Machine Learning

Lecture 2 – Linear Regression, Training & Loss

Dr SHI Lei

Last Lecture

- Definition
- Lifecycle
- Types of ML Systems
- Key Terminologies

Last Lecture

Definition

"A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E."

- Tom Mitchell, 1997

It could be used as a design tool to help us think clearly about what data to collect (E), what decisions the software needs to make (T) and how we will evaluate its results (P).

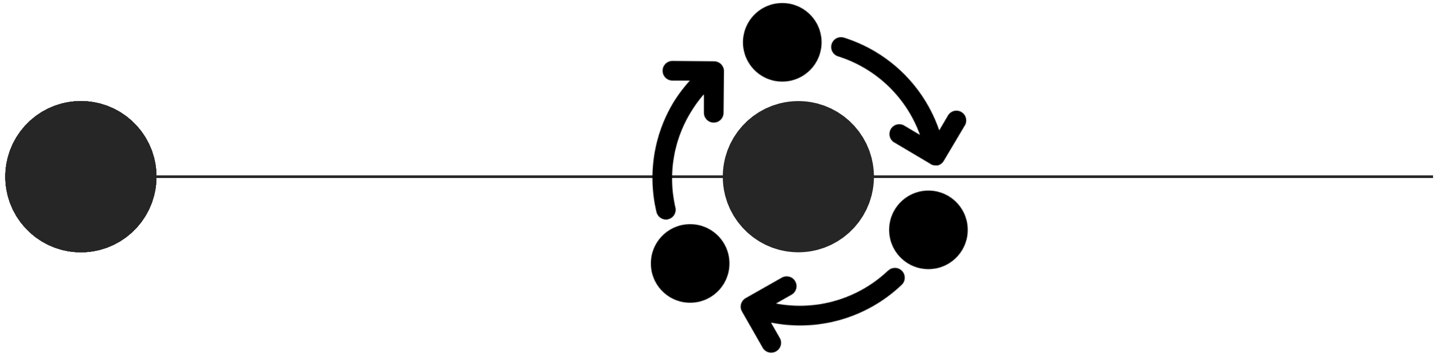
Last Lecture

7 steps in Machine Learning Lifecycle

1. Gathering data
2. Preparing data
3. Choosing model
4. Training
5. Evaluation
6. Hyper-parameter tuning
7. prediction

Last Lecture

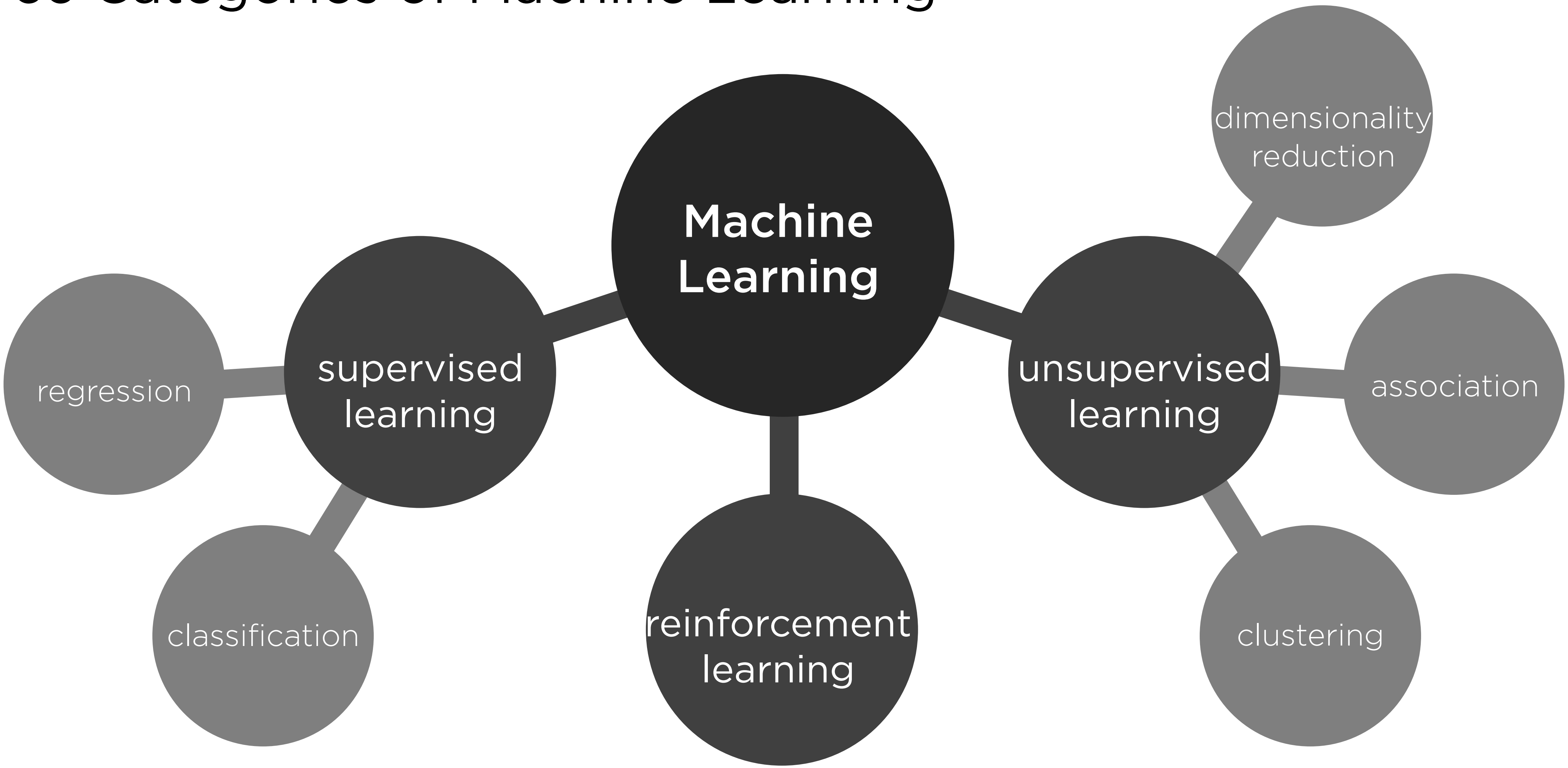
Training / Learning



Using data to answer questions

Last Lecture

Three Categories of Machine Learning



Last Lecture

Key Machine Learning Terminology

- **Example**: a particular instance of data, $\mathbf{x}$
  - **Label**: the variable to predict ($\mathbf{y}$)
  - **Features**: input variables describing data ($\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n\}$ or vector $\mathbf{X}$)
  - **Labelled example** has {features, label}: $(\mathbf{x}, \mathbf{y})$ to train the model
  - **Unlabelled example** has {feature, ?}: $(\mathbf{x}, ?)$ to predict on new data
- **Model** maps *examples* to predict labels: $\hat{y}$

  defined by internal parameters, which are learned
  - **Training** - creating or learning the model.
  - **Inference** - applying the trained model to unlabelled examples.

