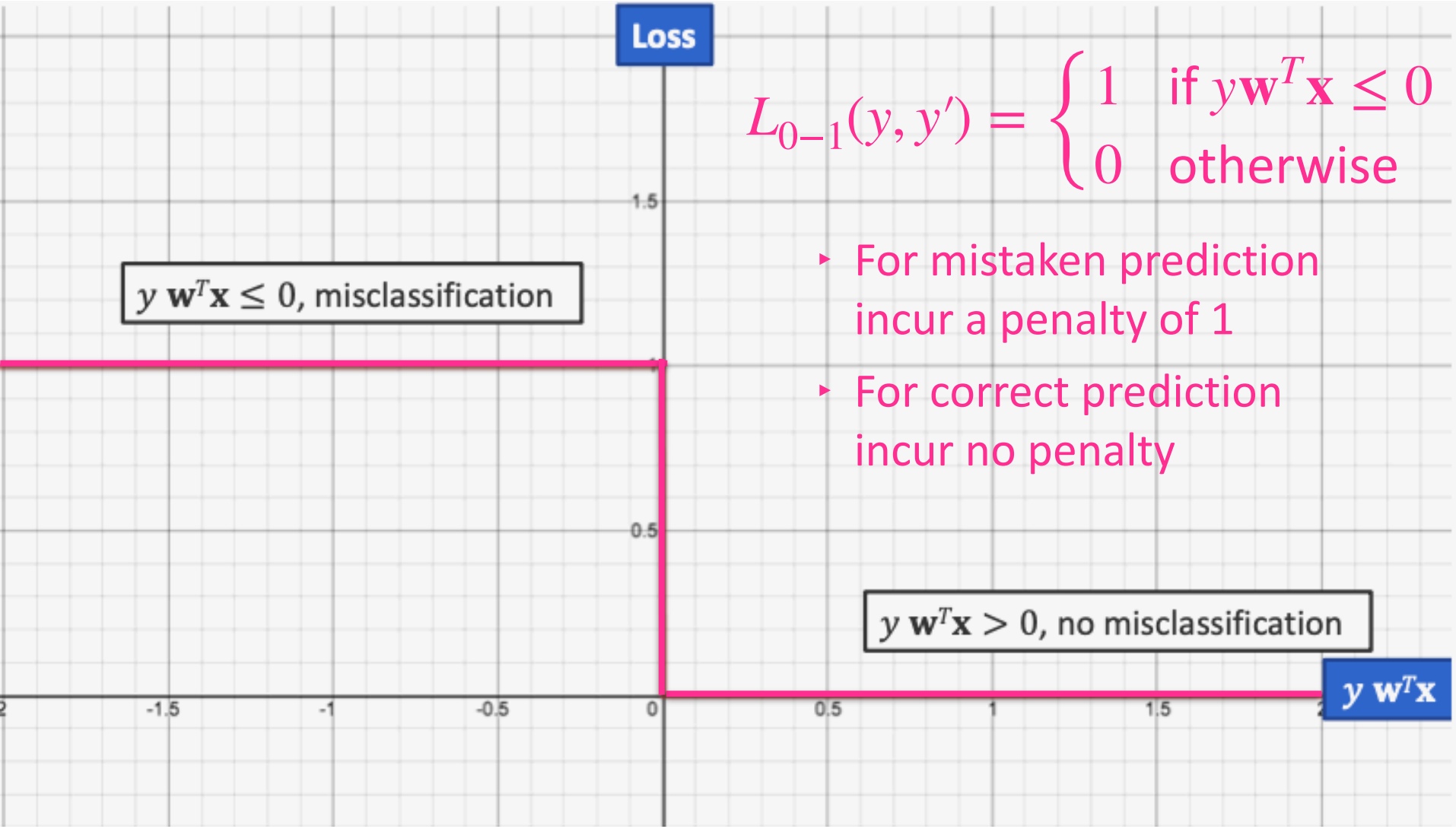
$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y \neq y \\ 0 & \text{if } y = y \end{cases}$$

For linear classifiers the prediction is $y' = \text{sgn}(\mathbf{w}^T\mathbf{x})$

- Mistake if $y\mathbf{w}^T\mathbf{x} \leq \mathbf{0}$

$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y\mathbf{w}^T\mathbf{x} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

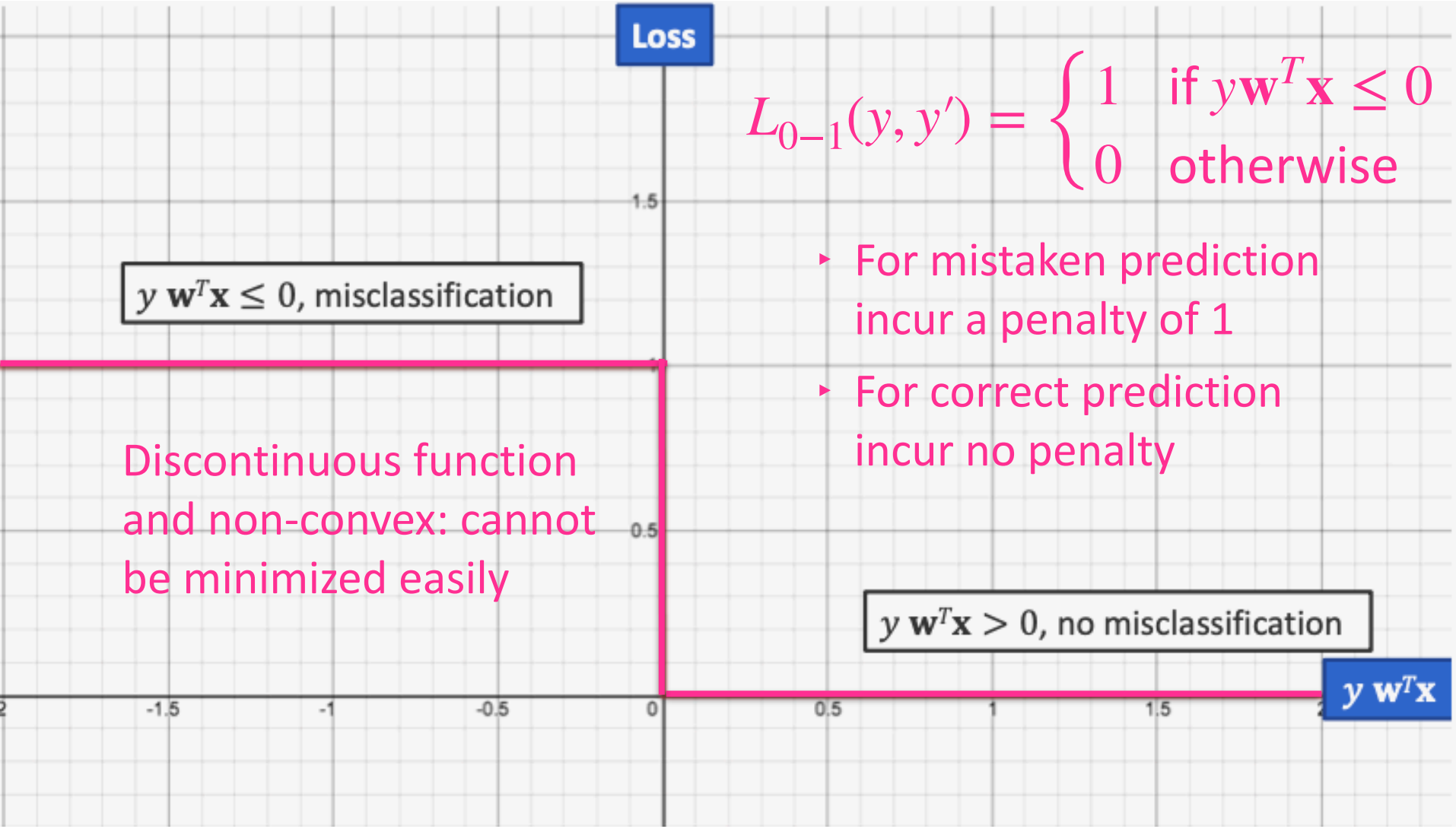
# 0-1 loss




$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y\mathbf{w}^T\mathbf{x} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

- For mistaken prediction incur a penalty of 1
- For correct prediction incur no penalty

# 0-1 loss




$$L_{0-1}(y, y') = \begin{cases} 1 & \text{if } y\mathbf{w}^T\mathbf{x} \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

- For mistaken prediction incur a penalty of 1
- For correct prediction incur no penalty

