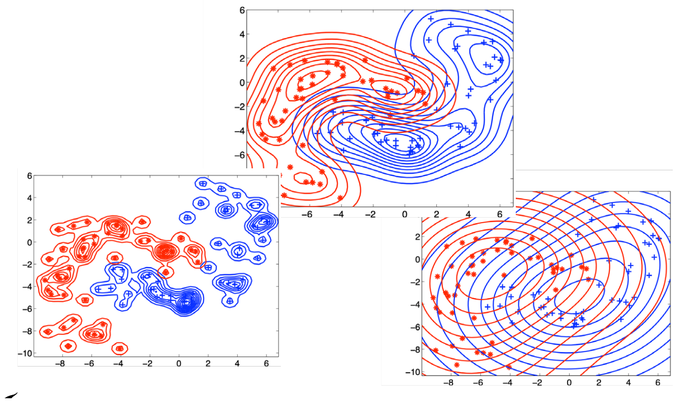
LINEAR DISCRIMINATIVE CLASSIFIERS

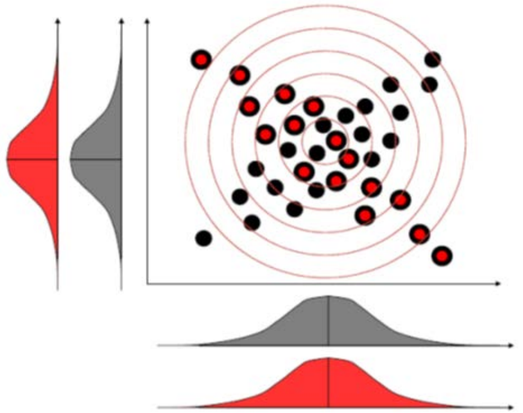
Linear Regression & Gradient Descent

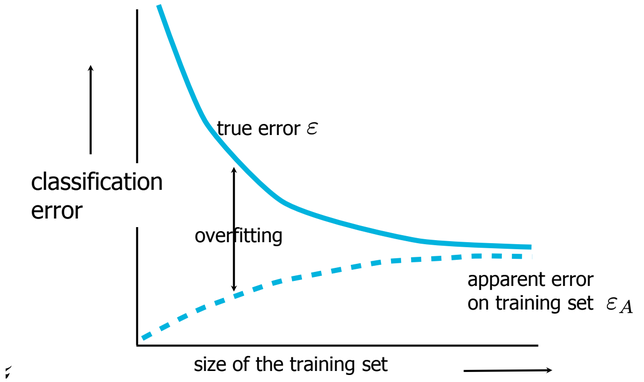
JESSE KRIJTHE - CSE2510 - MACHINE LEARNING

Last week



- Non-parametric density estimation and classifiers: Parzen and k-NN
- (Non)-parametric Naive Bayes classifier
- Classifier Evaluation
- Generalization and Bias/Variance trade-off





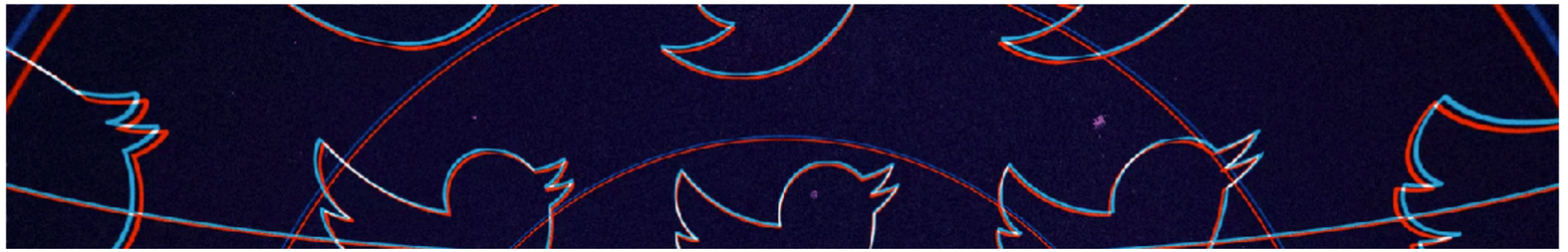
TECH | TWITTER

Twitter is looking into why its photo preview appears to favor white faces over Black faces