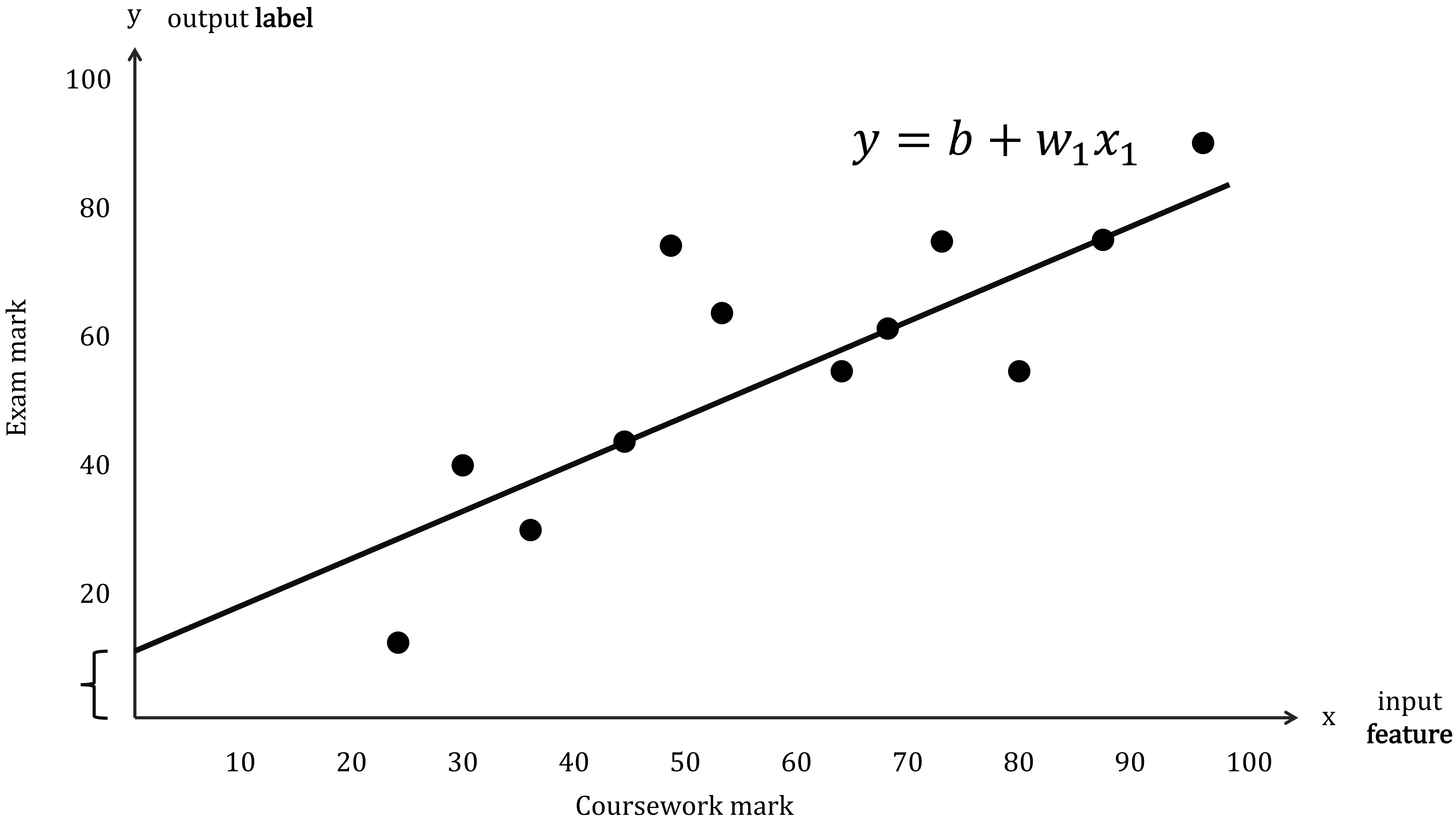
Today

- Linear Regression
- Training & Loss

Linear Regression

Linear Regression

A method for finding the straight line or hyperplane that best fits a set of points.



y
output **label**
100
80
60
40
20
Exam mark
$y = b + w_1 x_1$
x
input
feature
10
20
30
40
50
60
70
80
90
100
Coursework mark

$$y = b + w_1 x_1 \quad \Big| \quad \hat{y} = b + w_1 x_1$$

$\boldsymbol{\hat{y}}$ the predicted label (a desired output).

b the bias (the y-intercept), sometimes referred to as w_0.

$\mathbf{w_1}$ the weight of feature 1 (slope).

$\mathbf{x_1}$ a feature (a known input).

To infer (predict) the exam mark $\hat{y}$ for a new coursework mark value x_1, just substitute the x_1 value into this model.

$$\hat{y} = b + w_1 x_1$$

$$\hat{y} = b + w_1 x_1 + \mathrm{w}_2 \mathrm{x}_2 + \mathrm{w}_3 \mathrm{x}_3 + \ldots$$

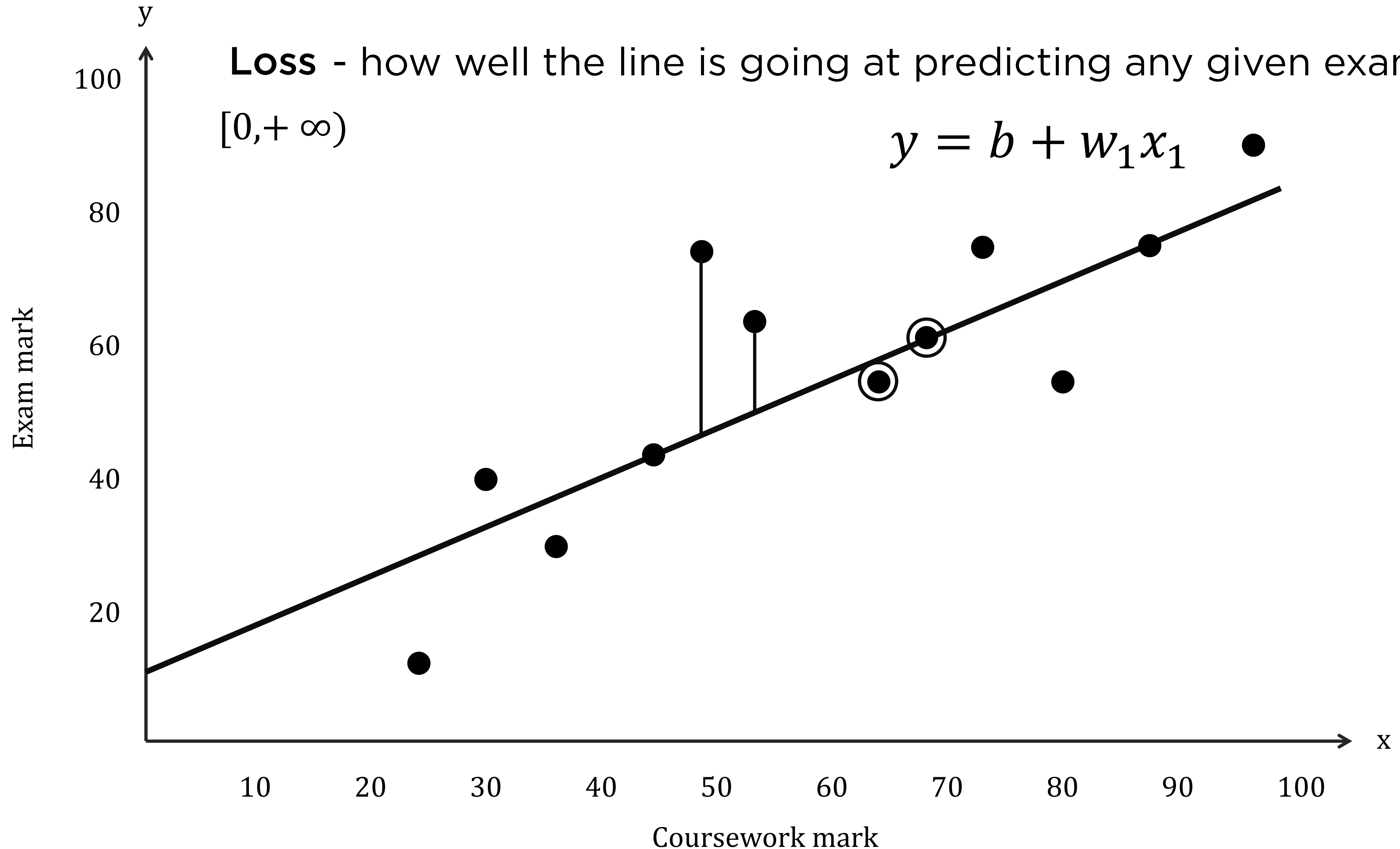
Is this a good line?

Loss

Loss - how well the line is going at predicting any given example.

$[0, +\infty)$

$$y = b + w_1 x_1$$



How do we define Loss?