Discontinuous function and non-convex: cannot be minimized easily

# Problems with 0-1 loss

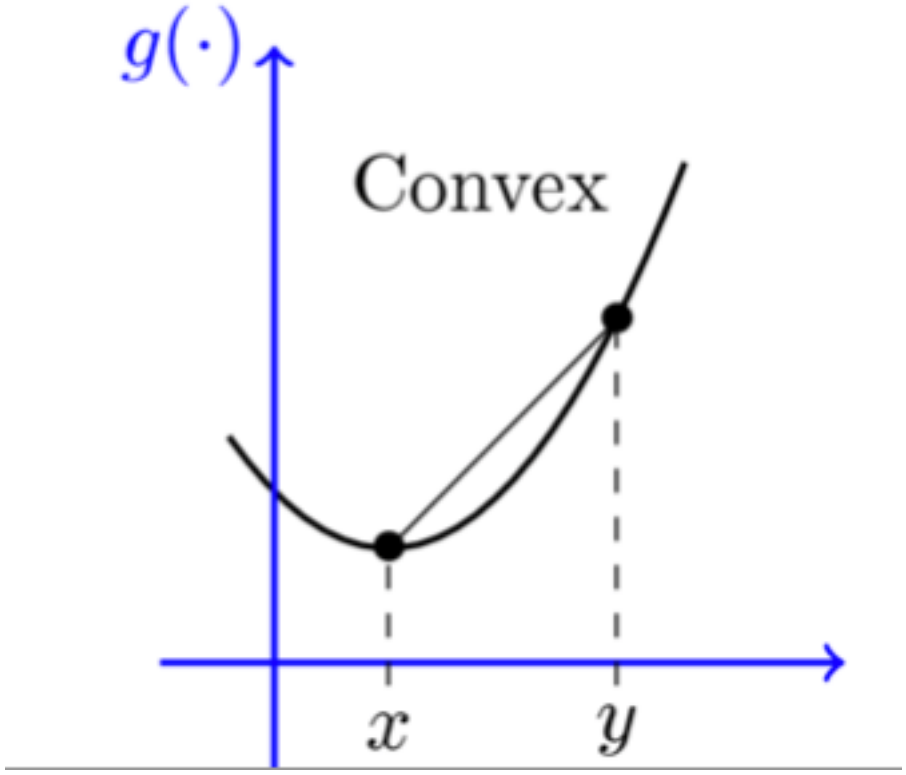
- 0-1 loss incurs the same loss value of 1 for all wrong predictions, no matter how far a wrong prediction is from the hyperplane
- Knowing the current value of loss function doesn't indicate how to modify current weights to further improve
- Computationally intractable to minimize 0-1 loss function: with $m$ examples need to check $2^m$ permutations (since can't minimize directly)
- What can we do? Make some assumptions: minimize "surrogate function" that is good enough

# A solution of sorts

Minimize a new function, a convex upper bound of classification error

# Convexity

A convex function has only one minimum




Intuitively, a function is convex if the line segment between any two points on the function is not below the function

# A solution of sorts

Minimize a new function, a convex upper bound of classification error

Perceptron loss

- Penalizes each wrong prediction by extent of violation
- The perceptron loss function is defined as

$$\frac{1}{m}\sum_{i=1:m} L_P(y_i, f(\mathbf{x}_i))$$

where $L_p(y_i, f(\mathbf{x}_i)) = \max(0, -y_i \mathbf{w}^T \mathbf{x}_i)$

# A solution of sorts

Minimize a new function, a convex upper bound of classification error

Perceptron loss

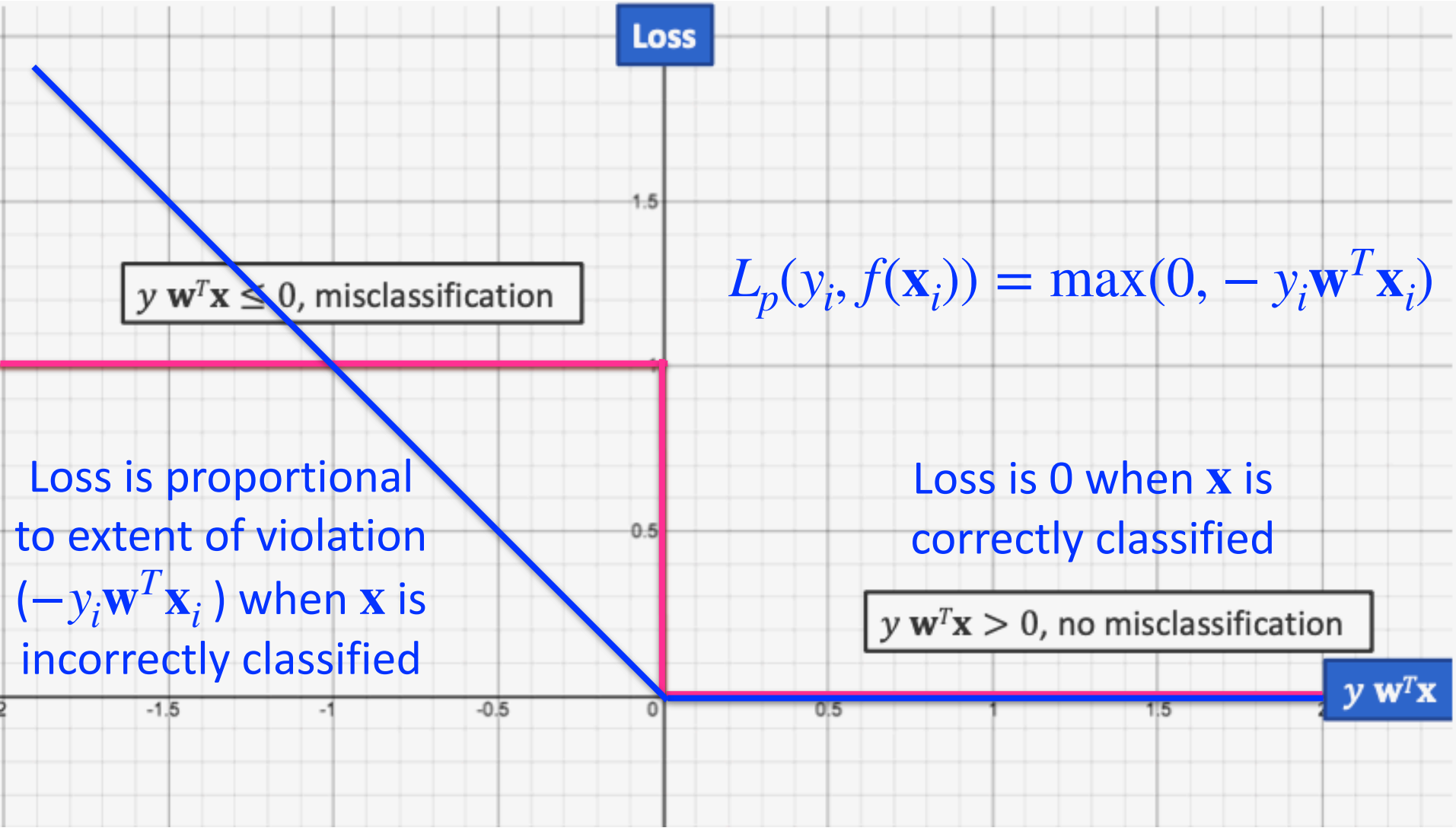
- Penalizes each wrong prediction by extent of violation
- The perceptron loss function is defined as

$$\frac{1}{m}\sum_{i=1:m} L_P(y_i, f(\mathbf{x}_i))$$

where $L_p(y_i, f(\mathbf{x}_i)) = \max(0, -y_i \mathbf{w}^T \mathbf{x}_i)$

- Note: max of constant and linear function is convex function

# Perceptron loss



# Square loss

$y \in \{-1, 1\}$

The square loss function is commonly used for regression problems

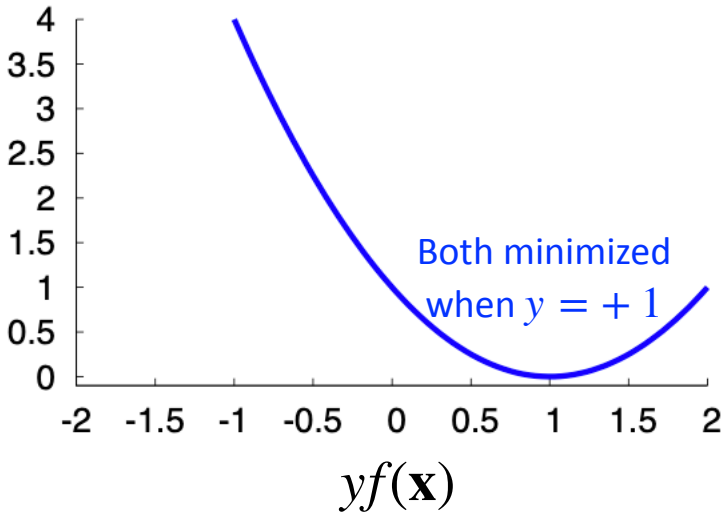
$$L_S(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$$

Square loss can also be used for binary classification problems

$$L_S(y, f(\mathbf{x})) = (1 - yf(\mathbf{x}))^2$$

Square loss tends to penalize wrong predictions excessively

Square loss as a function of $yf(\mathbf{x})$



## Learning as Loss Minimization

# HOW TO MINIMIZE A LOSS FUNCTION?

# What is our goal?

Most machine learning algorithms are some combination of a loss function + an algorithm for finding a local minimum.