Users discovered the problem with the neural network that crops photo previews

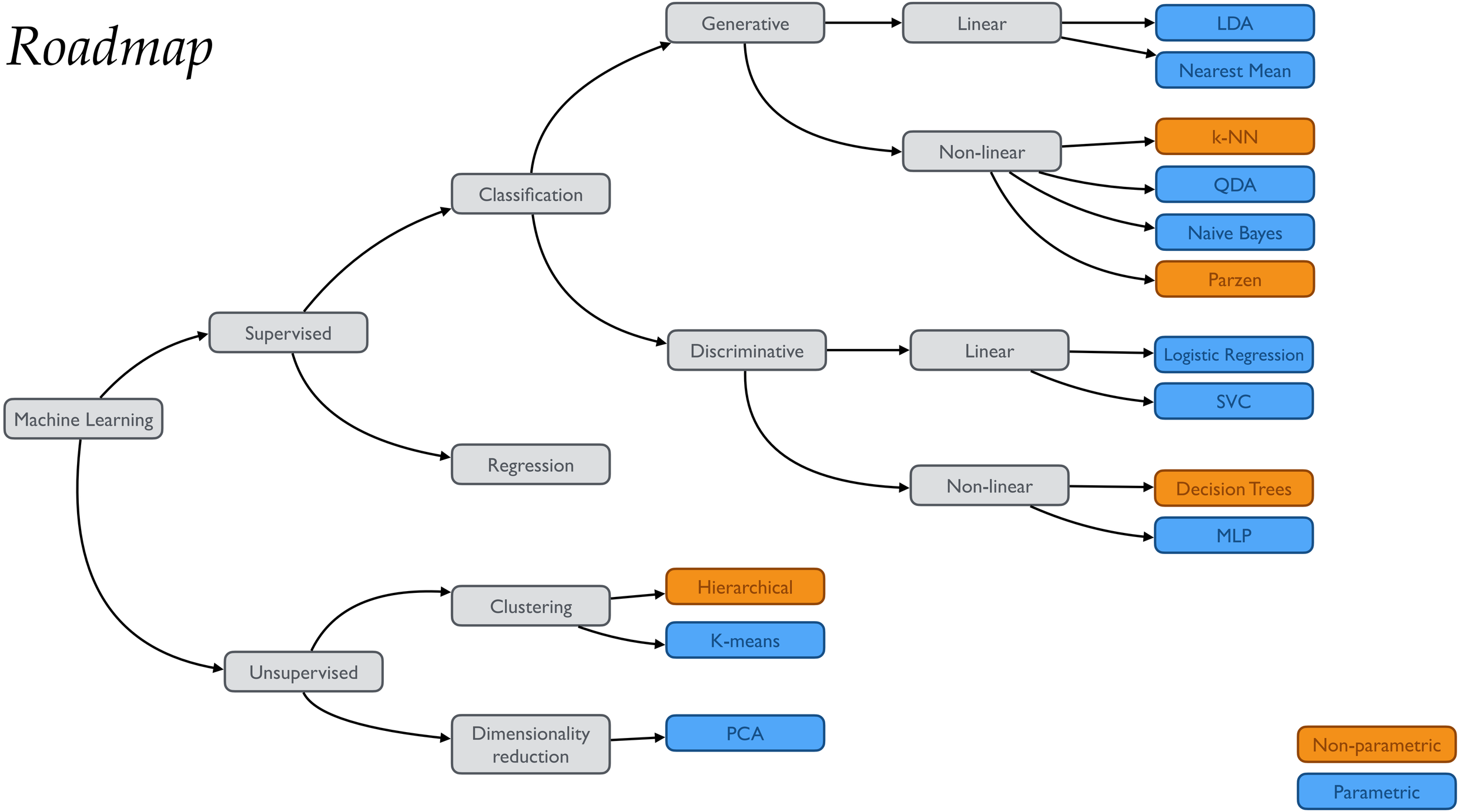
By Kim Lyons | Sep 20, 2020, 4:20pm EDT

SHARE



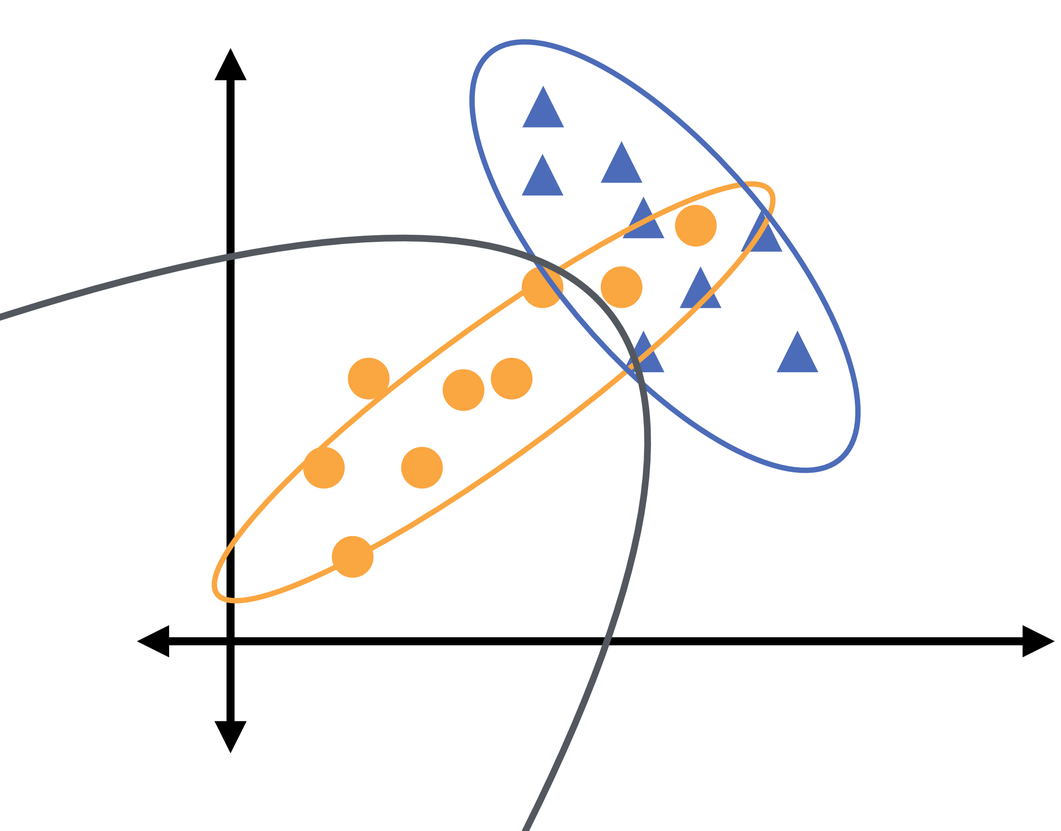
VERGE DEALS

Roadmap

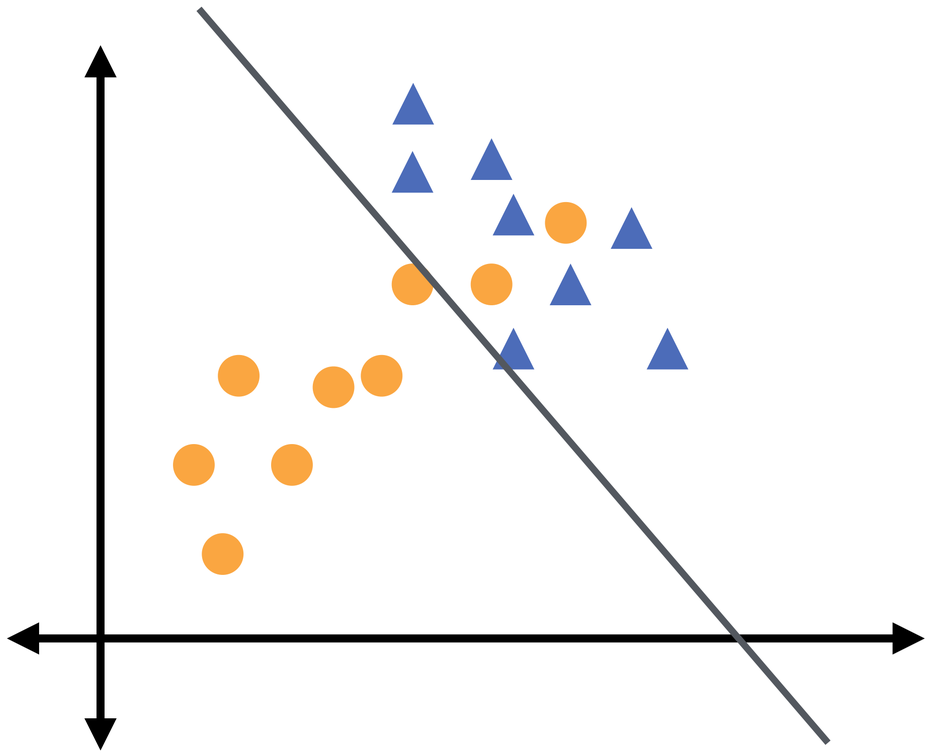

Machine Learning
Supervised
Classification
Generative
Linear
LDA
Nearest Mean
Non-linear
k-NN
QDA
Naive Bayes
Parzen
Discriminative
Linear
Logistic Regression
SVC
Non-linear
Decision Trees
MLP
Regression
Unsupervised
Clustering
Hierarchical
K-means
Dimensionality reduction
PCA
Non-parametric
Parametric

Generative vs. Discriminative

Model $P(X \mid Y)$ and $P(Y)$, or $P(X,Y)$ to derive $P(Y \mid X)$



Model $P(Y \mid X)$ or $h(X)$ directly



Learning Goals & Reading

LEARNING GOALS

After this week you will be able to

- **Distinguish between generative and discriminative models**
- **Reason about linear regression models and linear classifiers**
- **Explain what hypothesis and cost functions are**
- Implement **gradient descent** to train a given linear model
- Derive and implement logistic regression from its loss function
- Identify the principles behind support vector classifiers
- Describe some approaches to multi-class classification and their problems

LITERATURE

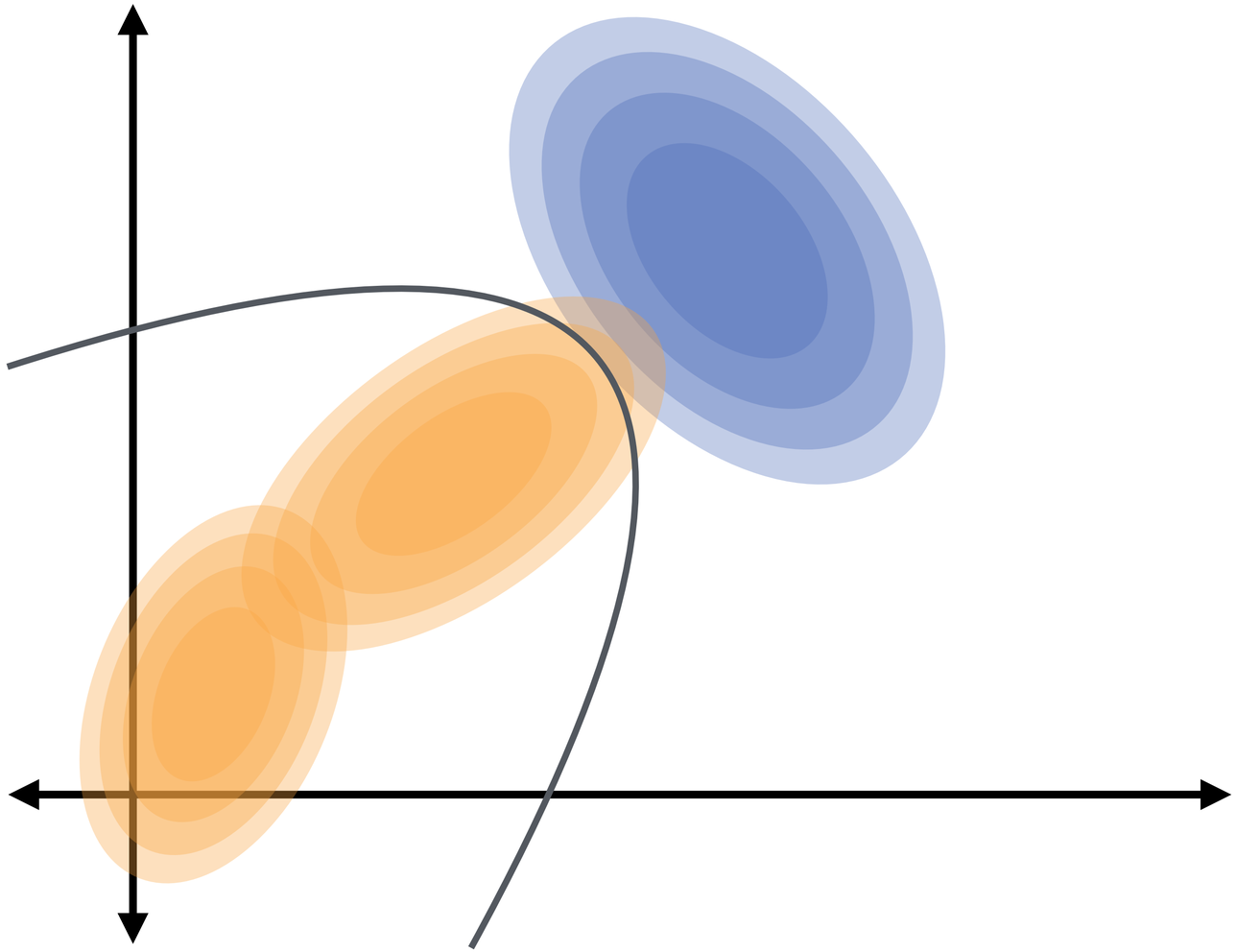
Bishop's "Pattern Recognition and Machine Learning"

- **Chapter 3 up to and including 3.1.3**
- **5.2.4**
- 4.3 up to and including 4.3.2
- 4.3.4
- 7.7 up to and including equation (7.7)
- 7.7.1 up to and including equation (7.22)
- 4.1.2

Today

- How to define a classifier/regressor through a loss function and hypothesis class
- What is “linear” about a linear model?
- Finding the classifier/regressor by minimizing the “loss” on the training set
- Constructing a linear regression model and logistic classifier by choosing a loss function and hypothesis class

Risk Minimization



Two classes, with the joint probability depicted on the left. How do we find a good discriminator?