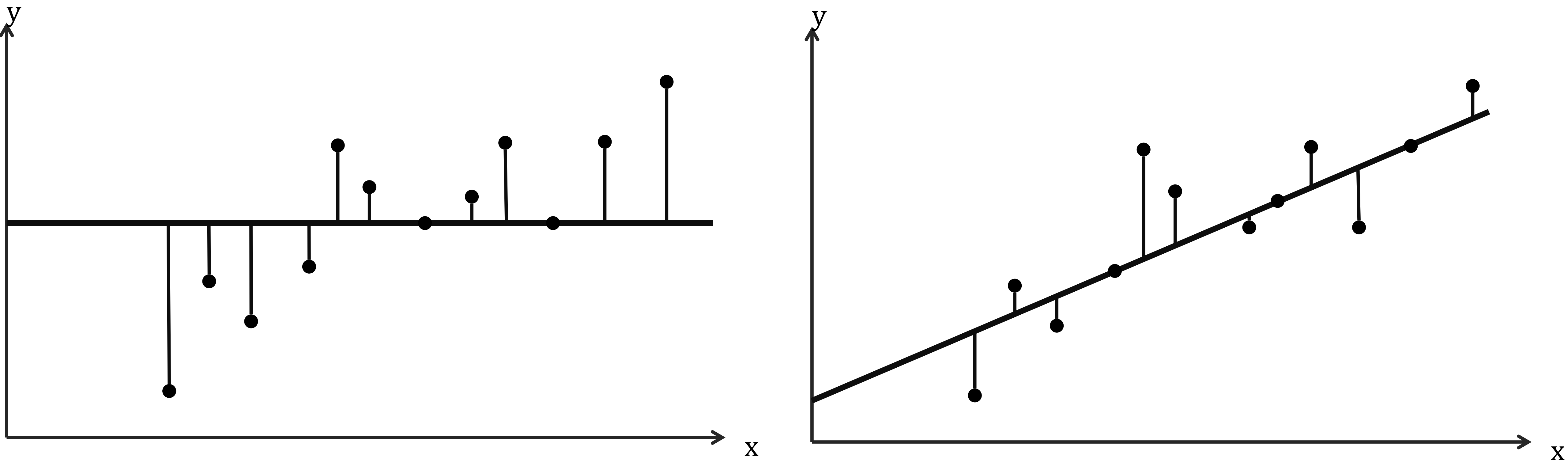
Training & Loss

Training a model: learning (determining) good values for all weights and the bias from labelled examples.

Loss: the penalty for a bad prediction.

Empirical Risk Minimisation: the process of examining many examples and attempting to find a model that minimise loss.

Goal of training: to find a set of weights and biases that have <u>low loss</u>, on average, across all examples.


y
x
y
x

Squared loss (L_2 Loss)

= the square of the difference between the label and the prediction

= $(\text{observation} - \text{prediction}(x))^2$

= $(y - \hat{y})^2$

Mean square error (MSE)

$$MSE = \frac{1}{N} \sum_{(x,y)\in D} (y - prediction(x))^2$$

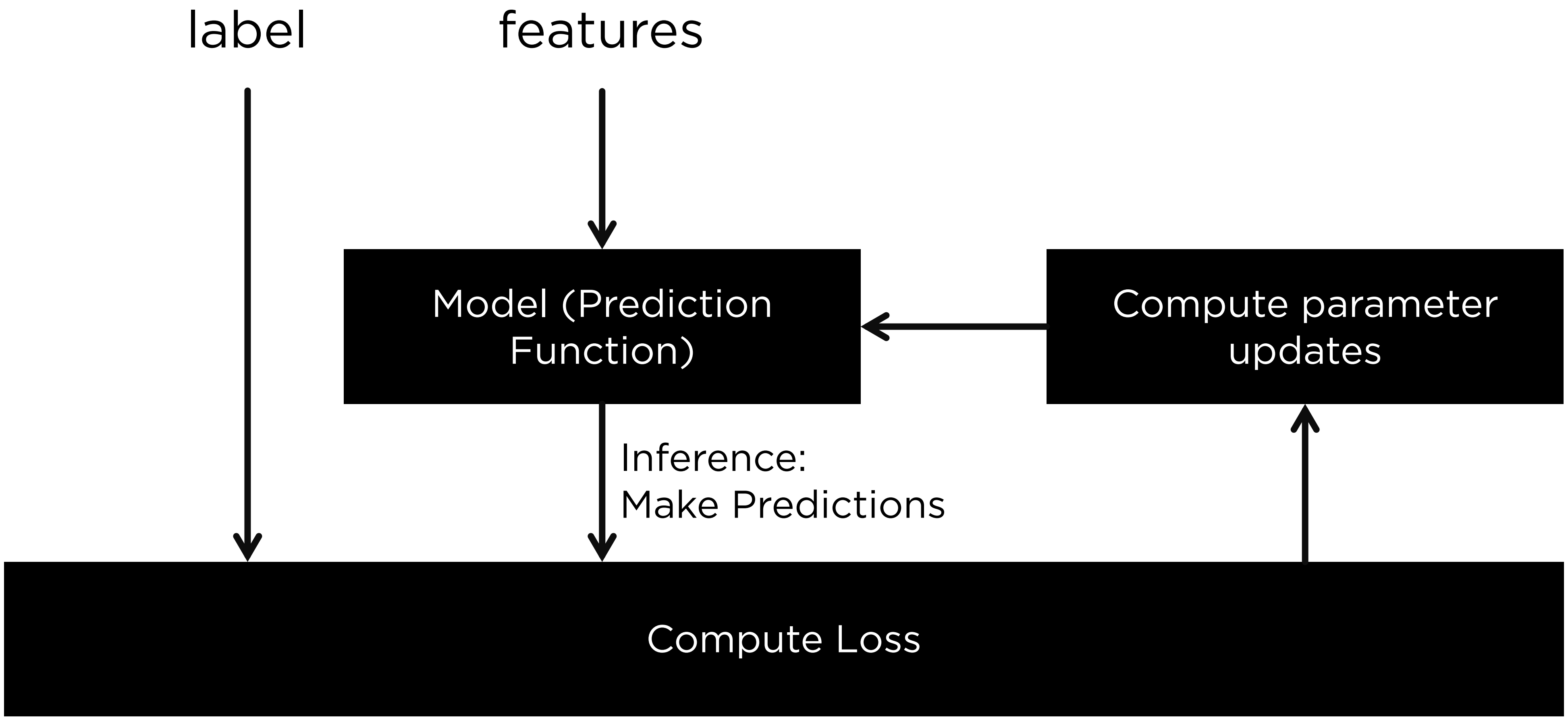
- (x, y) is an example where
  - x is the set of features used by the model to make predictions.
  - y is the example's label
- $\mathrm{prediction(x)}$ is a function of the weights & bias in combination with the set of features x.
- D is a dataset containing many labelled examples - (x, y) pairs.
- N is the number of examples in D.