Our goal is to **minimize the loss function**

- E.g., find a set of values for $\mathbf{w}$ for which the loss from the loss function is a minimum

# What is our goal?

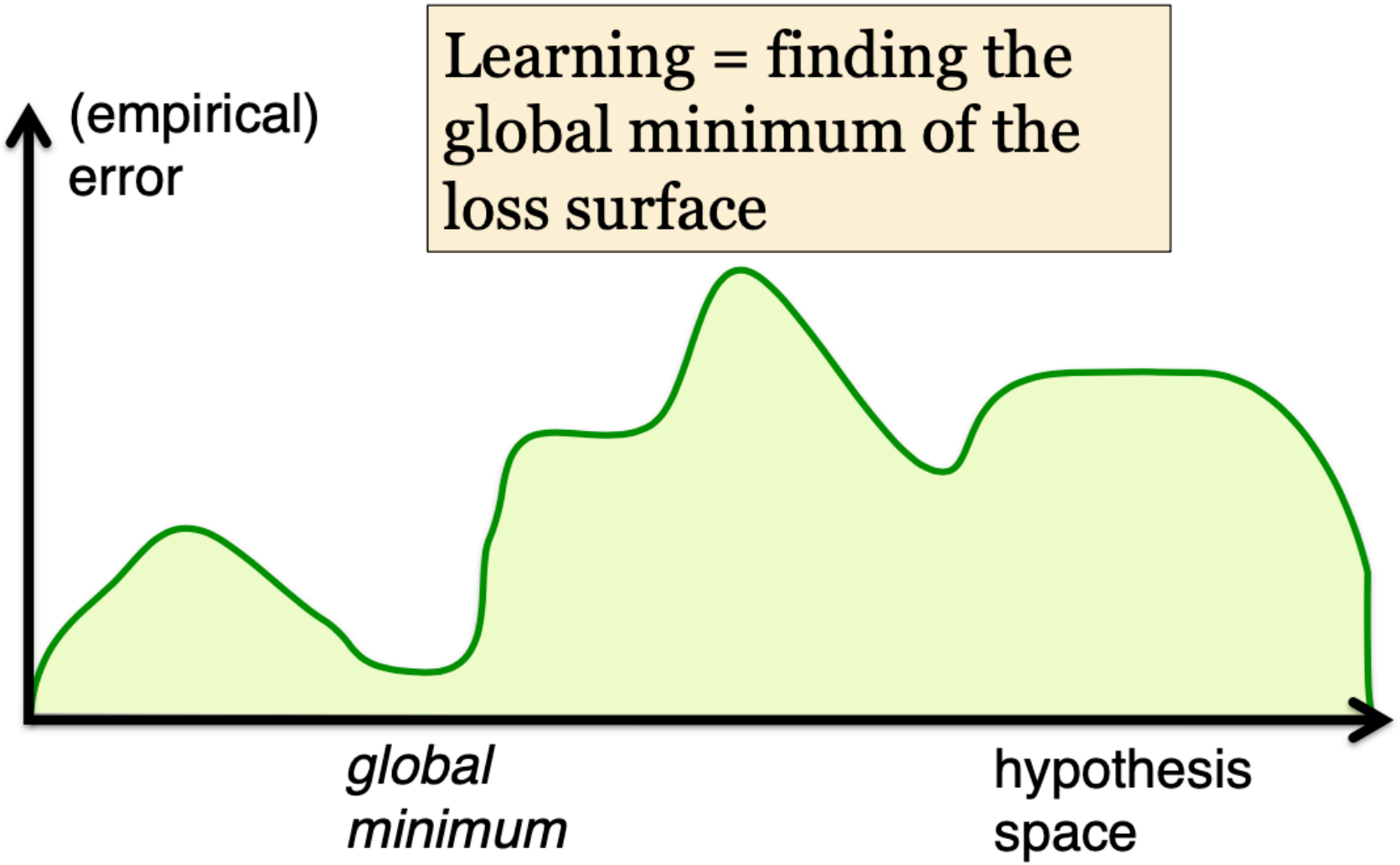
Most machine learning algorithms are some combination of a loss function + an algorithm for finding a local minimum.

Our goal is to **minimize the loss function**

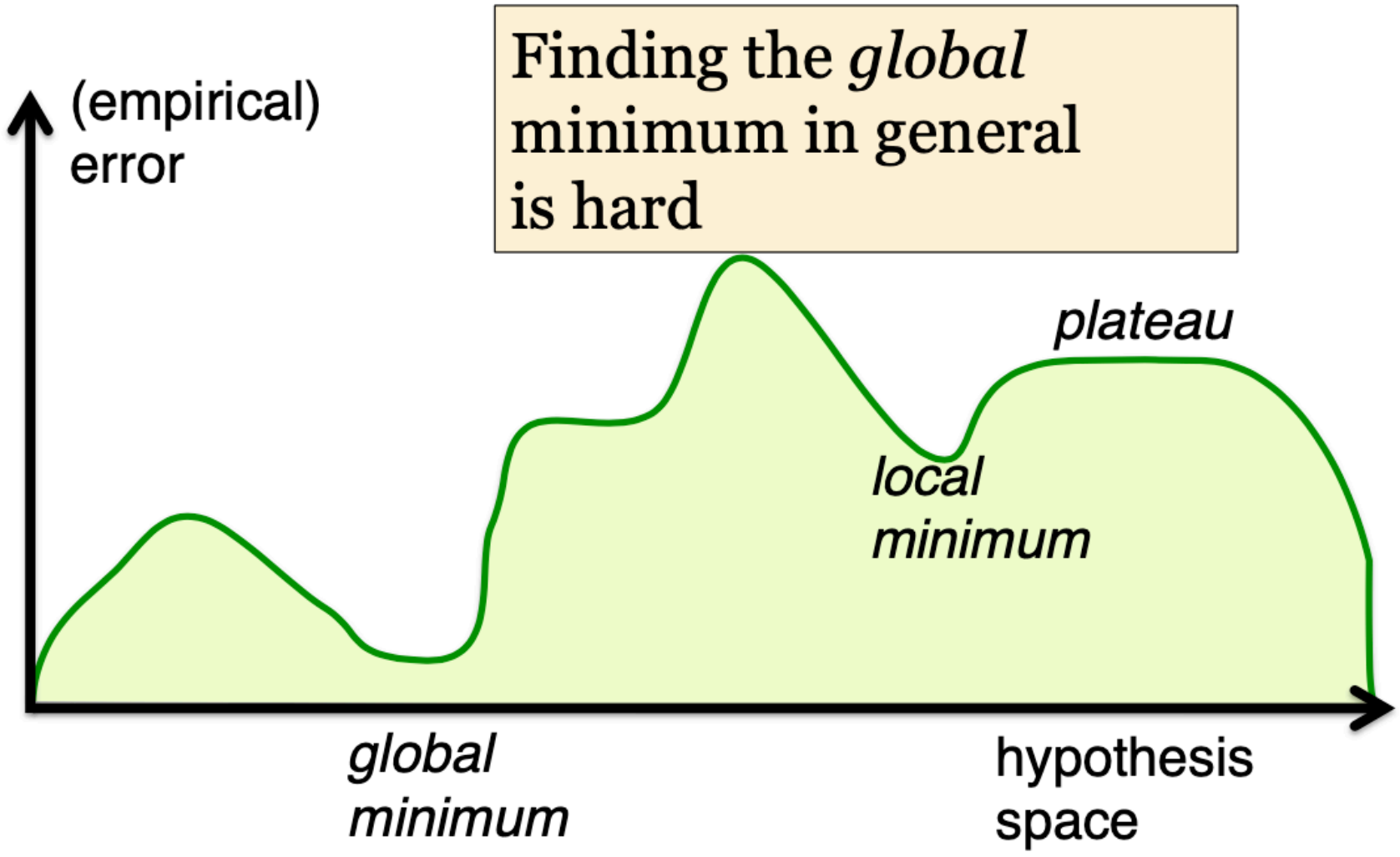
- E.g., find a set of values for $\mathbf{w}$ for which the loss from the loss function is a minimum

A continuous convex loss function allows a simpler optimization algorithm: can take derivative of function