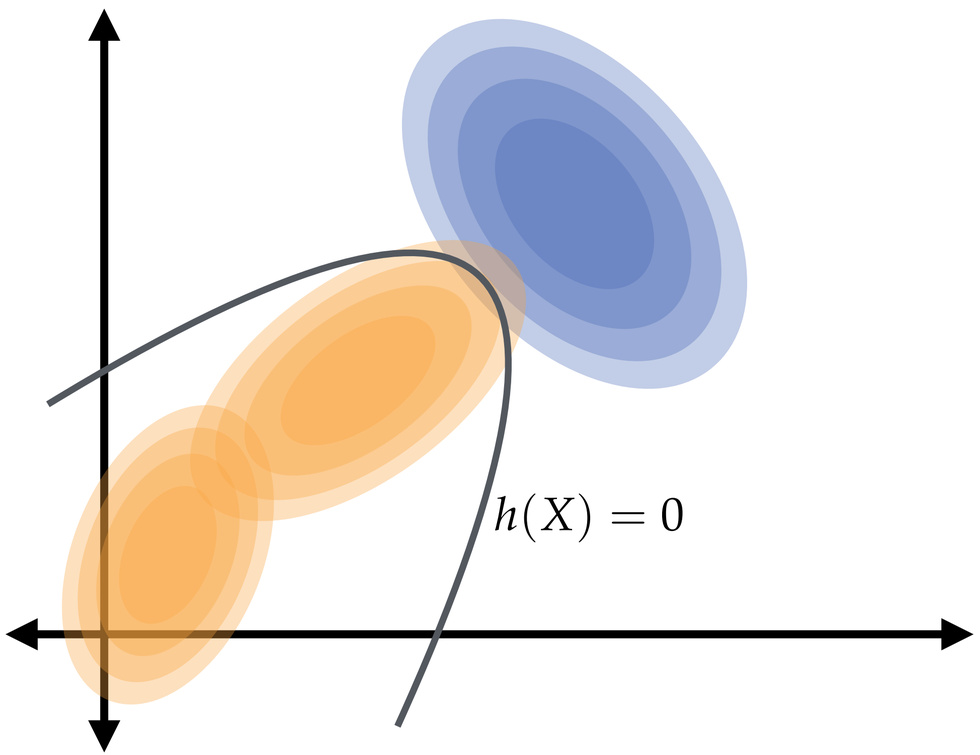
Here: rather than modelling the complete distribution $P(X,Y)$, directly model the decision/discriminant function ($P(Y \mid X)$ or $h(X)$) or decision boundary.

Last week's $g(X)$

2 choices:

- What can the classifiers look like?
- How do we measure how well the classifier works?

Minimizing the "Risk"



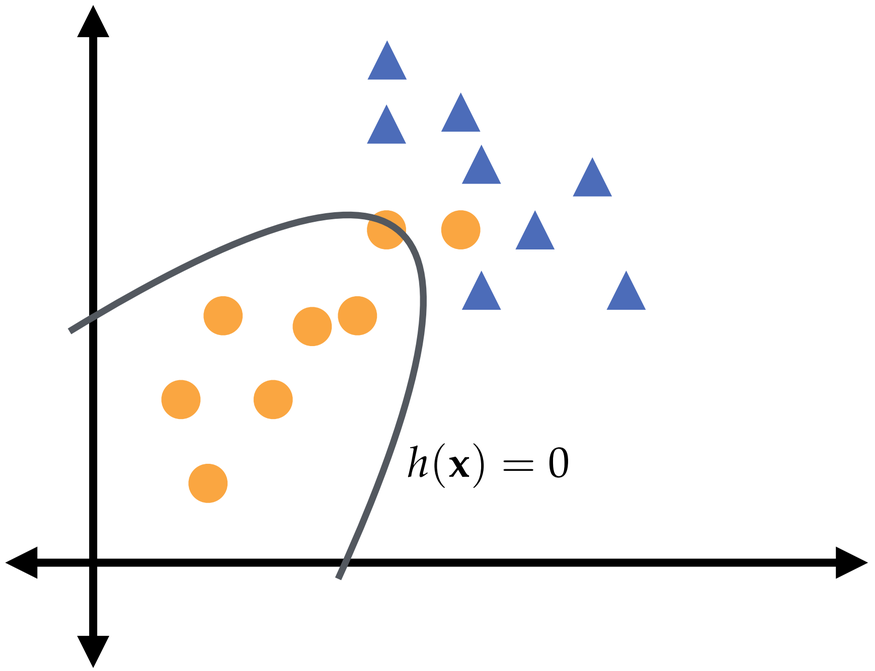
$$L(h(X), Y)$$

X, Y: random variables corresponding to the input features and outcome of interest

h: hypothesis function that maps input to a decision value

L: loss/cost function that measures how well the predicted outcome matches the real outcome

The Empirical Risk



$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

$\mathbf{x}$, y: observed values corresponding to the input features and outcome of interest

h: hypothesis function that maps input to a decision value

L: loss/cost function that measures how well the predicted outcome matches the real outcome

Classifier Construction using Empirical Risk Minimisation

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

Hypotheses

- When the outcome is continuous (regression), $h(\mathbf{x})$ can directly correspond to the function we want to find
- In classification $h(\mathbf{x})$ is the discriminant function which gives a real-valued output
  - To go from this output to a class decision, we still need to set a cut-off, for instance at 0:

$$y = \begin{cases} c_1, & \text{if } h(\mathbf{x}) > 0 \\ c_0, & \text{if } h(\mathbf{x}) \leq 0 \end{cases}$$

Hypothesis Class: Linear

weight vector

$$h(\mathbf{x}) = w_1 x_1 + w_2 x_2 + ... + w_D x_D + w_0 = \mathbf{x}^\top \mathbf{w} + w_0$$

bias/intercept

Decision function is a weighted linear combination of the input features, plus a constant (intercept/bias/threshold)

Trick: for notational convenience, we sometimes add the bias term to the weight vector, by adding a constant feature

$$h(\mathbf{x}) = w_1 x_1 + w_2 x_2 + ... + w_D x_D + w_0 1 = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{w} \\ w_0 \end{bmatrix}$$

Linear Hypotheses

$$h(\mathbf{x}) = w_1 x_1 + w_2 x_2 + ... + w_D x_D + w_0 = \mathbf{x}^\top \mathbf{w} + w_0$$

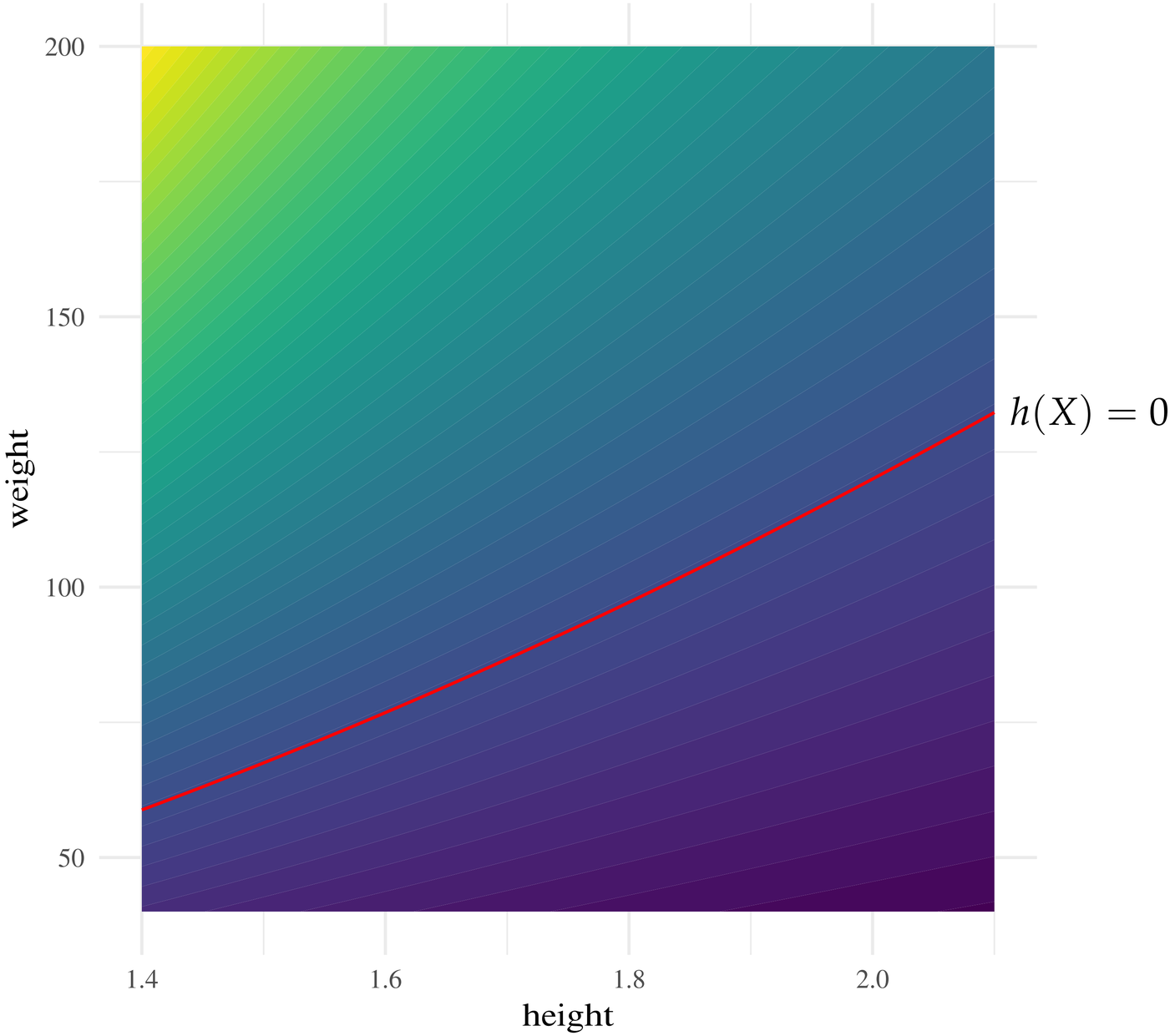
- Why linear?
  - "Simple"
  - Optimization is often possible and fast
  - Interpretable
  - Often/sometimes a reasonable approximation, locally
- We have seen linear classifiers before: LDA, nearest mean, …
- Linear hypothesis class: all different $\mathbf{w}$

Practice Question: BMI

Body Mass Index (BMI) is defined as weight (in kg) divided by height^2 (in m^2). Suppose I use a fixed rule: to classify people as healthy when BMI<30.