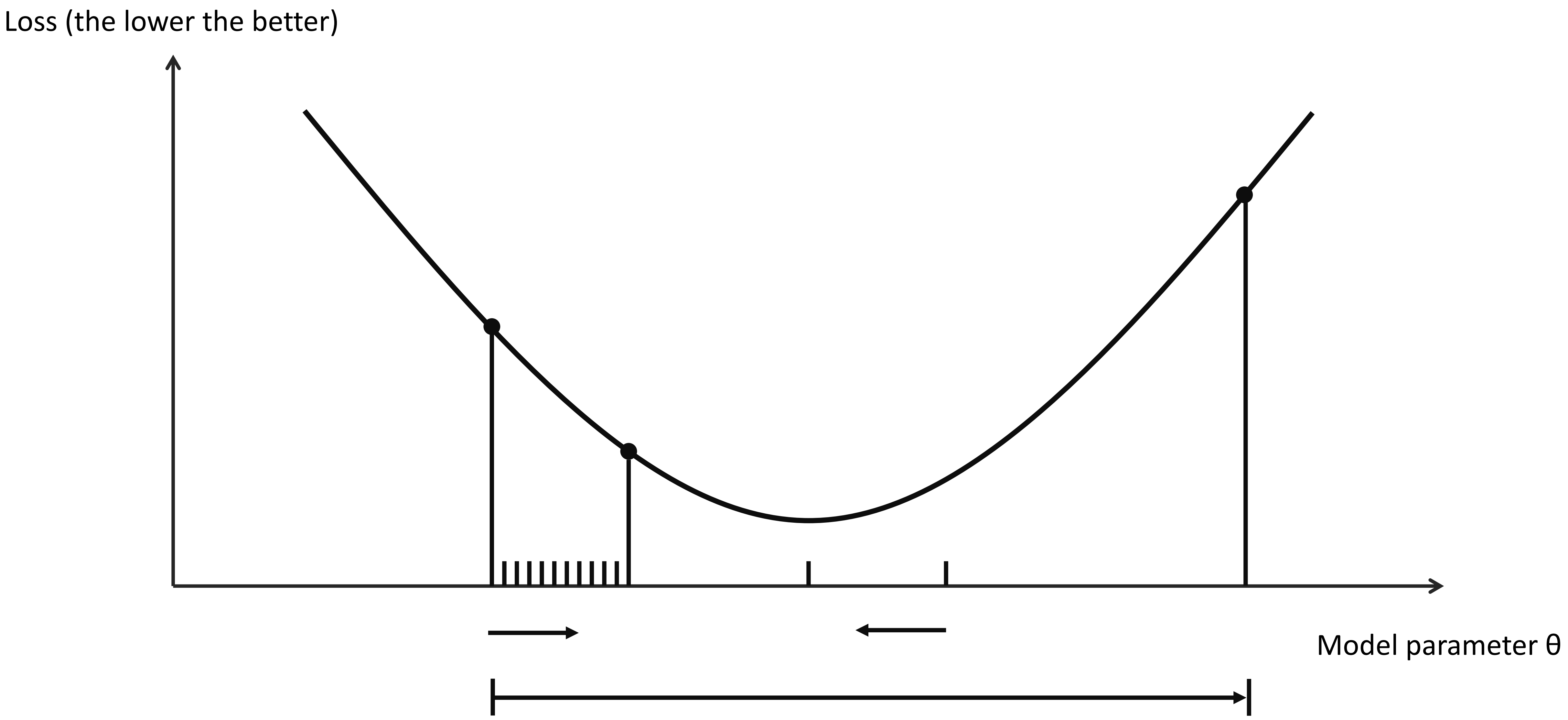
Reducing Loss

Reducing Loss

- Hyperparameters are the configuration settings used to tune how the model is trained.
- Derivative of $(\mathbf{y}\text{-}\,\hat{y})^2$ with respect to the weights and biases tells us how loss changes for a given example
  - Simple to compute and convex
- So we repeatedly take small steps in the direction that minimises loss
  - We call these Gradient Steps (but they are really negative Gradient Steps)
  - This strategy is called **Gradient Descent**

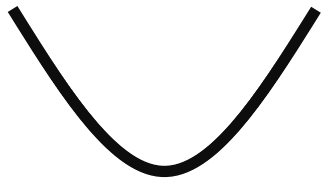
Block Diagram of Gradient Descent: an iterative approach



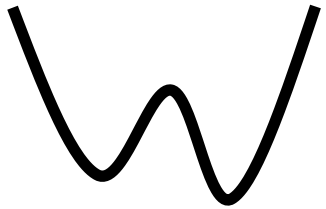

Loss (the lower the better)
Model parameter θ

Weight Initialisation

- For convex problem, weights can start anywhere (say, all 0s)
  - Convex: think of a bowl shape
  - Just one minimum



- Foreshadowing: not true for neural nets
  - Non-convex: think of an egg crate
  - More than one minimum
  - Strong dependency on initial values



Efficiency of Reducing Loss

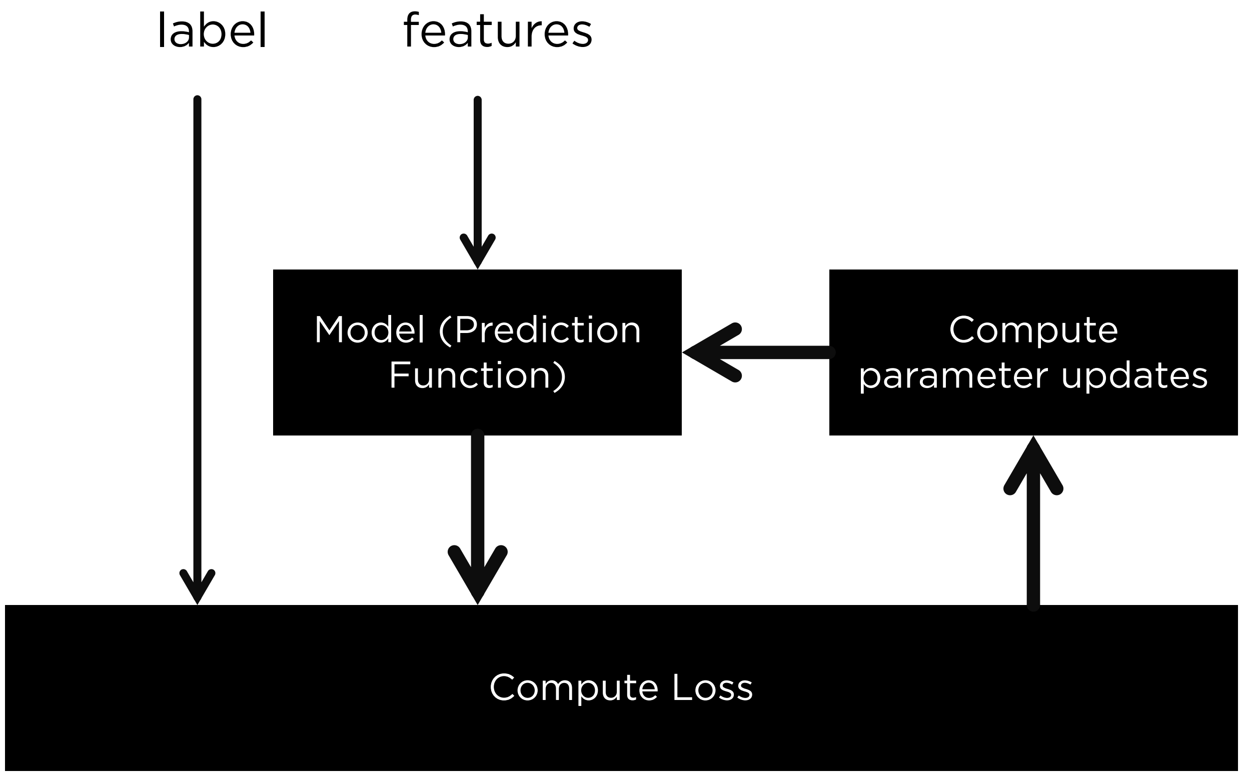
- Could compute gradient over entire dataset on each step, but this turns out to be unnecessary

- Computing gradient on small data examples works well
  - On every step, get a new random sample

- **Stochastic Gradient Descent**: one example at a time

- **Mini-Batch Gradient Descent**: batches of 10 – 1000
  - Loss & gradients are averaged over the batch

An iterative approach to train a model



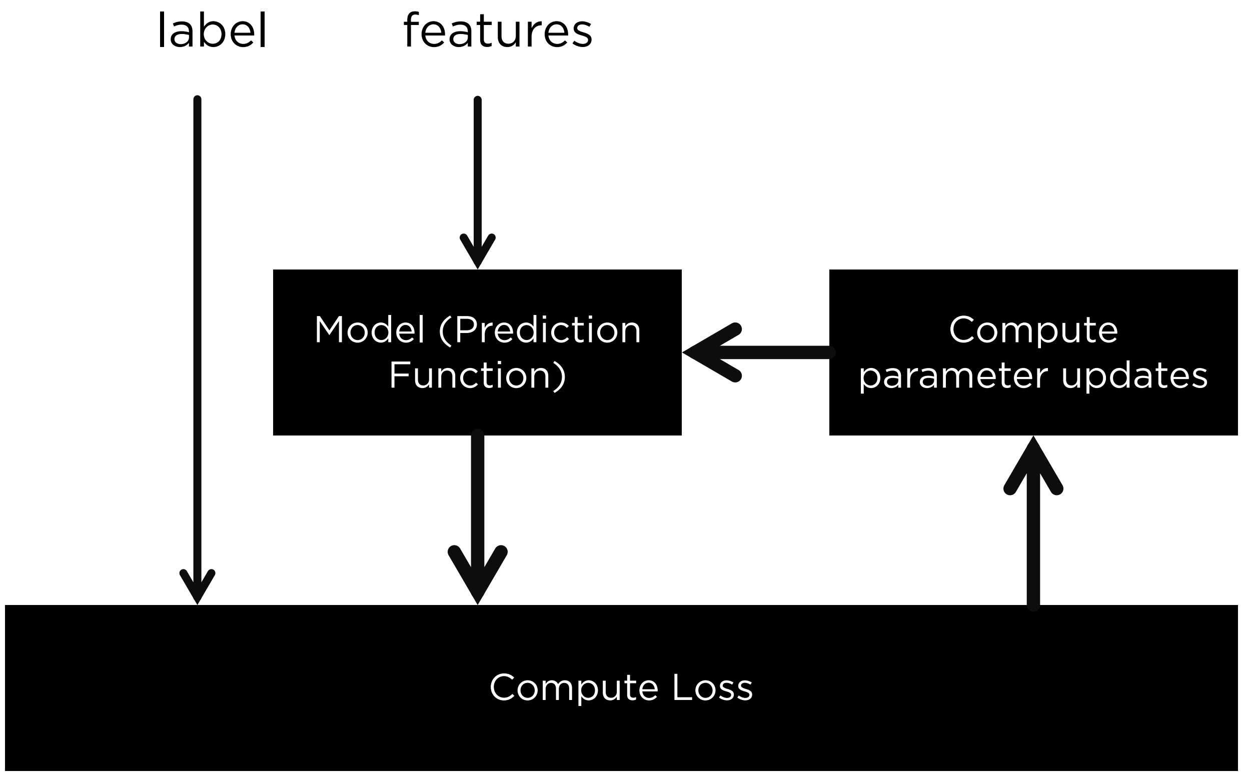
$$\hat{y} = \mathrm{b} + \mathrm{w}_1\mathrm{x}_1$$

What initial values should we set for b and w_1?