# The loss surface generally

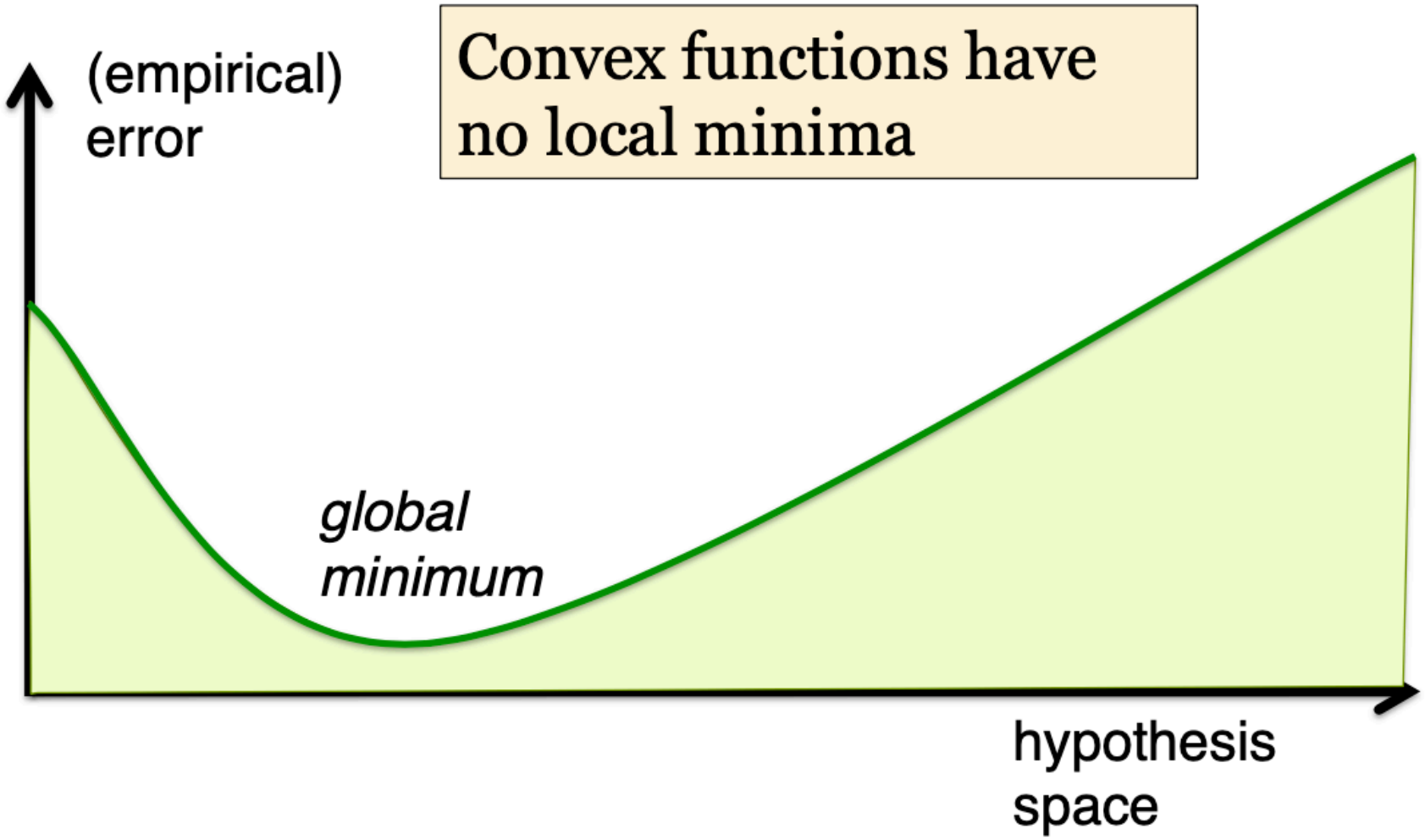


# The loss surface generally




Think of water running downhill: if want minima, could get stuck in local minimum, not global

# The loss surface generally



# How do we navigate loss function?

**Goal**: find weights for which loss is (global) minimum

Remember, the derivative of a function is zero at any local maximum or minimum. So one way to find a minimum is to set $f'(\mathbf{w}) = 0$ and solve for $\mathbf{w}$. But for most functions, there isn't a way to solve this

# How do we navigate loss function?

**Goal**: find weights for which loss is (global) minimum

Remember, the derivative of a function is zero at any local maximum or minimum. So one way to find a minimum is to set $f'(\mathbf{w}) = 0$ and solve for $\mathbf{w}$. But for most functions, there isn't a way to solve this

Instead use gradient descent: algorithmically search different values of $\mathbf{w}$ until you find one that results in a gradient near 0. More on this next class

## Evaluation

# BIAS AND VARIANCE

# Bias

Every learning algorithm requires assumptions about the hypothesis space.

Eg: “My hypothesis space is

- ...linear”
- …decision trees with 5 nodes”
- …a three layer neural network with rectifier hidden units”

# Bias

Every learning algorithm requires assumptions about the hypothesis space.

Eg: “My hypothesis space is

- ...linear”
- …decision trees with 5 nodes”
- …a three layer neural network with rectifier hidden units”

It's possible given the set of assumptions made, no function in that space is good enough. What is error of the best classifier within your hypothesis set? You cannot do better than that

# Bias

Every learning algorithm requires assumptions about the hypothesis space.

Eg: "My hypothesis space is

- ...linear"
- …decision trees with 5 nodes"
- …a three layer neural network with rectifier hidden units"

It's possible given the set of assumptions made, no function in that space is good enough. What is error of the best classifier within your hypothesis set? You cannot do better than that

Bias is the true error (loss) of the *best* predictor in the hypothesis set

Suppose hypothesis space cannot represent a function. What will the bias be?

# Bias

Every learning algorithm requires assumptions about the hypothesis space.

Eg: “My hypothesis space is

- …linear”
- …decision trees with 5 nodes”
- …a three layer neural network with rectifier hidden units”

It's possible given the set of assumptions made, no function in that space is good enough. What is error of the best classifier within your hypothesis set? You cannot do better than that

Bias is the true error (loss) of the *best* predictor in the hypothesis set

Suppose hypothesis space cannot represent a function. What will the bias be?

- Bias will be nonzero, possibly high

# Bias

Every learning algorithm requires assumptions about the hypothesis space.

Eg: "My hypothesis space is

- ...linear"
- …decision trees with 5 nodes"
- …a three layer neural network with rectifier hidden units"

It's possible given the set of assumptions made, no function in that space is good enough. What is error of the best classifier within your hypothesis set? You cannot do better than that

Bias is the true error (loss) of the *best* predictor in the hypothesis set

Suppose hypothesis space cannot represent a function. What will the bias be?

- Bias will be nonzero, possibly high

Underfitting: when bias is high

# Variance

What if the performance of a classifier is dependent on specific training set we have?

- Perhaps the model will change if we slightly change the training set

# Variance

What if the performance of a classifier is dependent on specific training set we have?

- Perhaps the model will change if we slightly change the training set

Variance describes how much the best classifier depends on the training set

- Increases when classifiers become more complex: with sufficient complexity classifier can just remember training set
- Decreases with larger training sets:  probability of overfitting a massive training set is low

# Variance

What if the performance of a classifier is dependent on specific training set we have?

- Perhaps the model will change if we slightly change the training set

Variance describes how much the best classifier depends on the training set

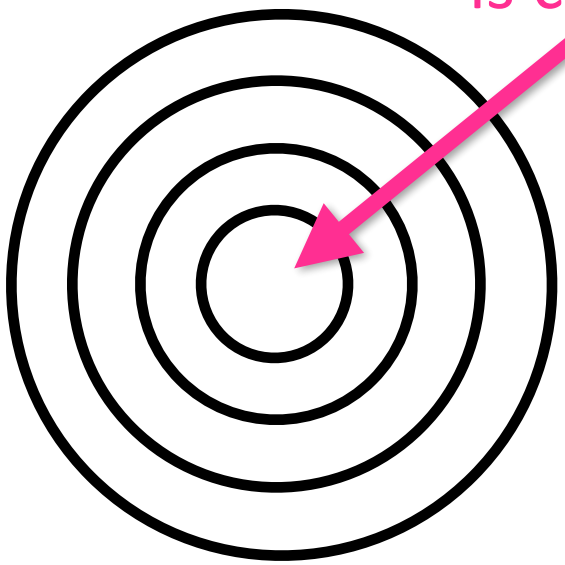
- Increases when classifiers become more complex: with sufficient complexity classifier can just remember training set
- Decreases with larger training sets:  probability of overfitting a massive training set is low

Overfitting: high variance

# Why care about bias vs. variance?

Can be helpful to understand bias vs. variance trade-offs to reason about why classifier may not be working well

# Let's play darts




Throw darts at dartboard

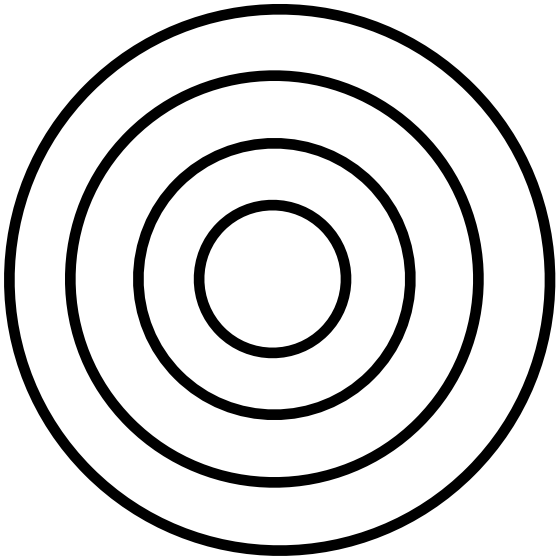
Every throw is a different classifier that is trained on a different training set

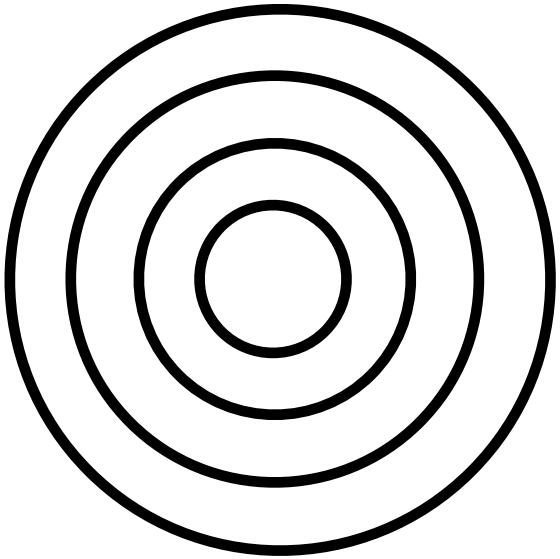
Goal is to hit center (target function).