1. **True or False**: Given the input features weight and height, I can construct a linear classifier that is exactly the same as this rule.

2. **True of False:** Given the input features weight, height and BMI, I can construct a linear classifier that is exactly the same as this rule.

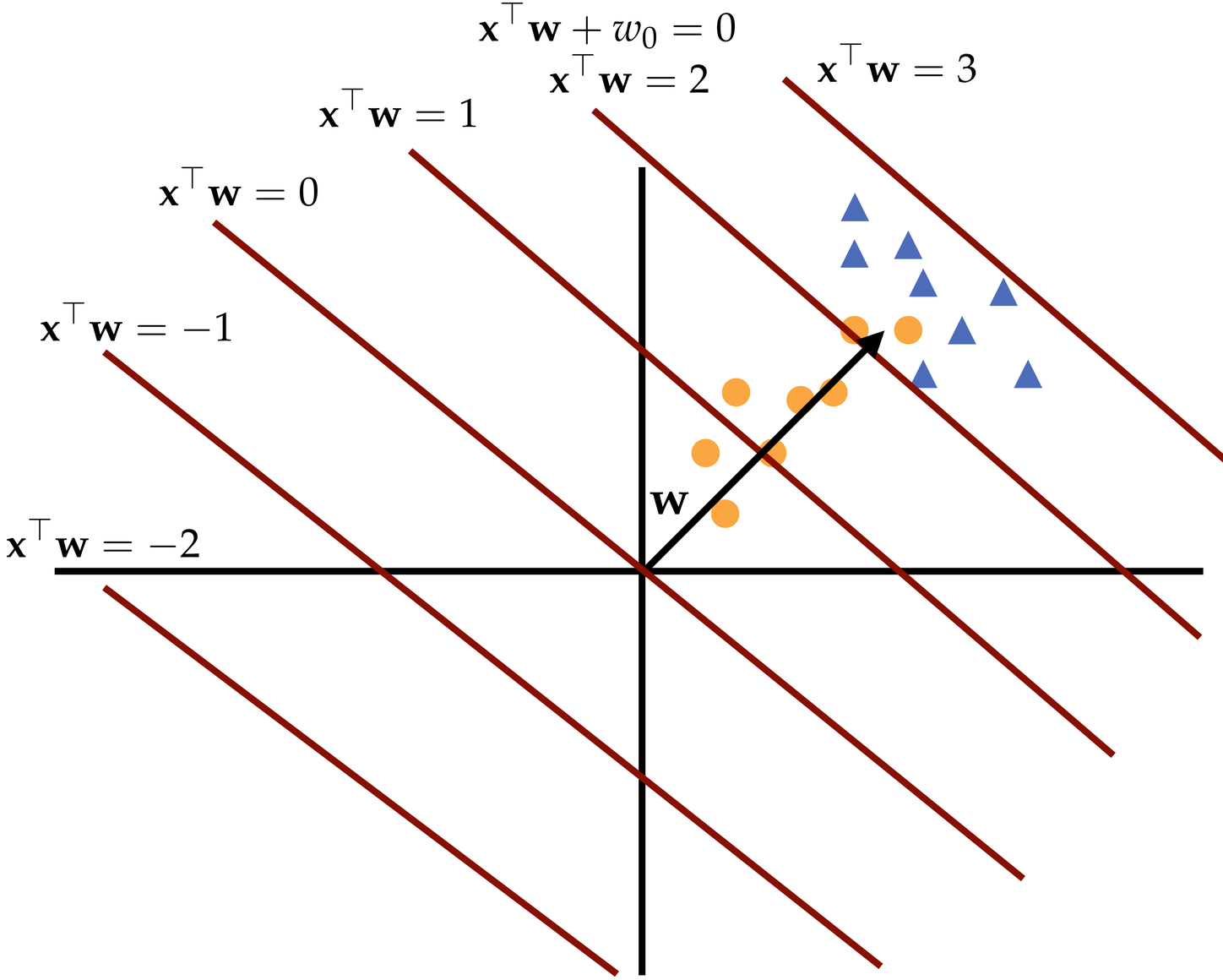

200
150
100
50
weight
1.4
1.6
1.8
2.0
height
$h(X) = 0$

BREAK

Linear Decision Boundaries

Decision boundary is where $\mathbf{x}^\top\mathbf{w}+w_0 = 0$. This means $\mathbf{w}$ is orthogonal to every vector "within the decision boundary". For a 2D problem, this means it has to be a hyperplane of dimension 1 (a line).

What does the linear decision boundary look like for a 3D, 1D or KD problem?



What is "linear" about a classifier?

The decision function (usually…) and the decision boundary

Side note: Multiple $\mathbf{w}$'s can give rise to the same decision boundary

For our choice for a hypothesis class,
let's go with all linear functions:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Classifier Construction using Empirical Risk Minimisation

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions

2. Define a cost (loss) function to measure the "quality" of each hypothesis

Cost/Loss function

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

- How do we measure how ”well” the model solves the problem?
- Different Problems: Regression & Classification
- Let’s start with regression