- Starting values aren't important.
- Could choose pick random values, but we'll just take the following trivial values instead:

$$\mathrm{b} = 0$$
$$\mathrm{w}_1 = 0$$

An iterative approach to train a model



$$\hat{y} = b + w_1 x_1$$

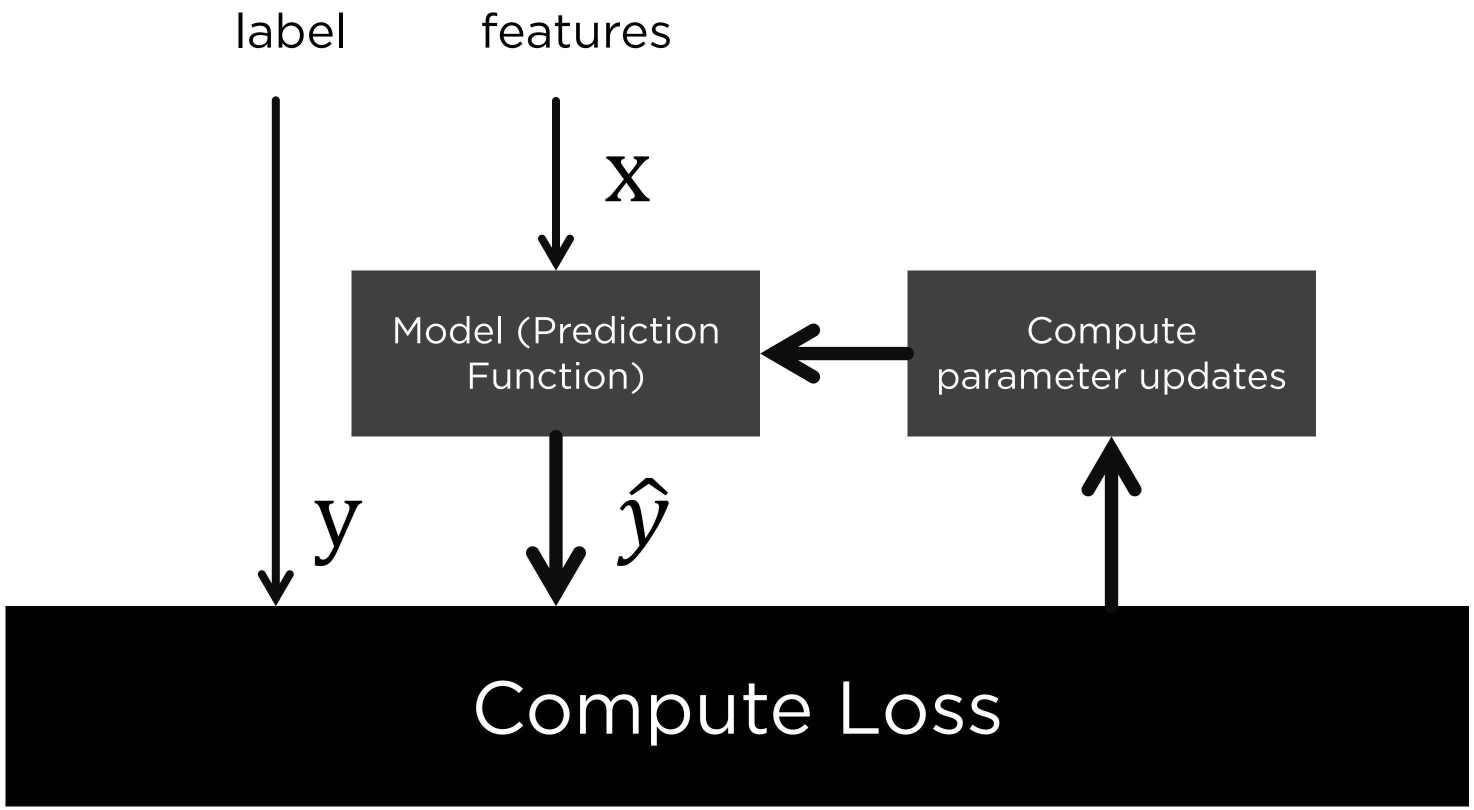
$$b = 0$$
$$w_1 = 0$$

$$\hat{y} = 0 + 0 \times x_1$$

$$x_1 = 10$$

$$\hat{y} = 0 + 0 \times 10$$
$$= 0$$

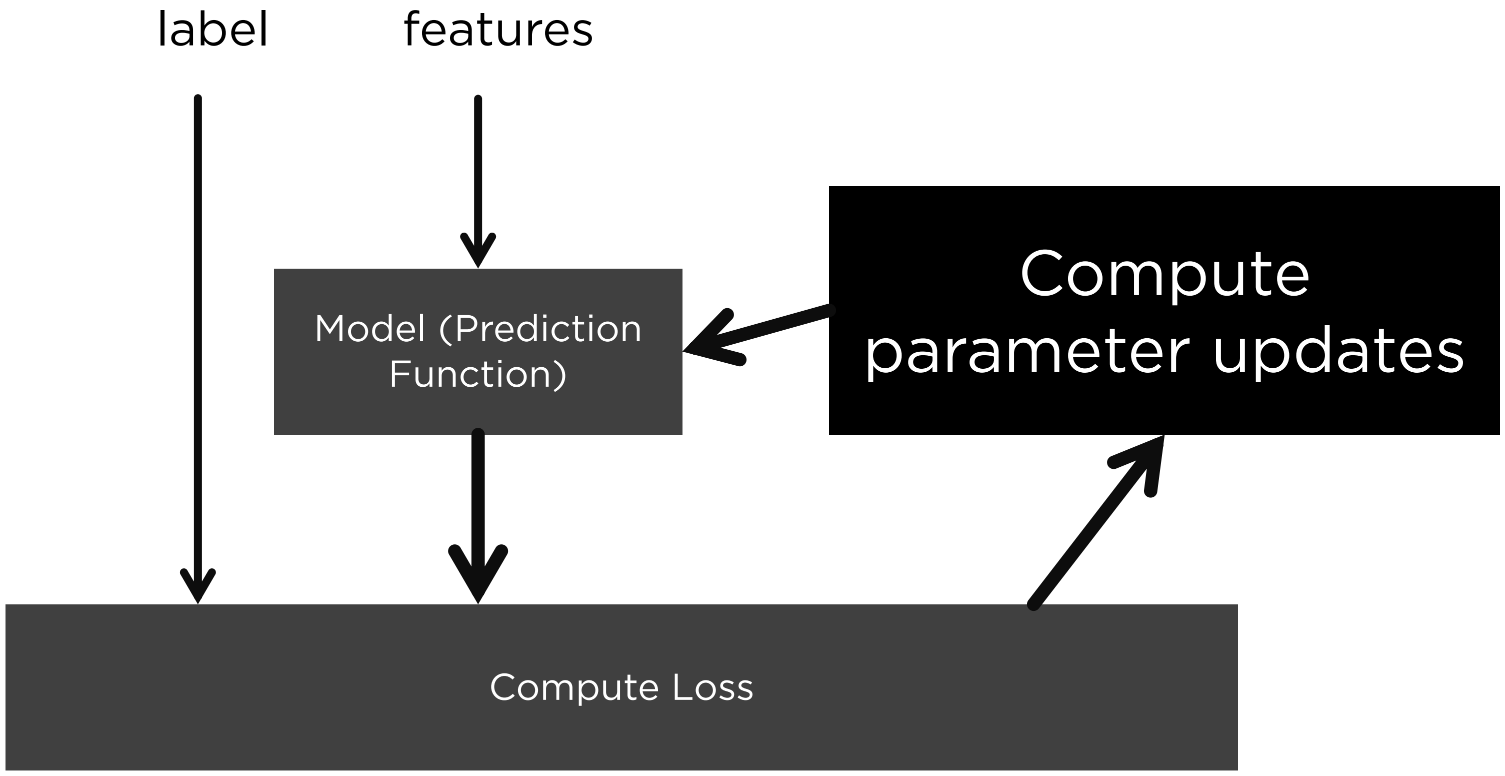
An iterative approach to train a model



Loss Function

- $\hat{\boldsymbol{y}}$ **:** the model's prediction for features x
- **y:** the correct label corresponding to features x.