# Let’s play darts

High bias

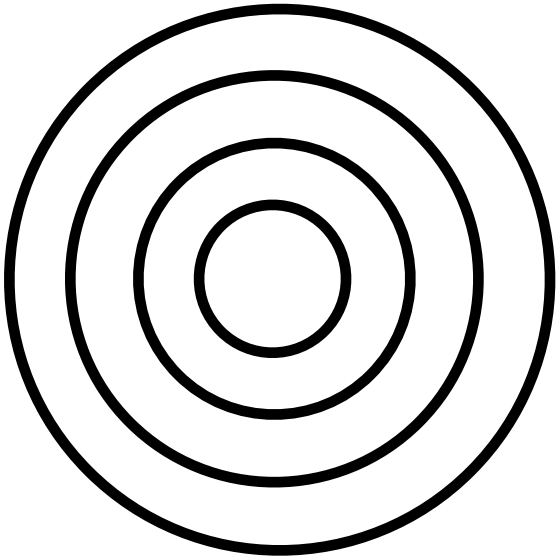
Hypothesis cannot represent target function

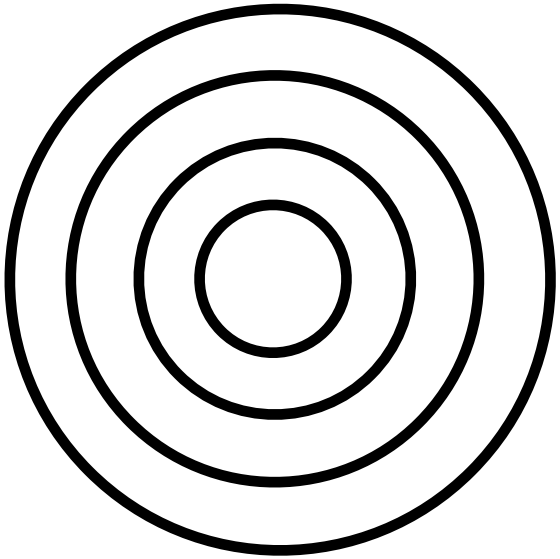




Low bias

Hypothesis can represent target function





Low variance

High variance

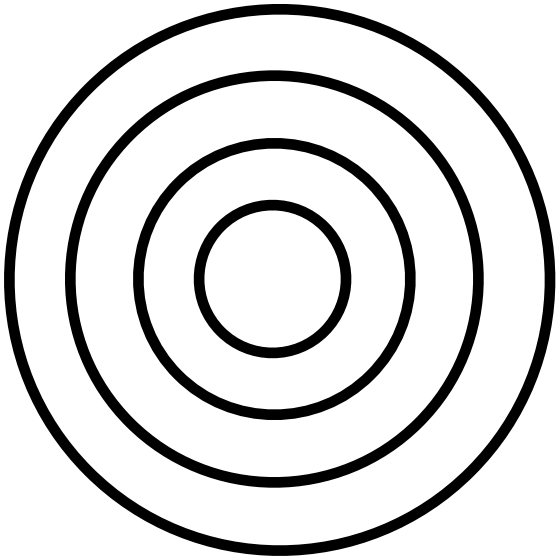
How much does classifier depend on training data?

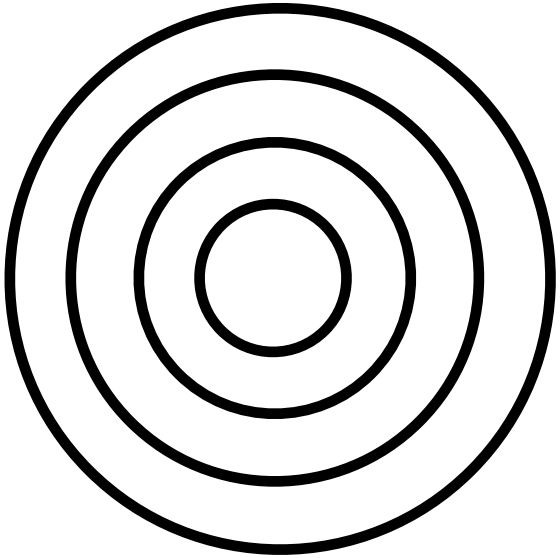
# Let's play darts

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models

## High bias

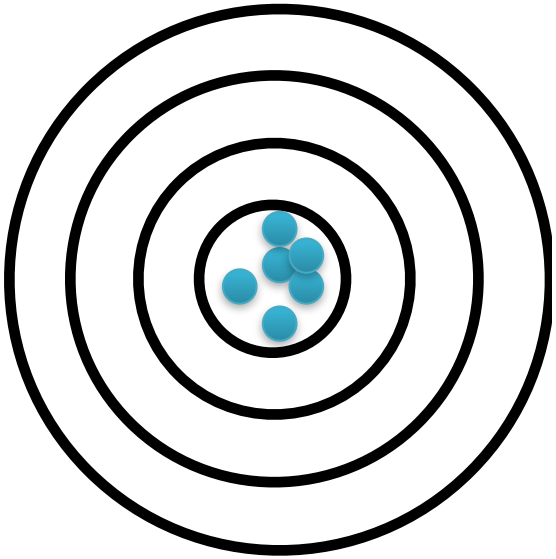
Hypothesis cannot represent target function





## Low bias

Hypothesis can represent target function



Classifier can actually find the true function, changing the data set does not impact this

Each dot is a model that is learned from a a different dataset

## Low variance

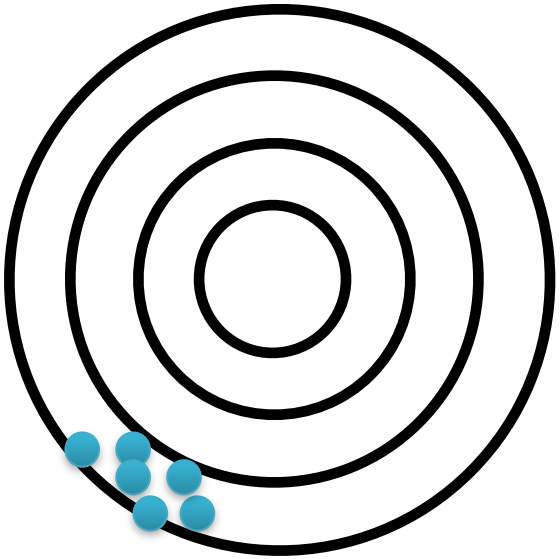
How much does classifier depend on training data?

# Let's play darts

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models

## High bias

Hypothesis cannot represent target function

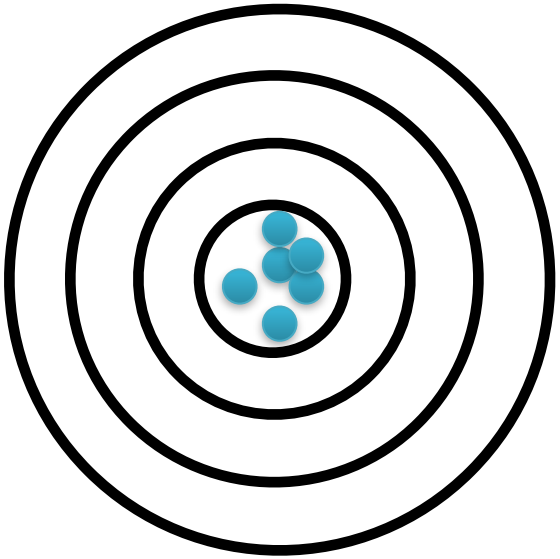


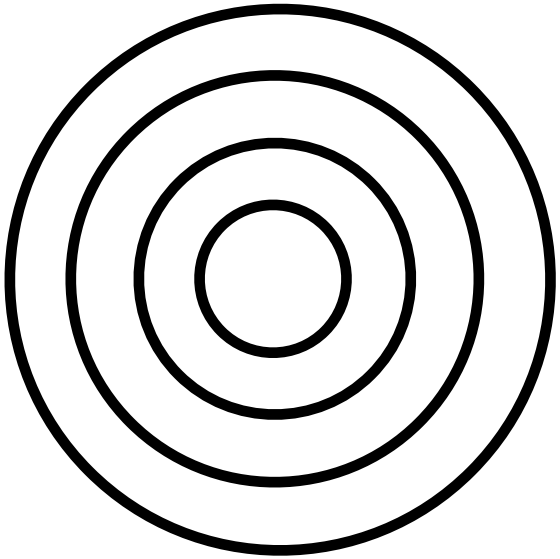
Your hypothesis space is just bad when bias is high. Different data sets can't help

## Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset





Low variance

High variance

How much does classifier depend on training data?

