COMP4650/6490 Document Analysis

Machine Learning Basics - Part I

ANU School of Computing

Administrative matters

- Labs
  - Lab2 solution released
  - No lab in this week
- Quiz 1
  - Marks and answers will be released later today
- Assignment 1
  - Due: 5pm Wednesday 16 August, AEST (UTC +10)
  - Extension application:
    24 hours before due date + supporting documentation

Topics covered in ML for NLP

- Machine Learning (ML) Basics
  - Linear / logistic regression, gradient descent, practical considerations in ML
- Representation in NLP
  - Sparse vectors, word2vec, latent semantic analysis (LSA), etc
- Clustering
- Deep Neural Networks (DNN)
  - Feedforward NN (MLP), RNN (LSTM, GRU), back-propagation
- Encoder-Decoder Models & Attention
- Transformers
  - Self-attention, multi-head attention, transformer architecture
- Pre-trained Language Models
  - Pre-training, transfer learning using pre-trained large language models (LLMs)
  - Capabilities, limitations and risks of LLMs

Australian National University

Outline

- Machine Learning Overview
  - Motivation, terminology, and notation
- Linear Regression
  - Motivating example
  - Expressing linear functions
  - Loss function for linear regression
  - Optimisation, gradient descent
  - Multiple linear regression
- Logistic Regression
  - Classification problem
  - The softmax function
  - Multinomial logistic regression, cross entropy loss

Australian National University

Outline

- Machine Learning Overview
  - Motivation, terminology, and notation
- Linear Regression
  - Motivating example
  - Expressing linear functions
  - Loss function for linear regression
  - Optimisation, gradient descent
  - Multiple linear regression
- Logistic Regression
  - Classification problem
  - The softmax function
  - Multinomial logistic regression, cross entropy loss

ML Overview

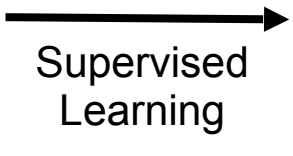
Motivation

- We want to create a program that can make predictions / inferences, e.g.
  - given the text of an email, predict would a human consider this email as spam
  - given the text of a movie review calculate a numeric review score
- One way to do this is by having the program learn from examples (called a training dataset)
  - For example, the program learns what kinds of emails get labelled as spam
  - In this example, the goal is to make predictions about new emails

ML Overview

Terminology

Inputs				Output
A1	A2	A3	A4	Y
-0.69	-0.72	Y	0.47	Healthy
-2.3	-1.2	N	0.15	Disease
0.32	-0.9	N	-0.76	Healthy
0.37	-1	Y	-0.59	Disease
-0.67	-0.53	N	0.33	Healthy
0.51	-0.09	Y	-0.05	Disease



$$Y = h(A_1, A_2, A_3, A_4)$$

Model
Hypothesis

- Supervised learning
  - Goal: given a training dataset with both the inputs and the outputs (i.e. the ground truth), find a function h of the inputs that returns the correct output (or *as close as possible*)
  - Categorical output ⇒ Classification problem
  - Scalar output ⇒ Regression problem
- Unsupervised learning, reinforcement learning, active learning, etc
  - Not covered in this course (except clustering)

ML Overview

Terminology

- Objects (or samples, observations, examples, data points)
  - e.g. documents, images, audio, video, medical diagnostics, etc.
- Variables (or features, attributes) describe an object
  - e.g. word occurrence, term frequency, tf-idf, pixel colour, etc.
- Dataset (or corpus, collection, set of examples)

	VAR 1	VAR 2	VAR 3	VAR 4	VAR 5	VAR 6	VAR 7	VAR 8	VAR 9	VAR 10	VAR 11	...
Object 1	0	1	2	0	1	1	2	1	0	2	0	...
Object 2	2	1	2	0	1	1	0	2	1	0	2	...
Object 3	0	0	1	0	1	1	2	0	2	1	2	...
Object 4	1	1	2	2	0	0	0	1	2	1	1	...
Object 5	0	1	0	2	1	0	2	1	1	0	1	...
Object 6	0	1	2	1	1	1	1	1	1	1	1	...
Object 7	2	1	0	1	1	2	2	2	1	1	1	...
Object 8	2	2	1	0	0	0	1	1	1	1	2	...
Object 9	1	1	0	1	0	0	0	0	1	2	1	...
Object 10	1	2	2	0	1	0	1	2	1	0	1	...