An iterative approach to train a model



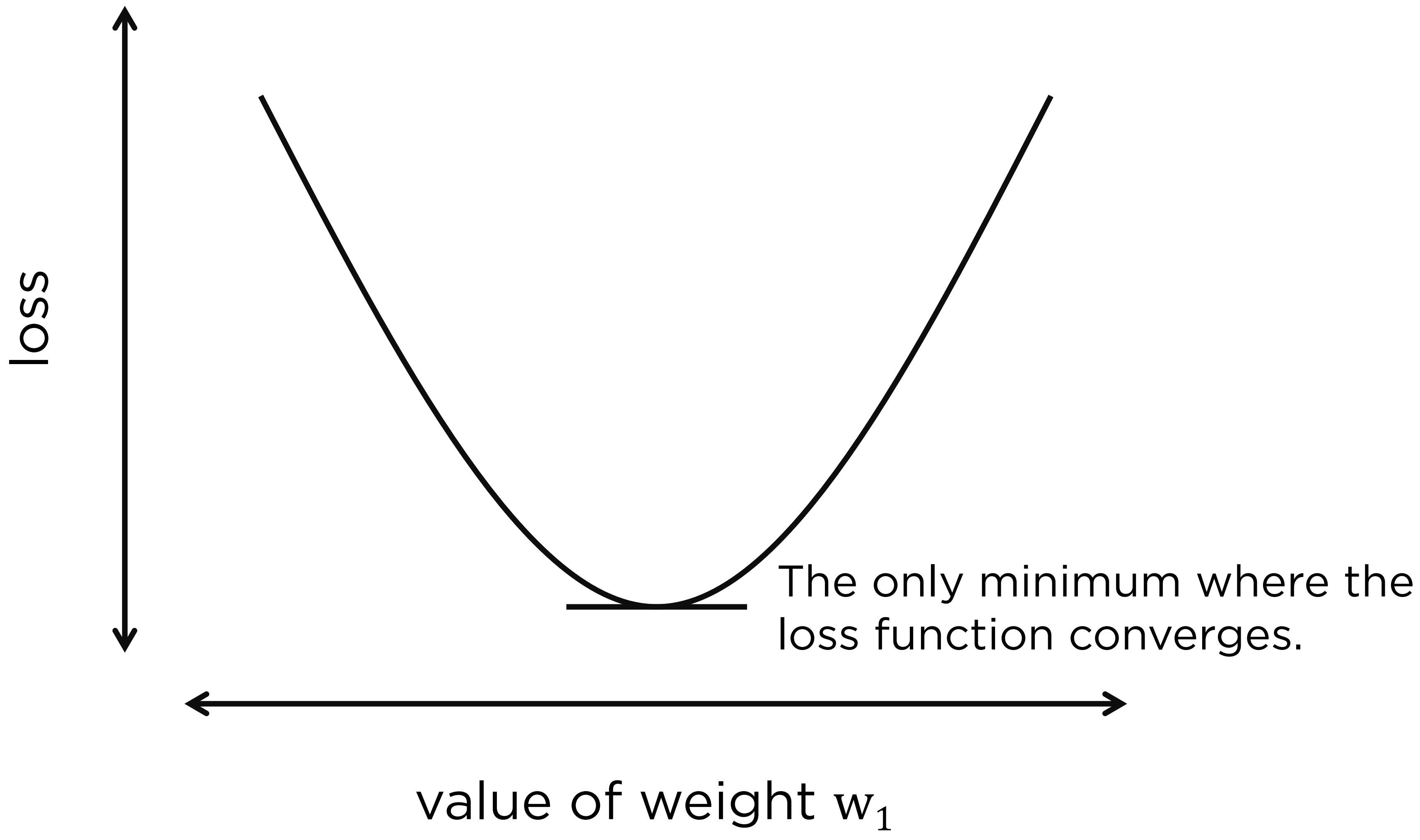
- Examines the value of loss function
- Generates new values for b and w_1