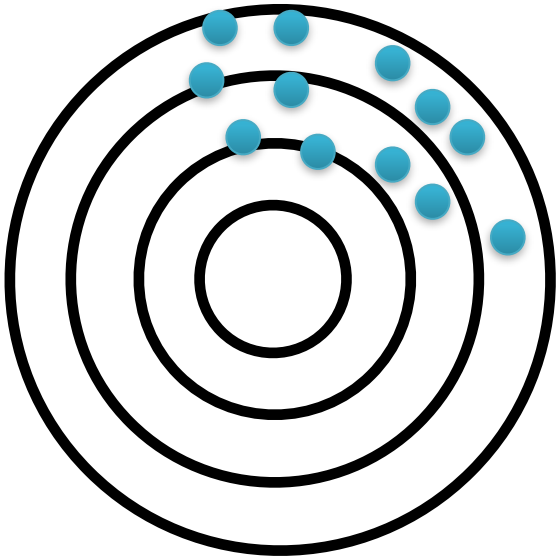
# Let's play darts

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models

## High bias

Hypothesis cannot represent target function

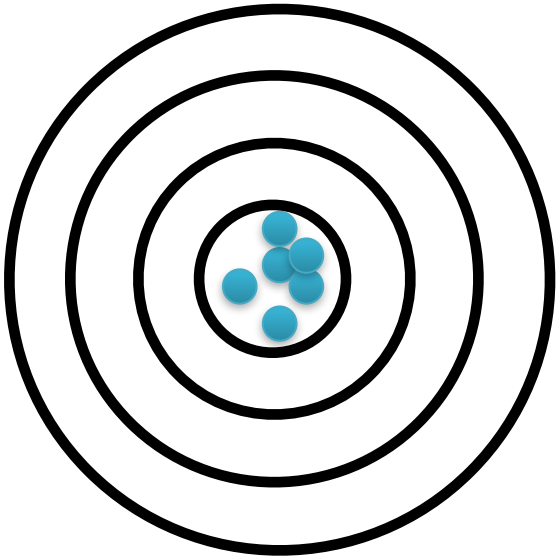
Model is very dependent on data as variance increases, so dots spread out



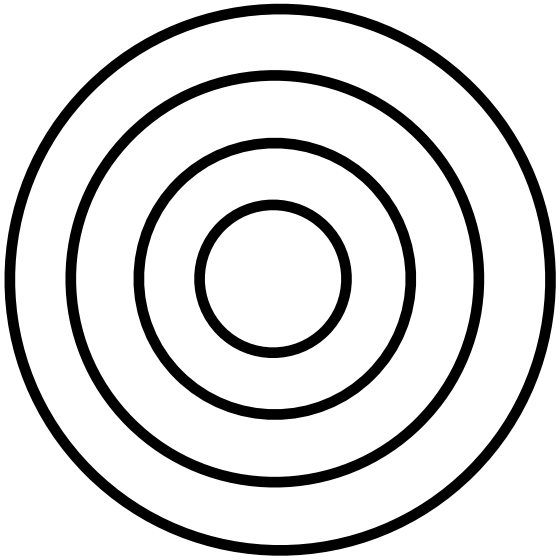
## Low bias

Hypothesis can represent target function

Each dot is a model that is learned from a a different dataset



Low variance



High variance

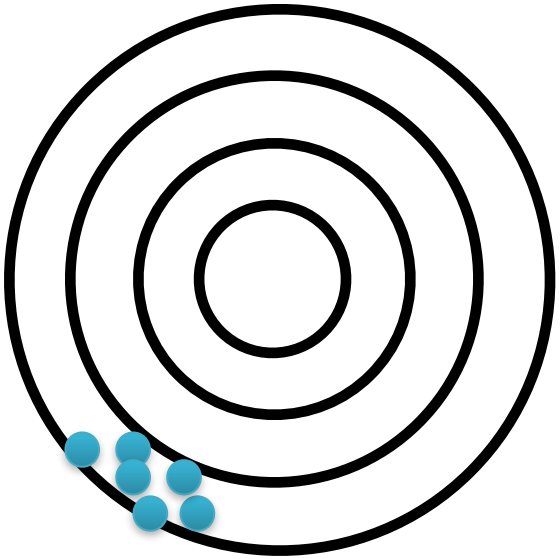
How much does classifier depend on training data?

# Let's play darts

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models

## High bias

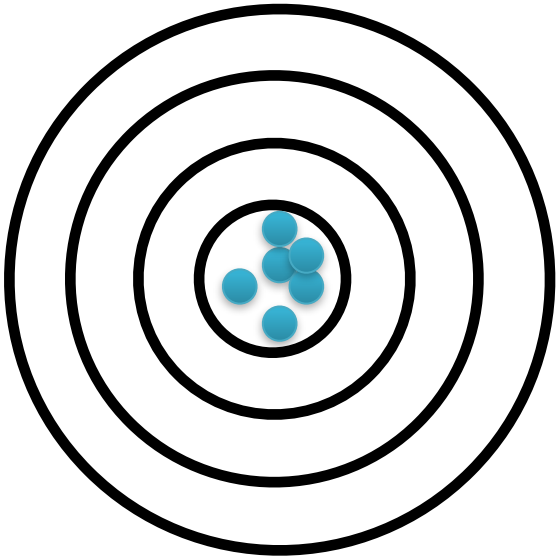
Hypothesis cannot represent target function

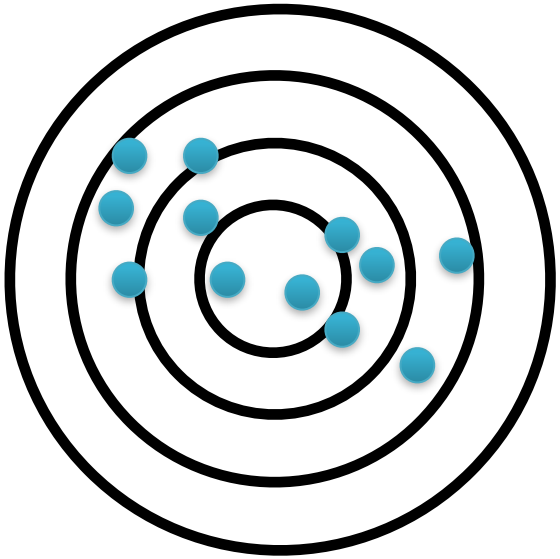


Maybe by chance you hit center, but where dart lands depends heavily on data set

## Low bias

Hypothesis can represent target function





Each dot is a model that is learned from a a different dataset

Low variance

High variance

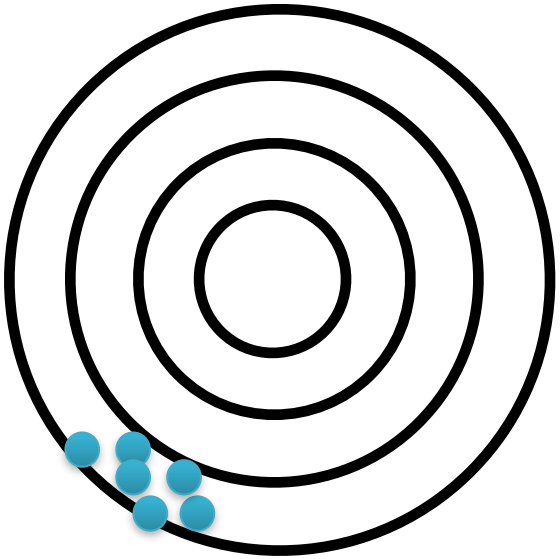
How much does classifier depend on training data?

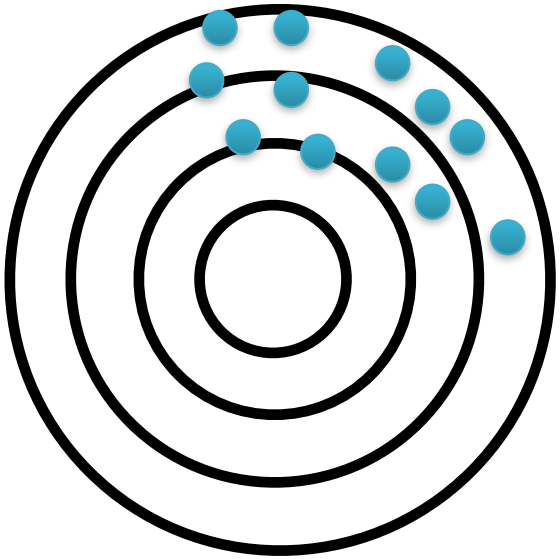
# Let's play darts

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models

## High bias

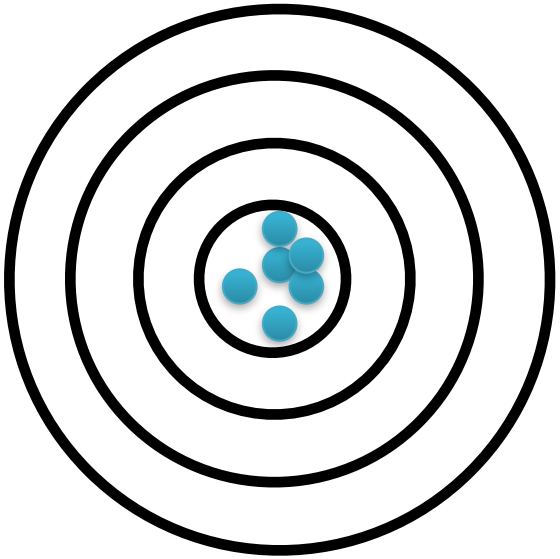
Hypothesis cannot represent target function