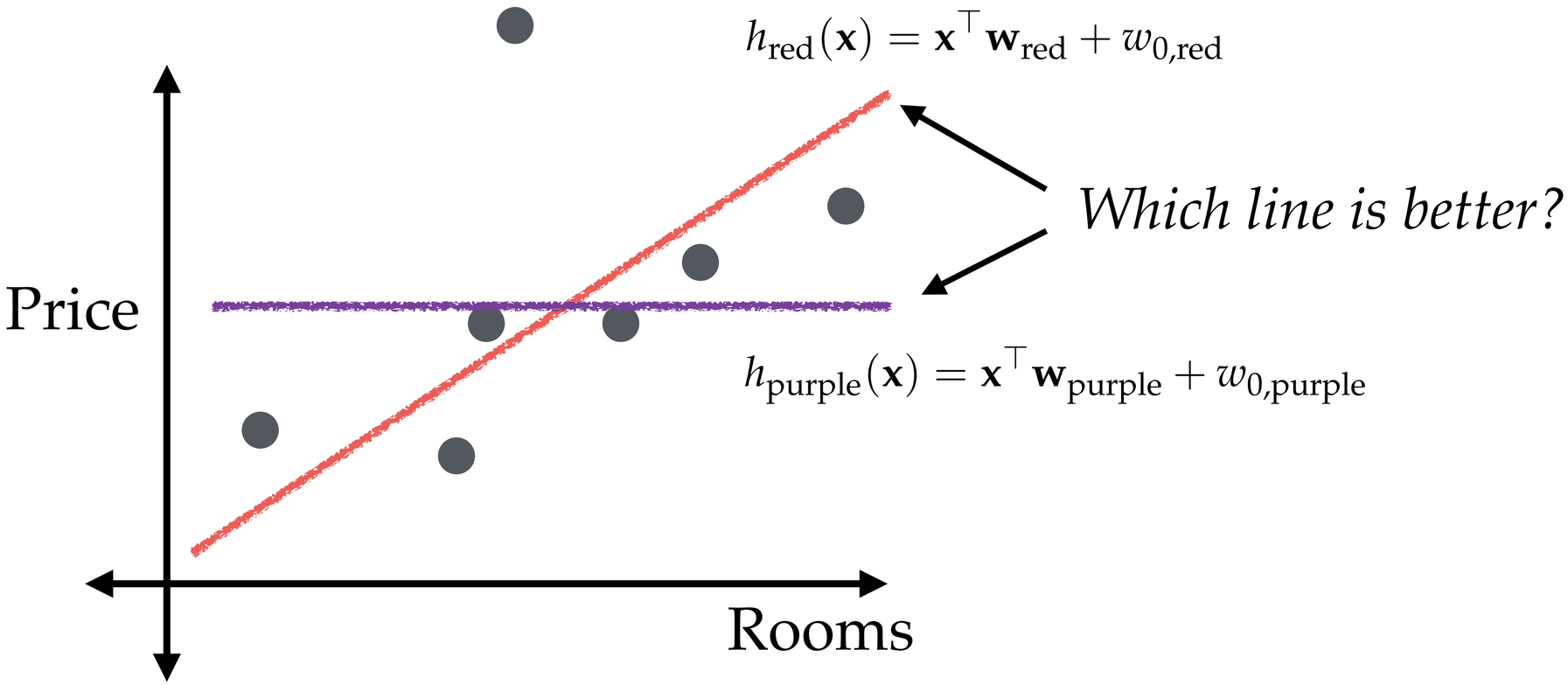
LINEAR REGRESSION

Regression Problem

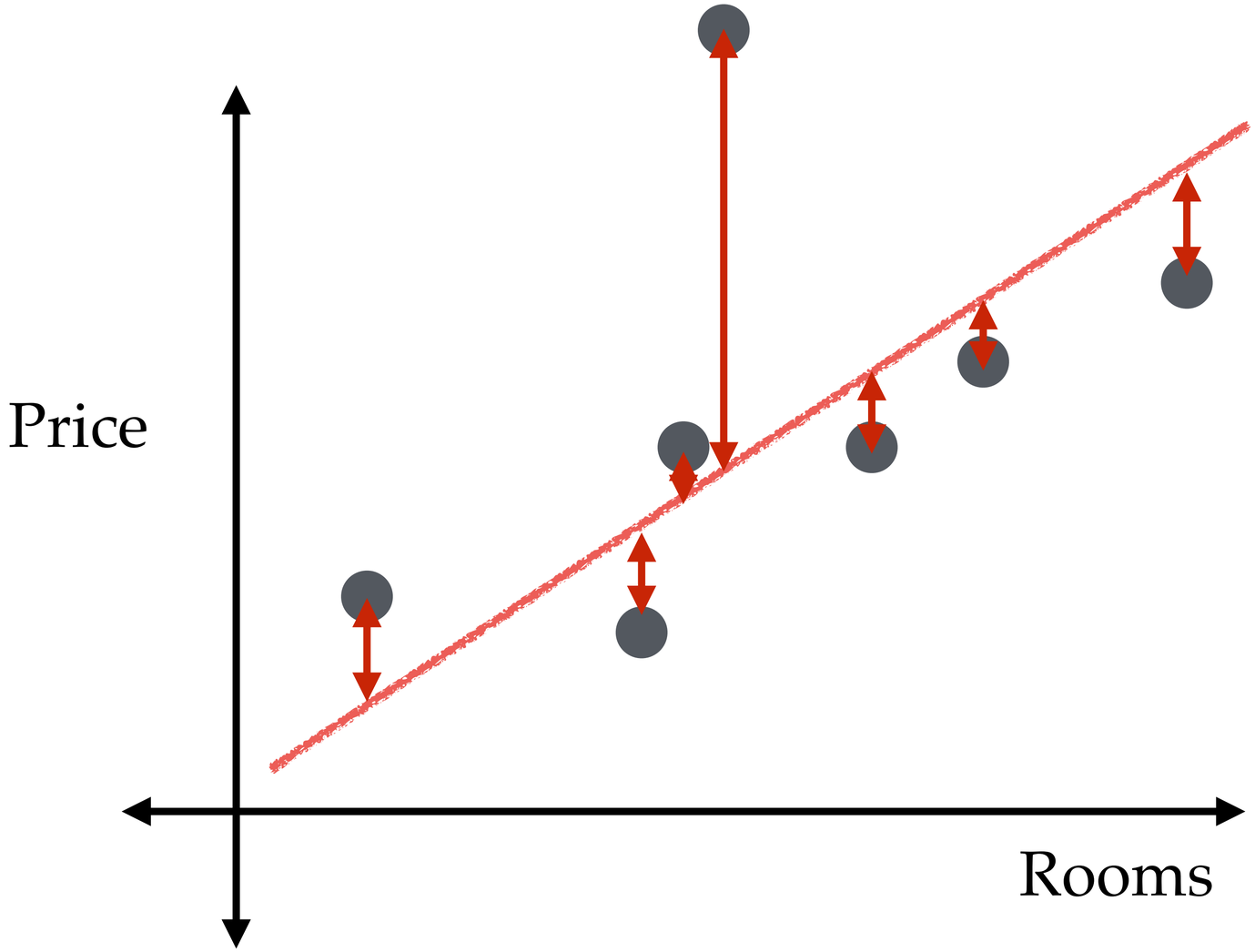
We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Task: predict the price of a house, given the number of rooms (note: continuous outcome, not discrete classes)



Regression Losses



We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Consider the difference between the predicted value and the true value:

$$(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)$$

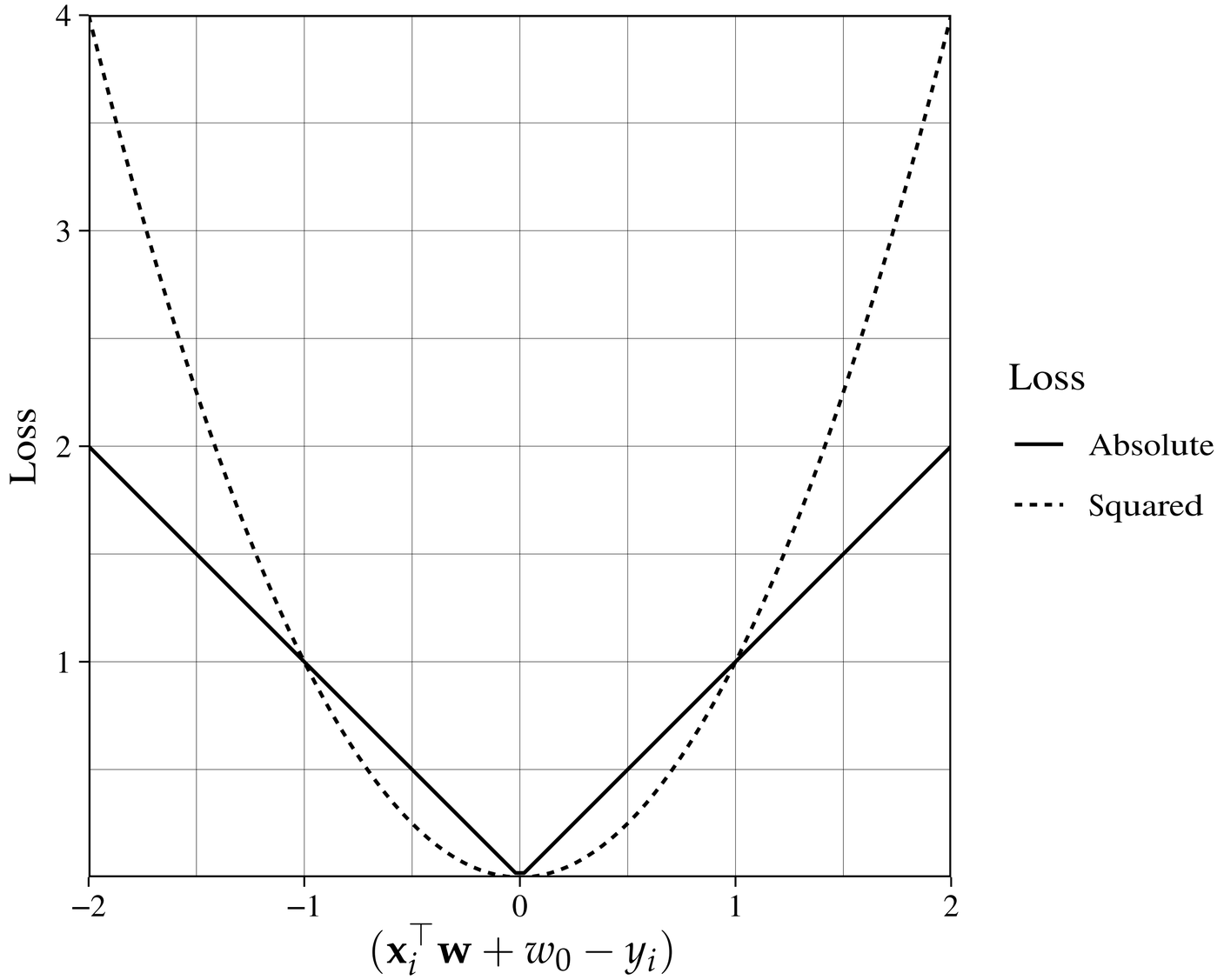
We can consider the absolute difference:

$$|\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i|$$

Or penalise large differences more strongly by taking the squared difference:

$$(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

Regression Losses

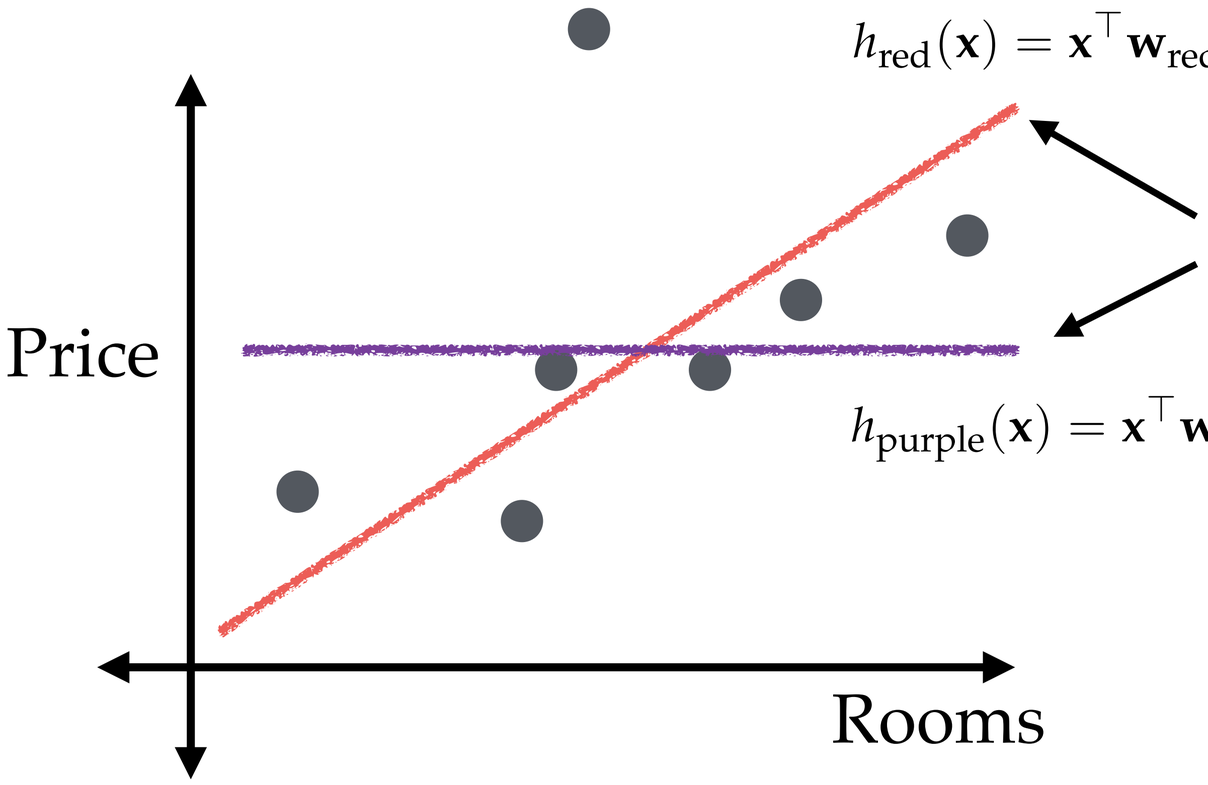


Regression Problem

We need to define a loss:

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{x}_i^\top \mathbf{w} + w_0, y_i)$$

Task: predict the price of a house, given the number of rooms (note: continuous outcome, not discrete classes)



$$h_{\text{red}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}_{\text{red}} + w_{0,\text{red}}$$

$$\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^\top \mathbf{w}_{\text{red}} + w_{0,\text{red}} - y_i)^2$$

Which line is better?

$$h_{\text{purple}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}_{\text{purple}} + w_{0,\text{purple}}$$

$$\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^\top \mathbf{w}_{\text{purple}} + w_{0,\text{purple}} - y_i)^2$$

Classifier Construction using Empirical Risk Minimisation

$$\min_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^{N} L(h(\mathbf{x}_i), y_i)$$

1. Define the class of possible functions
2. Define a cost (loss) function to measure the "quality" of each hypothesis
3. Measure the average cost on the training data
4. Find the function that minimises this cost

Defining the regression solution

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

Cost function: Squared loss

Hypothesis class: all linear models

Minimization procedure: …

Minimizing the Empirical Risk

$$\min_{\mathbf{w}, w_0 \in \mathbb{R}^{D+1}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2 = \min_{\mathbf{w}, w_0} J(\mathbf{w}, w_0)$$

How do we find the best $\mathbf{w}$?

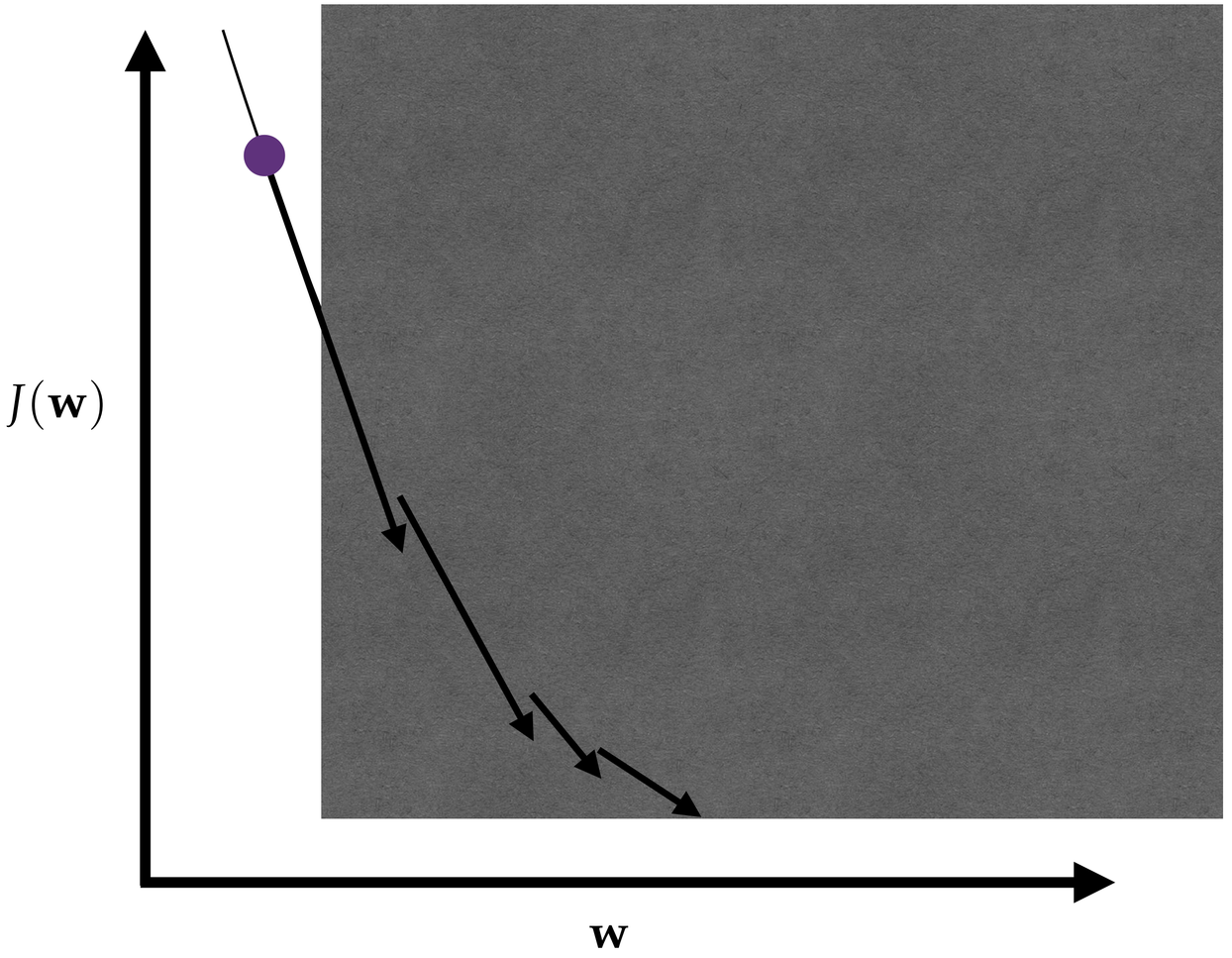
- Taking the derivative, finding $\mathbf{w}$ for which it is 0?
- Trying all possible values?
- Starting at some $\mathbf{w}$ and keep making small changes to improve it?

I’m on a mountain, in the fog. My car is in the lowest point of the valley. What is a good strategy to get back to the car (fast)?