$$\hat{y} = \mathrm{b} + \mathrm{w}_1\mathrm{x}_1$$

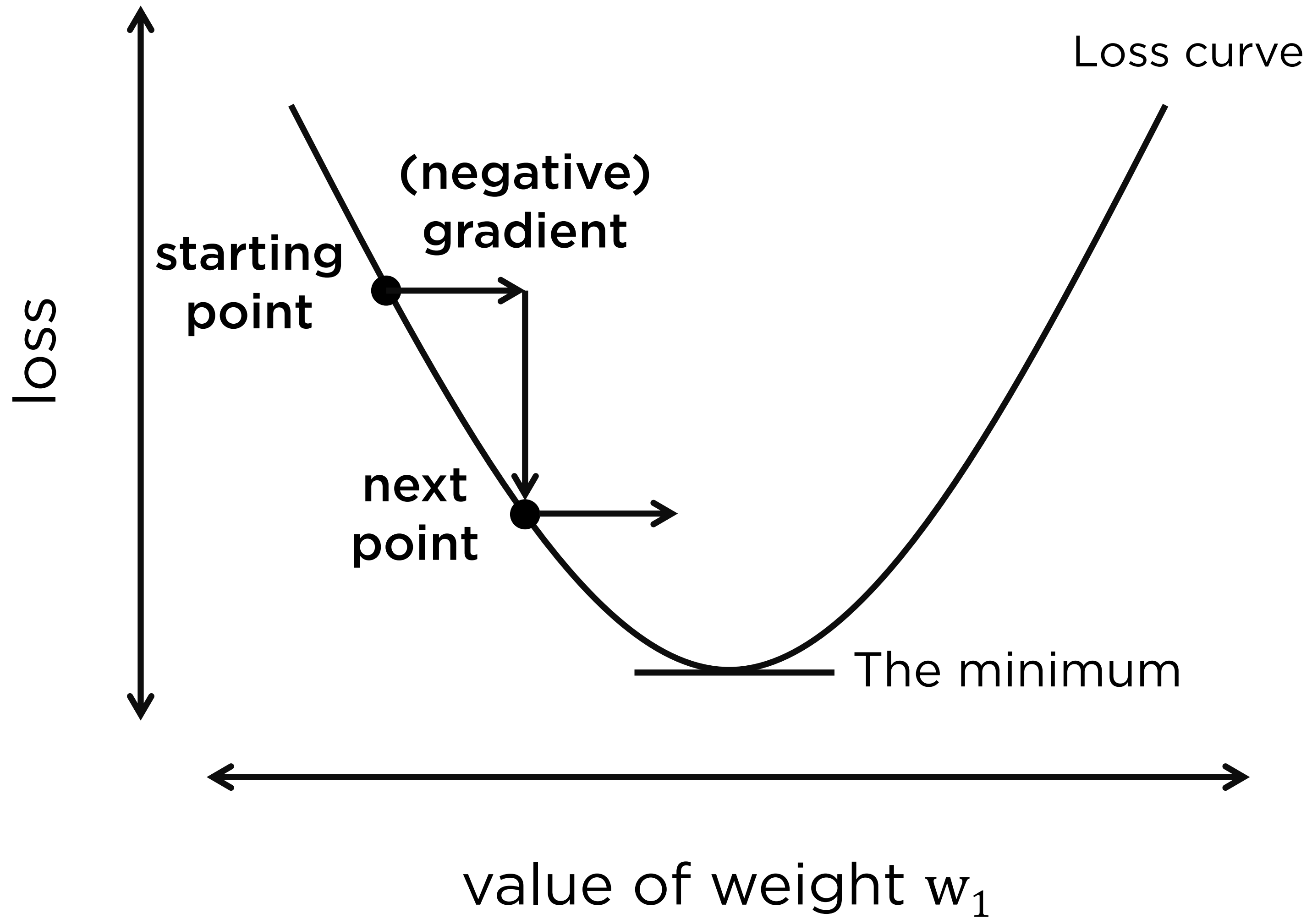
An iterative approach to train a model

Compute
parameter updates

Regression problems yield convex loss vs weight plots

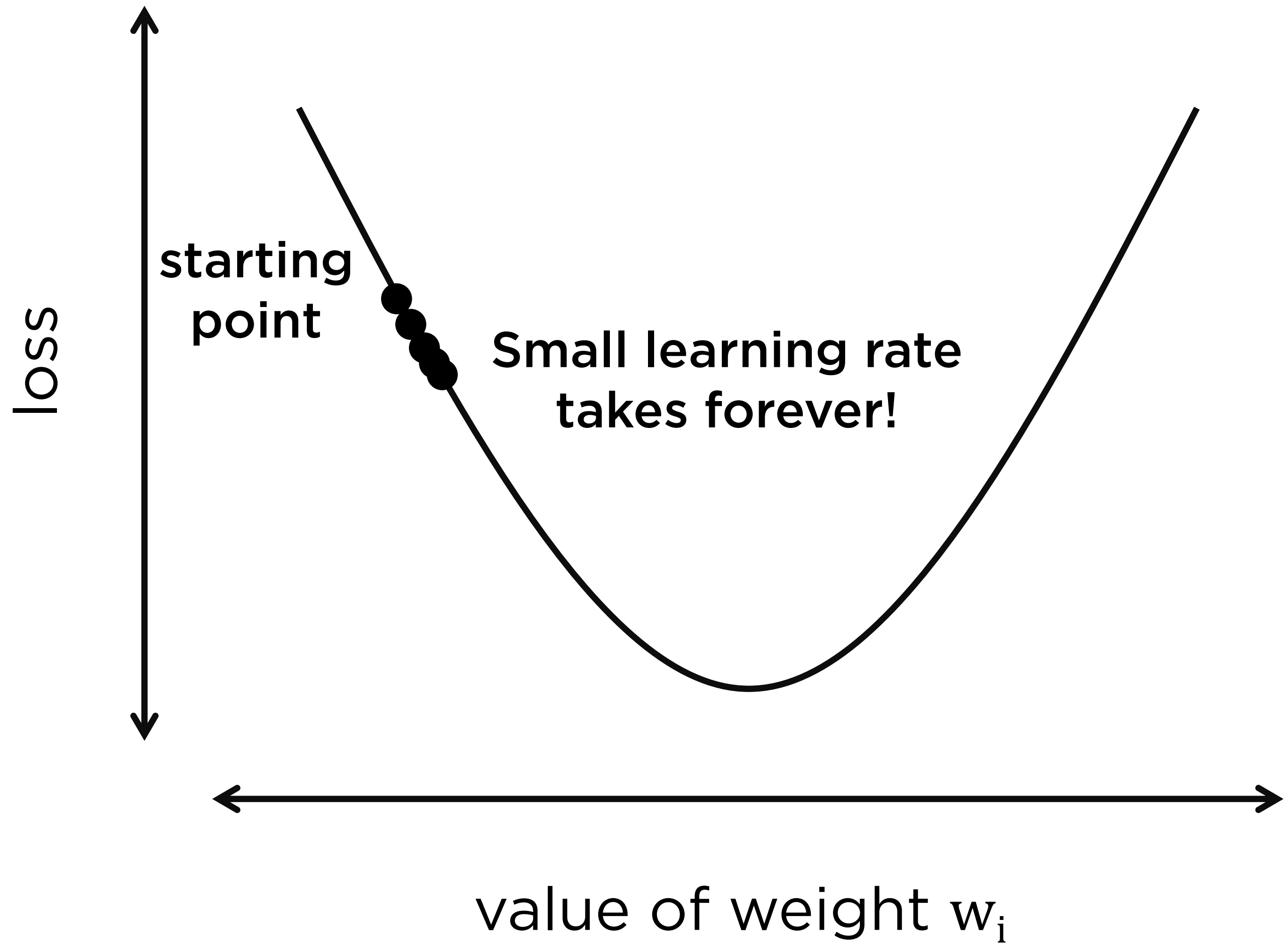


Gradient Descent



Learning Rate

too small

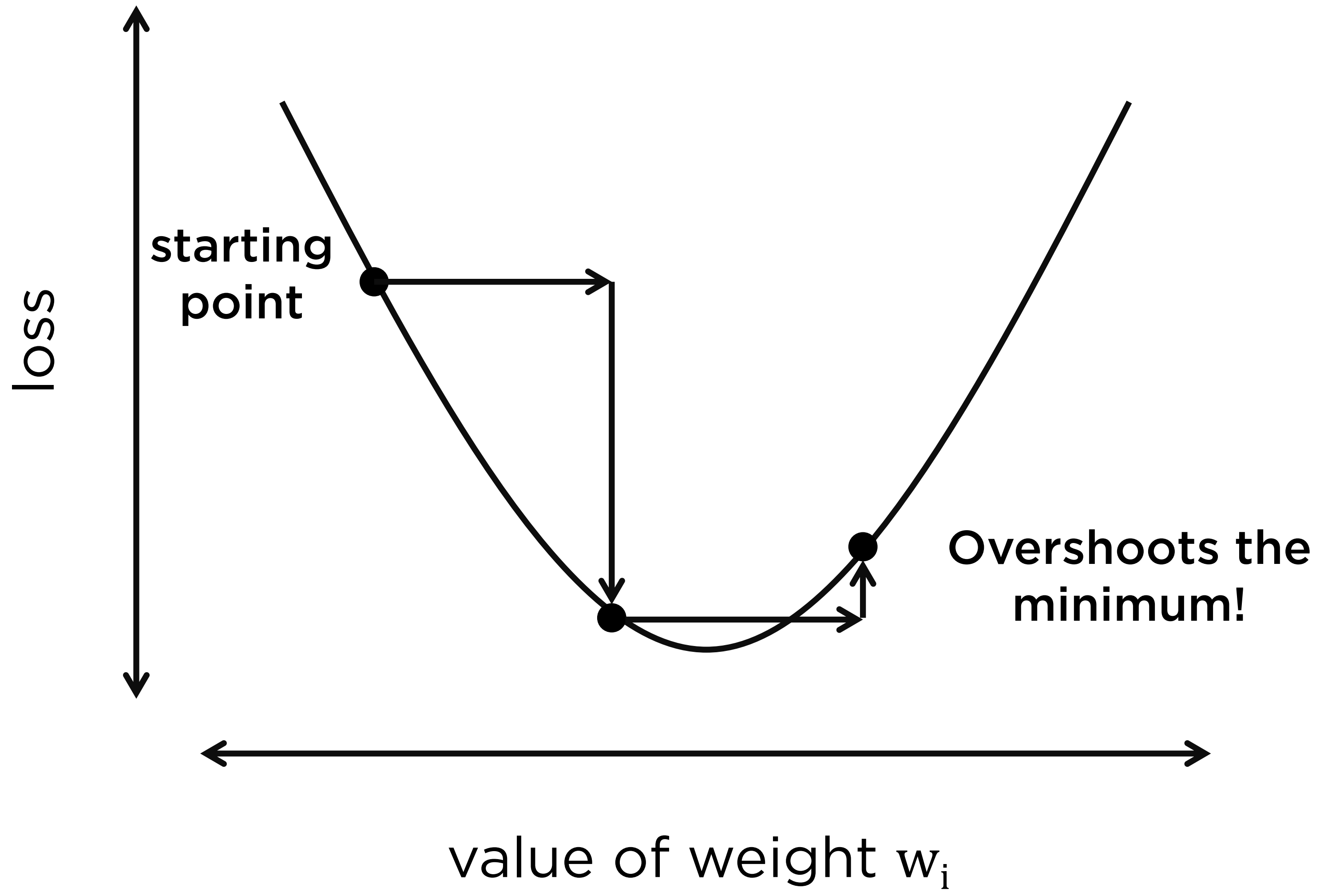


IF
Gradient magnitude = 2.5
Learning rate = 0.01

Then
The next point = 0.025

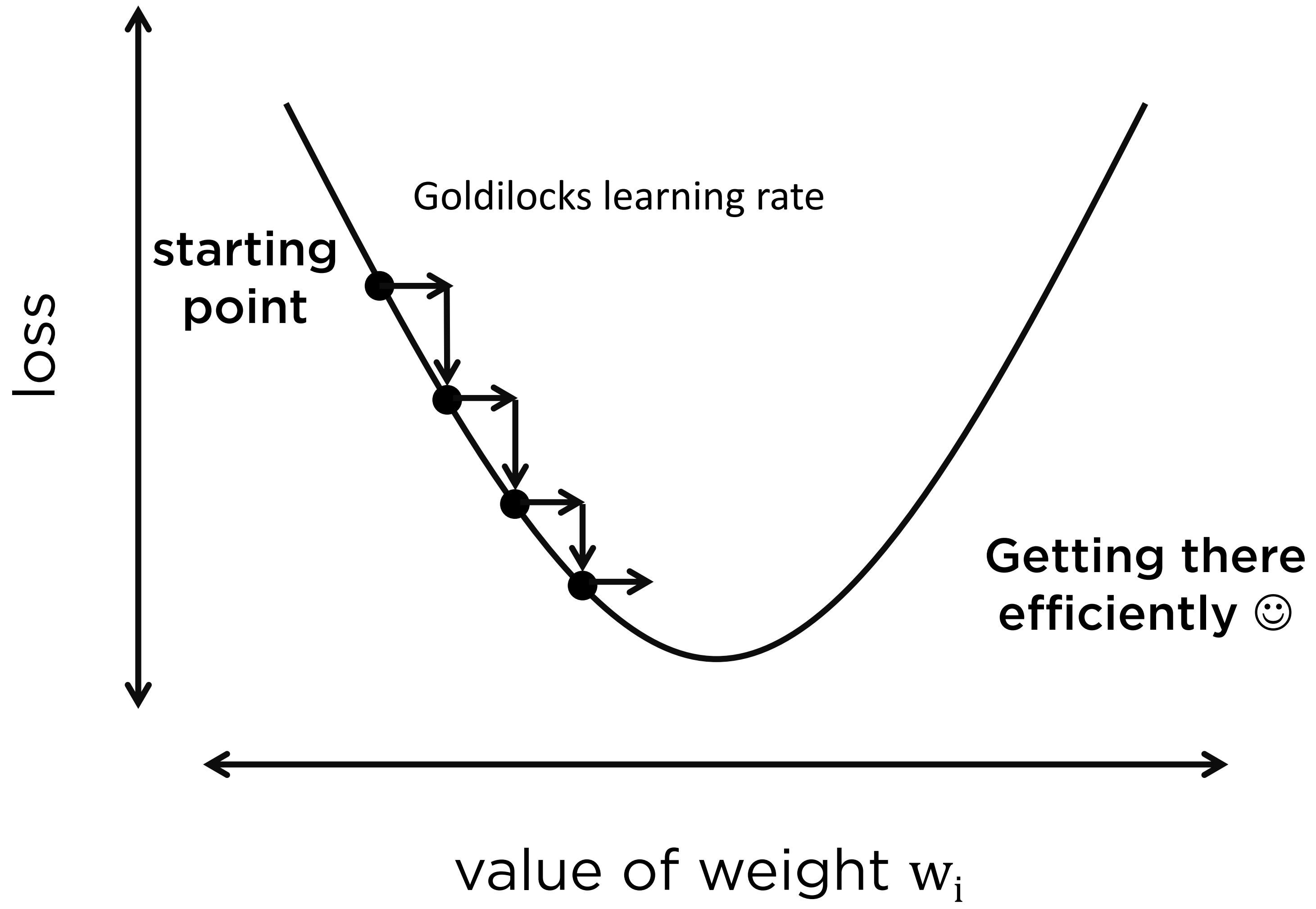
Learning Rate

too large



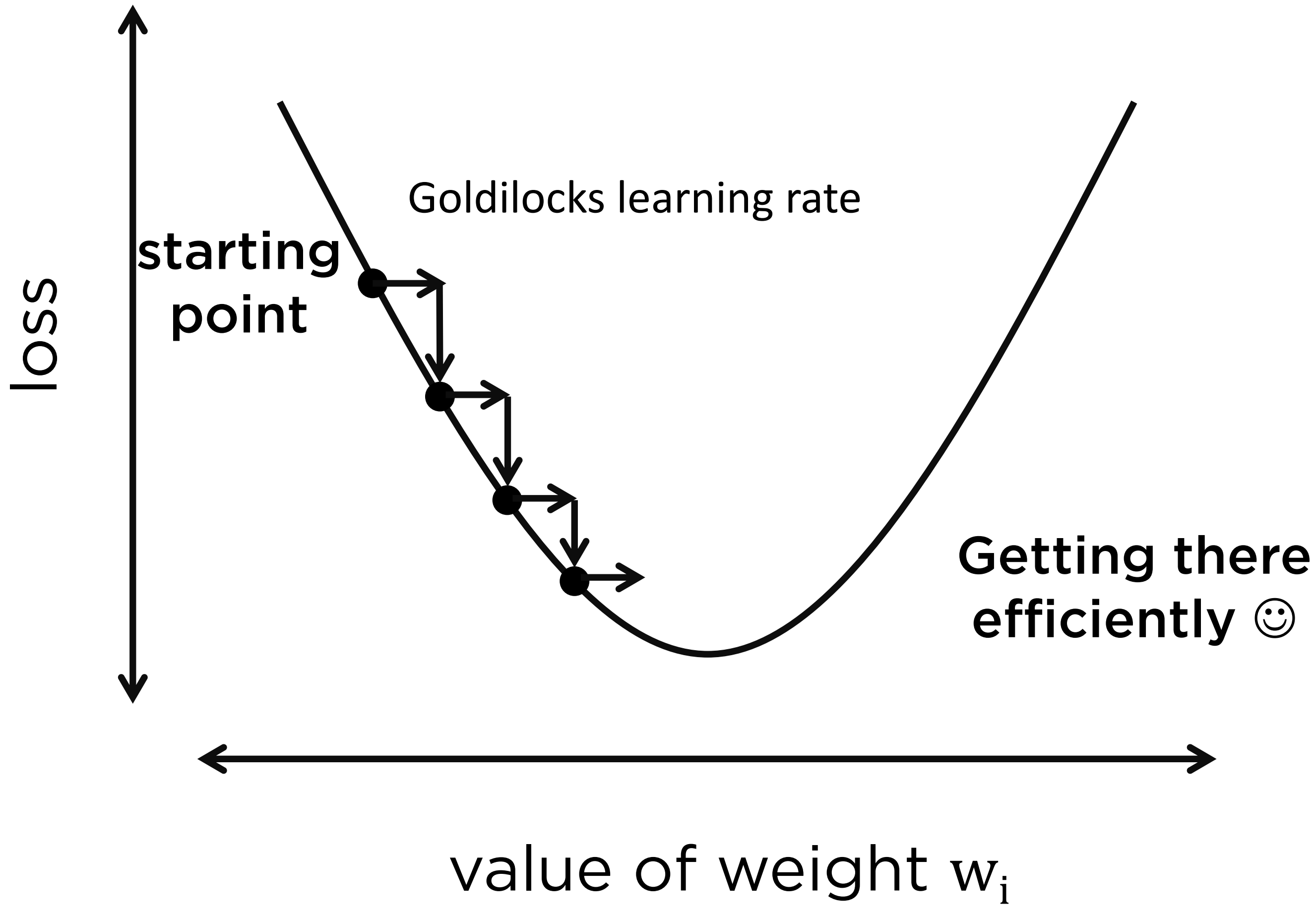
Learning Rate

Just about right (the **Goldilocks** learning rate)



Learning Rate

Just about right (the **Goldilocks** learning rate)



- The ideal learning rate in one-dimension is

$$\frac{1}{f(x)''}$$

(the inverse of the 2^{nd} derivative of f(x) at x).

- The ideal learning rate for 2 or more dimensions is the inverse of the **Hessian** (matrix of second partial derivatives).

Summary

Today

- Linear Regression
- Training & Loss
  - Training (an iterative approach)
  - Reducing Loss
  - Gradient Descent
  - Learning Rate

Homework

- Optimising learning rate (**Goldilocks** and **Hessian**)?
- Chapter 4 – sections 1 & 2 reading & programming exercises

Next Lecture

- Generalisation, Training & Test Set, Representation