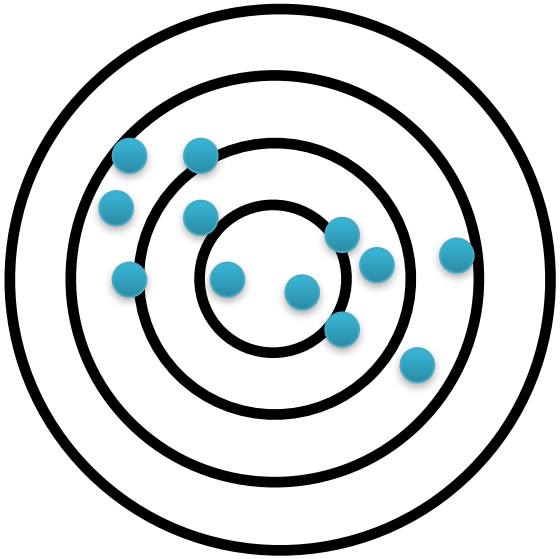




## Low bias

Hypothesis can represent target function





Each dot is a model that is learned from a a different dataset

Low variance

High variance

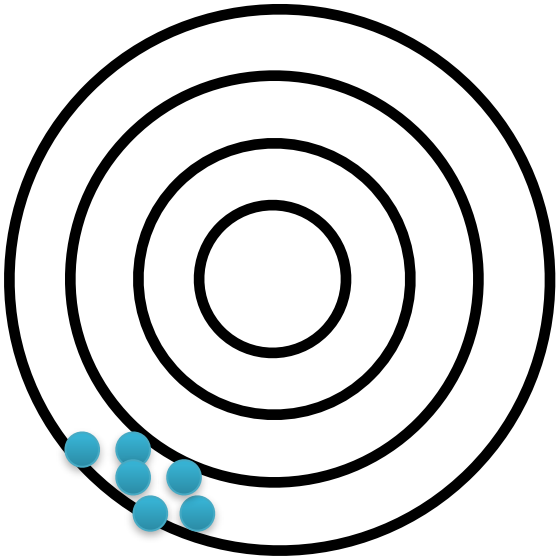
How much does classifier depend on training data?

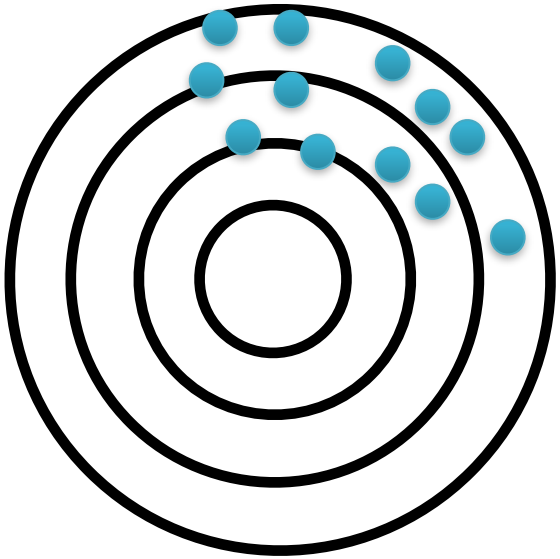
# Let's play darts

Dartboard = hypothesis space
Bullseye = target function
Darts = learned models

## High bias

Hypothesis cannot represent target function

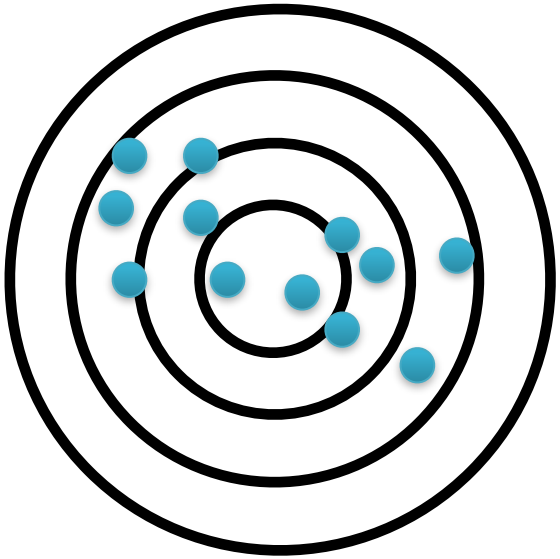




Note: underfitting is not necessarily the opposite of overfitting.

High bias and high variance: both underfitting and overfitting at same time: bad hypothesis space and overfitting to data

But often conceptually a trade-off between bias and variance



## High variance

How much does classifier depend on training data?

# Trade-off between bias vs. variance

Error = bias + variance (+ noise)

High bias

- Both training and test error can be high
- Arises when the classifier cannot represent (underfits) the data

High variance

- Training error can be low, but the test error will be high
- Arises when the learner overfits the training set (essentially remembering the training set

# Impact of amount of data

Small datasets

- Variance is a concern
- Even small changes in training set may change optimal parameters significantly
- Low variance is more important than bias considerations

Large datasets

- Variance is not really a concern
- Having lots of data means sufficient points to represent data distribution accurately
- Low bias is more important than variance consideration

# Managing of bias and variance

Ensemble methods reduce variance

- Multiple classifiers are combined, and averages tend to be more robust
- E.g., bagging, boosting

Decision trees of a given depth

- Reducing depth **increases bias** because makes function space smaller
- Increasing depth **decreases bias** (because can represent more things), increases variance (because can represent or overfit more things)

Neural networks (aka multi-layer perceptron)

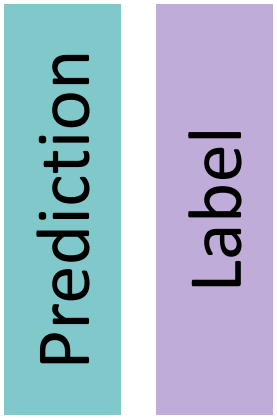
- Deeper models (more layers) decrease bias but can increase variance

# Evaluation

## OTHER METRICS

# Evaluation metrics

To evaluate model, compare predicted labels to actual

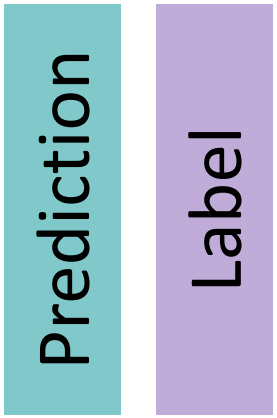



Accuracy: proportion of examples where we predicted correct label

$$\text{accuracy} = \frac{\text{\# of correct predictions}}{\text{\# of examples}}$$

# Evaluation metrics

To evaluate model, compare predicted labels to actual




Accuracy: proportion of examples where we predicted correct label

$$\text{accuracy} = \frac{\text{\# of correct predictions}}{\text{\# of examples}}$$

Error: proportion of examples where we predicted incorrect label

$$\text{error} = 1 - \text{accuracy}$$

$$\text{accuracy} = \frac{\text{\# of incorrect predictions}}{\text{\# of examples}}$$

# Problem

Accuracy or error may not always be the right way to measure performance of classifier …

Suppose # of examples for one class is very different than # of examples for another class

E.g.,

- 99 examples are positive
- 1 example is negative

# Problem

Accuracy or error may not always be the right way to measure performance of classifier …

Suppose # of examples for one class is very different than # of examples for another class

E.g.,

- 99 examples are positive
- 1 example is negative

Just predicting positive for every example gives 99% accuracy!

# Another problem

Accuracy or error may not always be the right way to measure performance of classifier …

Suppose there are differential misclassification costs – say, getting a positive wrong costs more than getting a negative wrong

- Consider a medical domain in which a false positive results in an extraneous test but a false negative results in a failure to treat a disease

# The beginnings of a solution

Let's separately ask what is the performance of classifier on positive labels and what is performance on negative labels.

Or ask what is the performance on each class of labels separately

# Example

Suppose dataset has two classes of labels, A and B

- 10 examples truly labeled A
- 90 examples truly labeled B
- Classifier predicts A for 8 examples.

# Example

Suppose dataset has two classes of labels, A and B

- 10 examples truly labeled A
- 90 examples truly labeled B
- Classifier predicts A for 8 examples.

What fraction of class A examples did the classifier discover?

- 10 examples are truly labeled A
- 8 of truly labeled A examples found by classifier

Recall is 80%

# Example

Suppose dataset has two classes of labels, A and B

- 10 examples truly labeled A
- 90 examples truly labeled B
- Classifier predicts A for 8 examples.