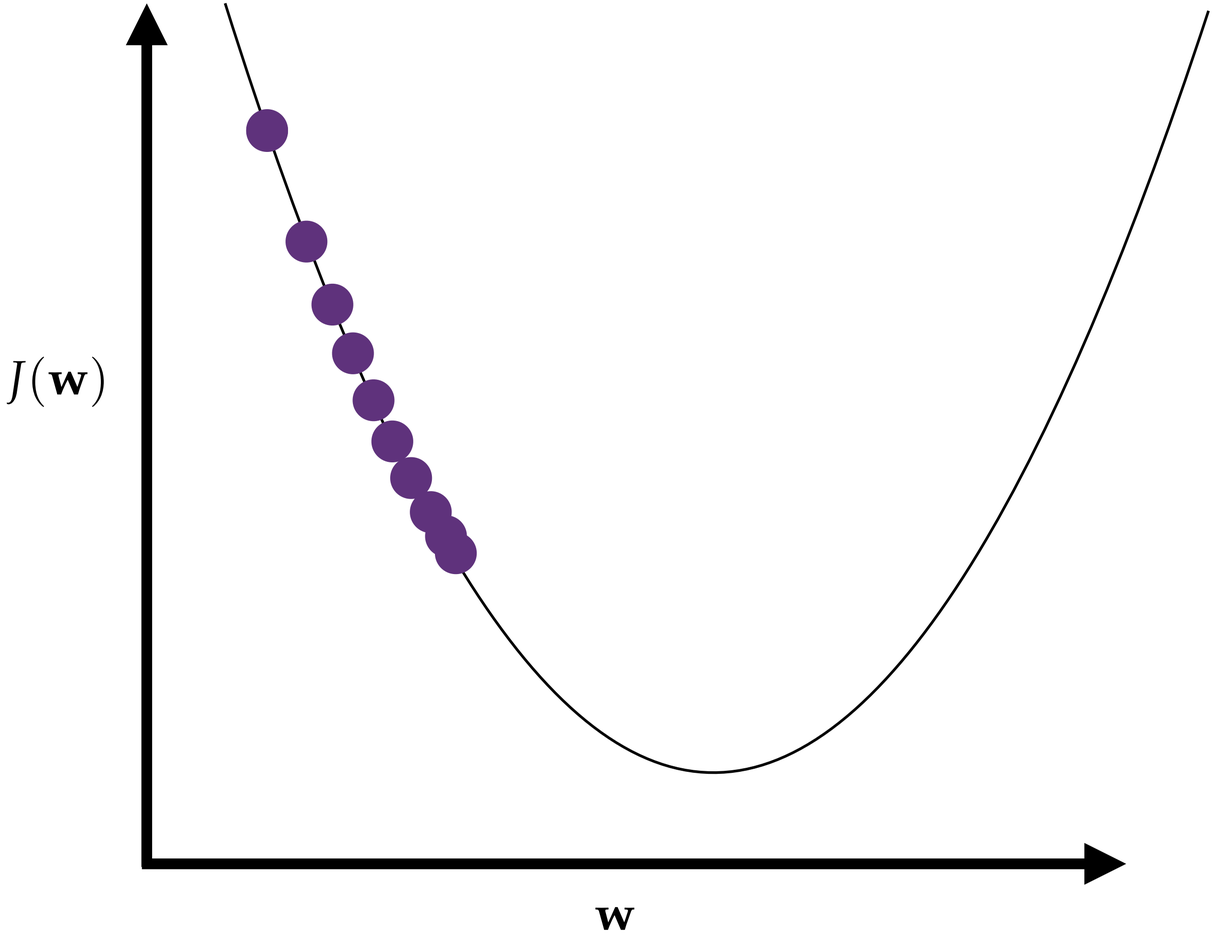
Gradient Descent

$$w_j^{t+1} = w_j^t - \alpha \left. \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \right|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

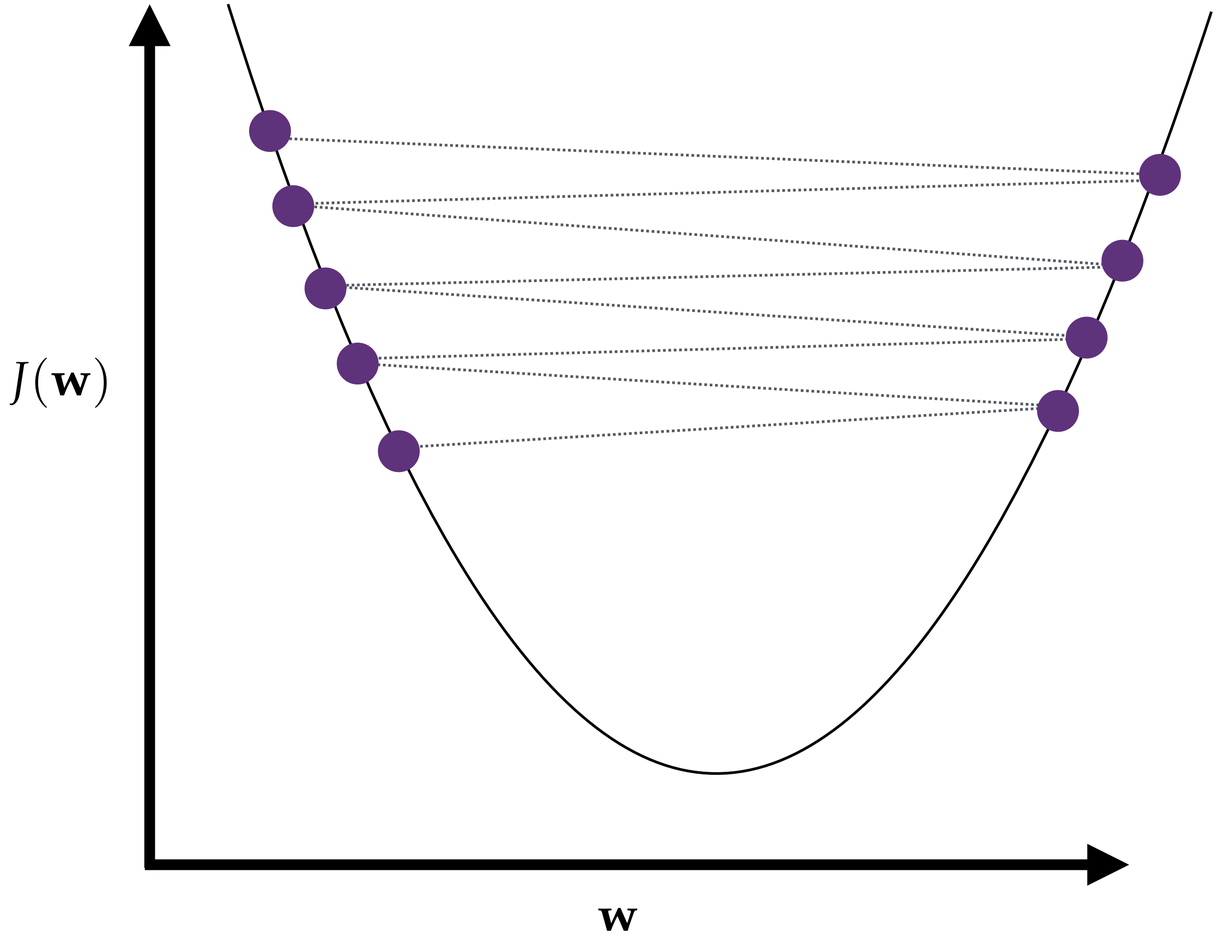
- Look at the slope, and follow the direction of the steepest slope
- How far do I go once I choose the direction (stepsize)? How many "steps" should I take?



Stepsize α



Too Small?



Too Large?

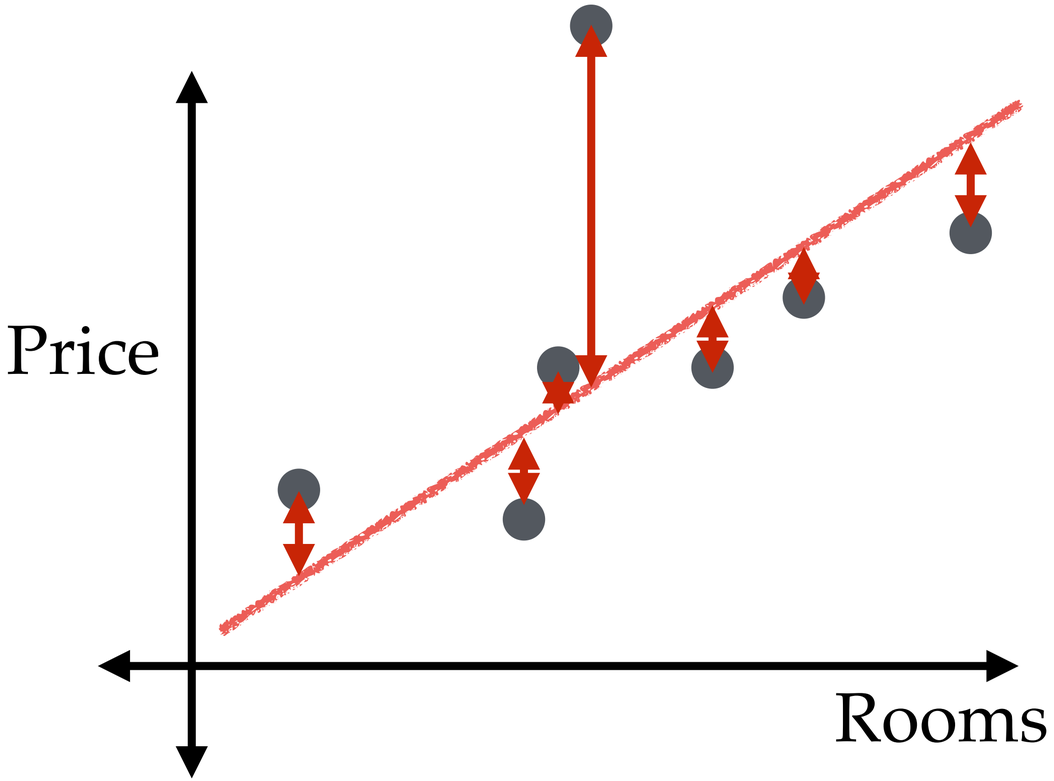
Gradient Descent Procedure

Given an objective function *J*, learning rate (step size) α, and number of iterations T.