ML Overview

Notation

- Given data $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, find a good $f_{\mathbf{w}}(\boldsymbol{x})$
  - **Bold** denotes vectors
  - Predictions from the model are denoted with a hat: $\hat{y} = f_{\mathbf{w}}(\boldsymbol{x})$

Examples / Data points / Inputs	$\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \sim X$
Labels / Annotations / Targets	$y^{(1)}, \ldots, y^{(n)} \sim Y$
Predictor / Model	$f_{\mathbf{w}}(\boldsymbol{x}) : X \to Y$
Features of data point k	$\boldsymbol{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_d^{(k)})$

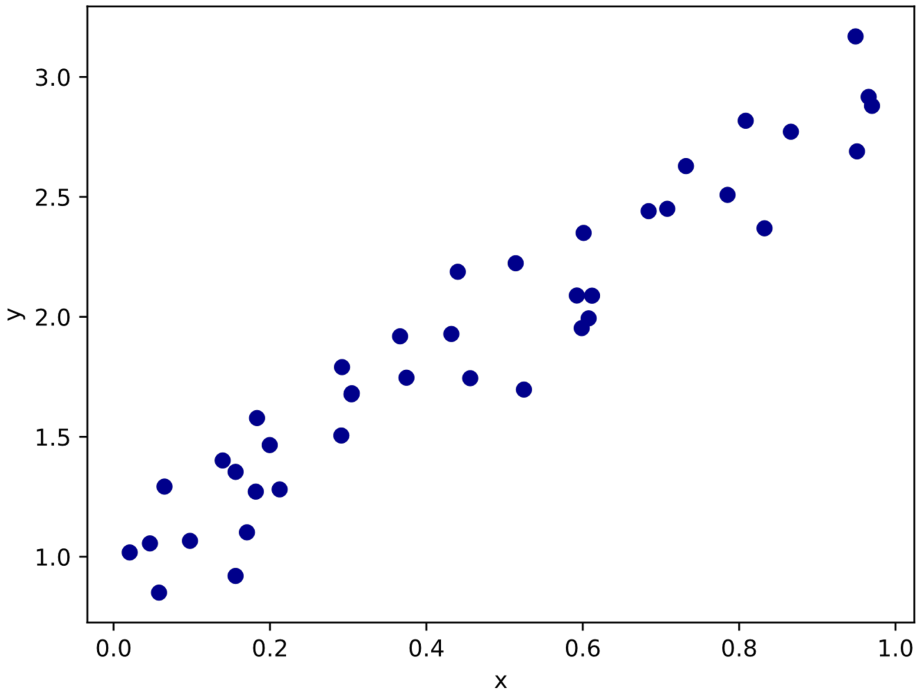
Australian National University

Outline

- Machine Learning Overview
  - Motivation, terminology, and notation
- Linear Regression
  - Motivating example
  - Expressing linear functions
  - Loss function for linear regression
  - Optimisation, gradient descent
  - Multiple linear regression
- Logistic Regression
  - Classification problem
  - The softmax function
  - Multinomial logistic regression, cross entropy loss