What fraction of class A examples did the classifier discover?

- 10 examples are truly labeled A
- 8 of truly labeled A examples found by classifier

# Example

Suppose dataset has two classes of labels, A and B

- 10 examples truly labeled A
- 90 examples truly labeled B
- Classifier predicts A for 8 examples.

What fraction of class A examples did the classifier discover?

- 10 examples are truly labeled A
- 8 of truly labeled A examples found by classifier

Recall is 80%

What fraction of classifier's predictions of class A were correct

- 10 examples are truly labeled A
- 80 examples are labeled A by classifier

# Example

Suppose dataset has two classes of labels, A and B

- 10 examples truly labeled A
- 90 examples truly labeled B
- Classifier predicts A for 8 examples.

What fraction of class A examples did the classifier discover?

- 10 examples are truly labeled A
- 8 of truly labeled A examples found by classifier

Recall is 80%

What fraction of classifier's predictions of class A were correct

- 10 examples are truly labeled A
- 80 examples are labeled A by classifier

Precision is 10%

# Precision-Recall analysis

What fraction of class "label" examples did the classifier discover?

$$\text{Recall( label )} = \frac{\text{Correct predictions( label )}}{\text{Correct predictions( label )} + \text{ Missed examples( label )}}$$

What fraction of classifier's predictions of class "label" were correct

$$\text{Precision( label )} = \frac{\text{Correct predictions( label )}}{\text{Correct predictions( label )} + \text{ Incorrect predictions( label )}}$$

# Precision-Recall analysis

What fraction of class "label" examples did the classifier discover?

$$\text{Recall( label )} = \frac{\text{Correct predictions( label )}}{\text{Correct predictions( label )} + \text{ Missed examples( label )}}$$

What fraction of classifier's predictions of class "label" were correct

$$\text{Precision( label )} = \frac{\text{Correct predictions( label )}}{\text{Correct predictions( label )} + \text{ Incorrect predictions( label )}}$$

By default, precision and recall computed for the positive label, as that is usually the case of interest and the one usually with fewer example (e.g., diagnosing diseases in patients, identifying spam emails)

# Combining into one number

Sometimes easier to work with a single number as performance measure

F1 score balances precision and recall: harmonic mean of precision and recall

$$f_1 = \frac{2pr}{p + r}$$

Training to minimize F1 is difficult, but can choose hyper parameters for which F1 is maximized

# Practical advice

What measure to use depends on domain and how balanced your dataset is

All labels matter and dataset is balanced:

- Use accuracy

Not all labels matter or dataset not balanced:

- Use precision, recall, F1

# Problems

What if the number of examples for one class is different than the number of examples for another class?

What if you have more than 2 classes? How do you know whether all classes are being predicted equally well?

# Confusion matrices

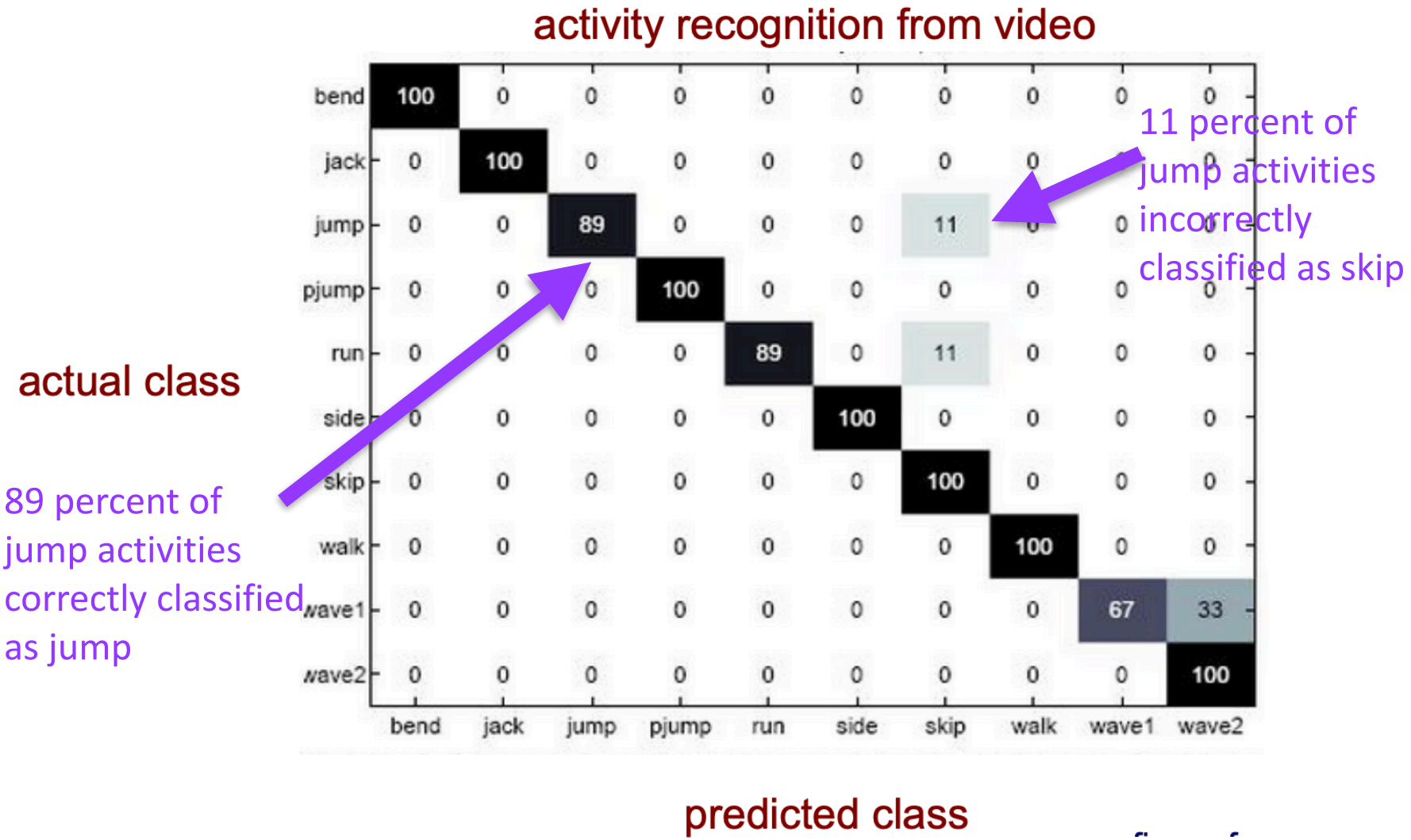
Help us understand what types of mistakes are made by a learned model

Confusion matrix

- Summarizes performance of a classification model
- Shows ways in which your model is confused (types of errors made) when model makes predictions

# Confusion matrices

Table contains counts of correct and incorrect classifications




from http://vision.jhu.edu/

# Midterm

## OVERVIEW

# What you should know

1. General supervised learning
   - Supervised learning, instance spaces, label spaces, hypothesis spaces
   - Classification vs. regression vs. multi-class classification. What they are anyhow they are defined by the label space
   - Understanding why we need to restrict hypothesis space
   - General issues in supervised learning: hypothesis spaces, representation (i.e., features), learning algorithms

2. Decision trees
   - What is a decision tree? What can they represent?
   - How to predict with a decision tree
   - Expressivity, counting the number of decision trees
   - Dealing with continuous features
   - Learning algorithm: The ID3 algorithm, entropy, information gain
   - Overfitting (applicable not just to decision trees) and how to deal with it when training decision tress
   - Variants of the entropy-based information gain measure

# What you should know

3. Linear classifiers
   - What are linear classifiers? Why are they interesting?
   - Linear separability. What can linear classifiers express? What can they not express? Examples of functions that are linearly separable and not linearly separable
   - What is the role of the bias?
   - Feature expansion to predict a broader set of features

4. Perceptron
   - The original algorithm
   - Variants of the Perceptron algorithm
   - Margin of a classifier

5. Least mean squares regression (depending on what we get through on Wed.)
   - What is LMS regression?
   - The idea of learning via minimizing a cost function
   - Gradient and stochastic gradient descent for LMS. The difference between them

# Midterm format

In class on Wednesday, March 30

- Closed books, closed notes
- Covers material in lectures 1 to 14

Will not ask questions on

- Probability problems
- Geometric interpretation of perceptron

5 questions

- Short questions: supervised learning generally, bias, variance, metrics
- Short questions: about decision tree, perceptron, linear classifiers
- Decision tree question
- Perceptron question
- Something requiring a bit more thought

# Problem 1

Why must we make assumptions about hypothesis space?

If label is real-valued, what kind of learning problem do you have?

# Problem 2

Suppose you train a perceptron on a dataset but the training accuracy is always near 50%. Assume you tuned the perceptron as best you could. Provide a suggestion for something that could potentially improve the training accuracy, and explain why this might help.

# Problem 3

Describe the Perceptron algorithm and explain how how it works on a specific problem

# Problem 4

Build a decision tree that expresses a particular boolean function: A OR B, A AND NOT B, …

Explain how you would build a decision tree using continuous valued data

Explain how entropy measures the purity of splits in the ID3 algorithm and why that is useful