1. Pick a starting value (for instance, random) for $\mathbf{w}$ and w_0
2. For T iterations, for all j = *0,1,…,D*, do:

$$w_j^{t+1} = w_j^t - \alpha \left. \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \right|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

Gradient descent in our Regression problem



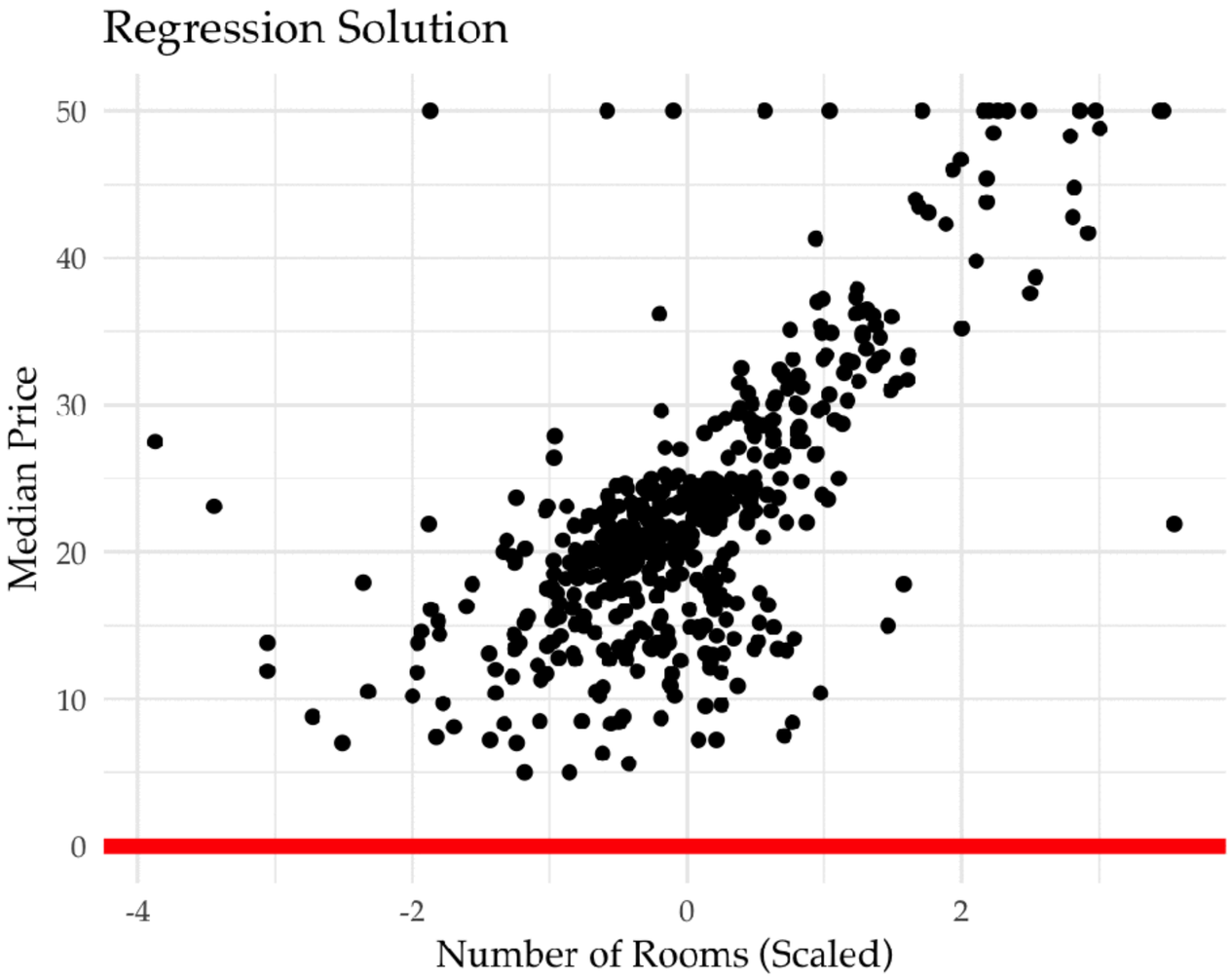
$$J(\mathbf{w}, w_0) = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i)^2$$

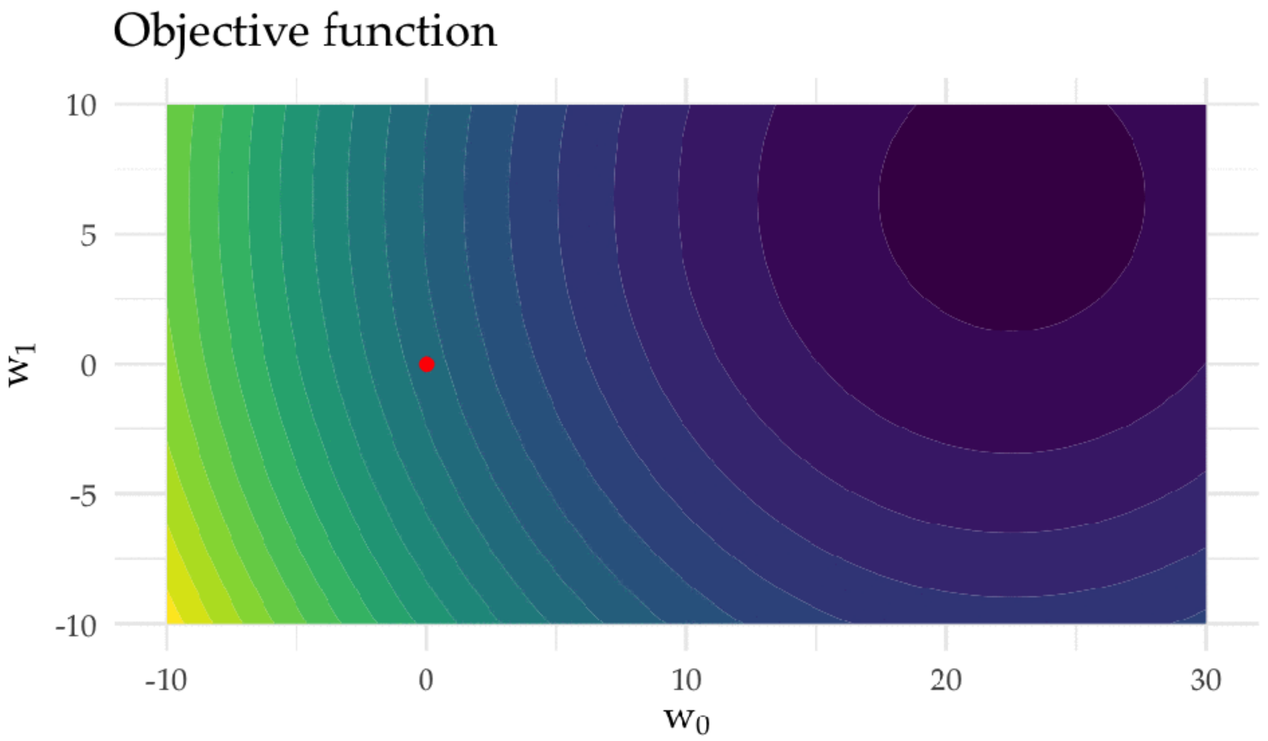
Derive the gradient:

$$\frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} =$$

$$\frac{\partial J(\mathbf{w}, w_0)}{\partial w_0} =$$

Linear Regression: Visualizing the procedure





Practice Question: Maximization

Recall the update we use in gradient descent is

$$w_j^{t+1} = w_j^t - \alpha \left. \frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} \right|_{\mathbf{w}, w_0 = \mathbf{w}^t, w_0^t}$$

Suppose we want to maximize the function J. How should the update be different?

Stochastic Gradient Descent

$$\frac{\partial J(\mathbf{w}, w_0)}{\partial w_j} = \frac{1}{N} \sum_{i=1}^{N} 2(\mathbf{x}_i^\top \mathbf{w} + w_0 - y_i) x_i^{(j)}$$

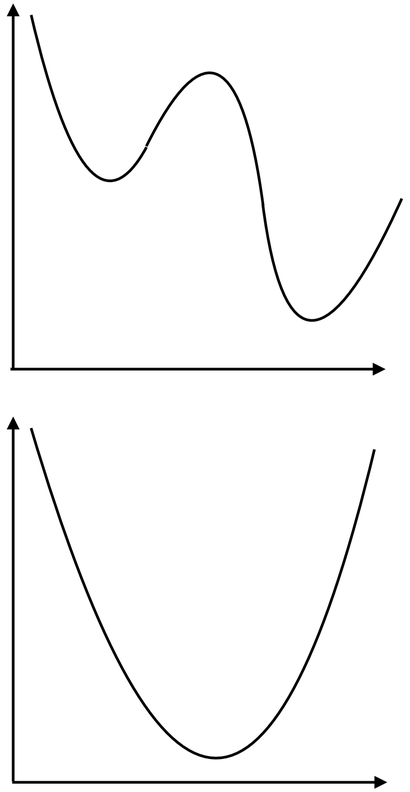
- To calculate the full gradient, we need to sum over all objects, before we take a step
- Instead we could estimate the gradient using one, or a few objects, and take a step using this estimate of the gradient
- The step is less "precise", but we can take many more steps in the same amount of time
- *Epoch*: visiting all the data once
- So in stochastic gradient descent, we do updates within an epoch, while regular gradient descent does only one update per epoch.

Extensions

- Fixing the step size seems naïve: smarter choices
  - Second order methods: take the curvature of the function into account
  - Line search
  - Momentum
- Many other descent procedures, for instance, conjugate gradient
- Note: using this gradient descent iterative procedure was not necessary here! There is a closed form/analytic solution! For our next model, this will not be the case.

Does GD lead to a good/the best solution?

- Depends on the cost function and function class: What does the objective function look like?
- For convex functions, with the right settings, and the right amount of patience, we can reach the global minimum.
- The global minimum is the classifier with the minimal value for the cost function, which may not be the best solution for the problem! Recall: we want the solution to work well for the *expected* loss



$$\min_{h \in \mathcal{H}} \mathbb{E}_{X,Y}[L(h(X), Y)]$$

We looked at regression, but what about classification?

THANKS