Linear Regression

Motivating example

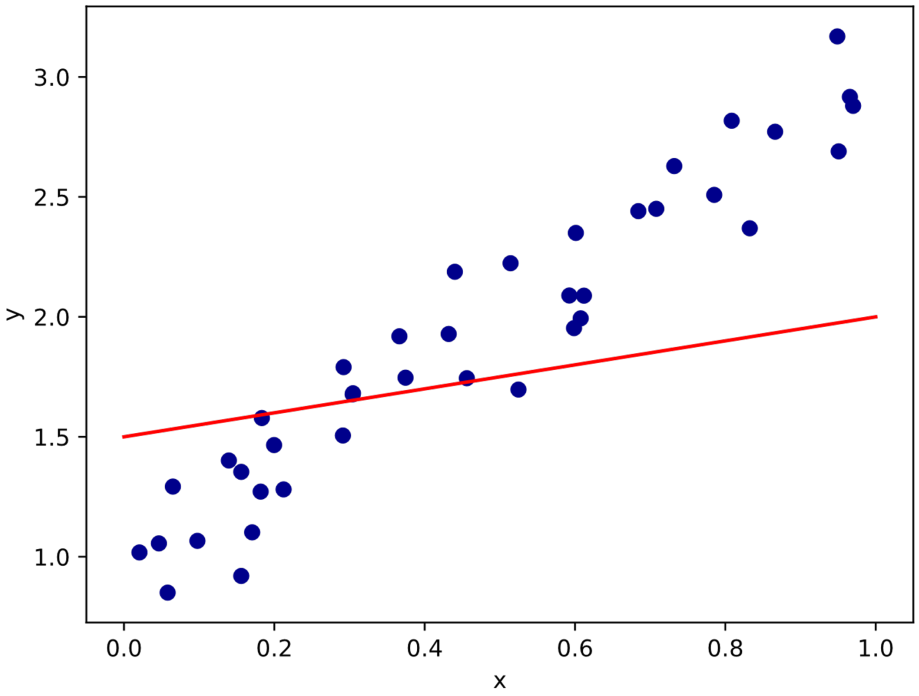


- What function would you choose?

- Training dataset
  - A collection of 40 data points (x, y) pairs
  - Each x value is labelled with its correct y value
- Goal
  - Predict the y value of a point with $x = 1.5$
- Approach
  - Create a program that can tell us what the y value of a new x is.
- Key idea
  - Find a *simple* function which explains the training dataset
  - For all of the x value of points in the training dataset, the output of our function should be *as close as possible* to the true y value.

Linear Regression

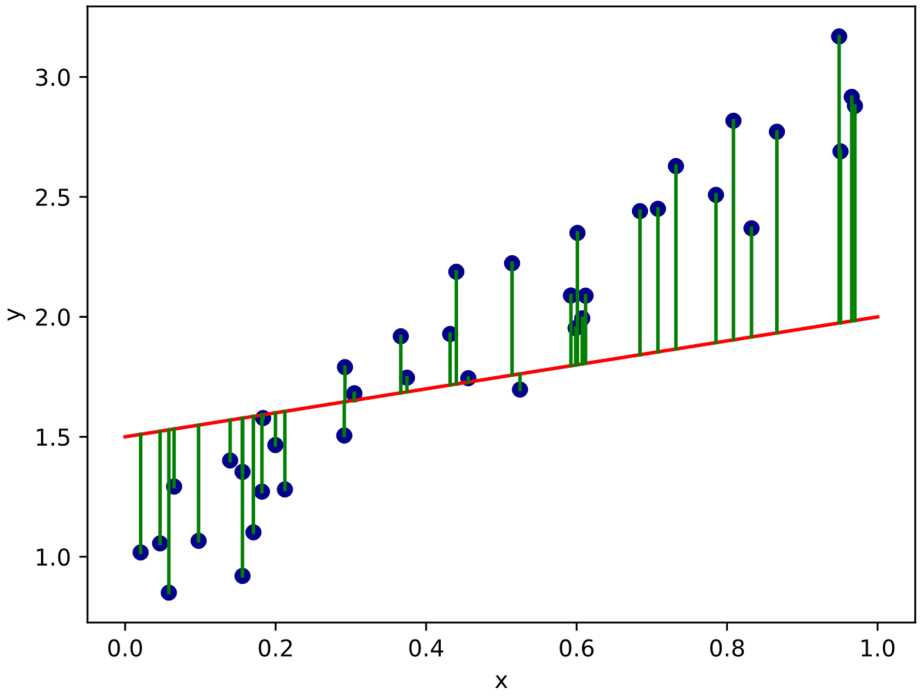
Motivating example



- Try fitting a straight line through the points
- Is this a good model of the data?

Linear Regression

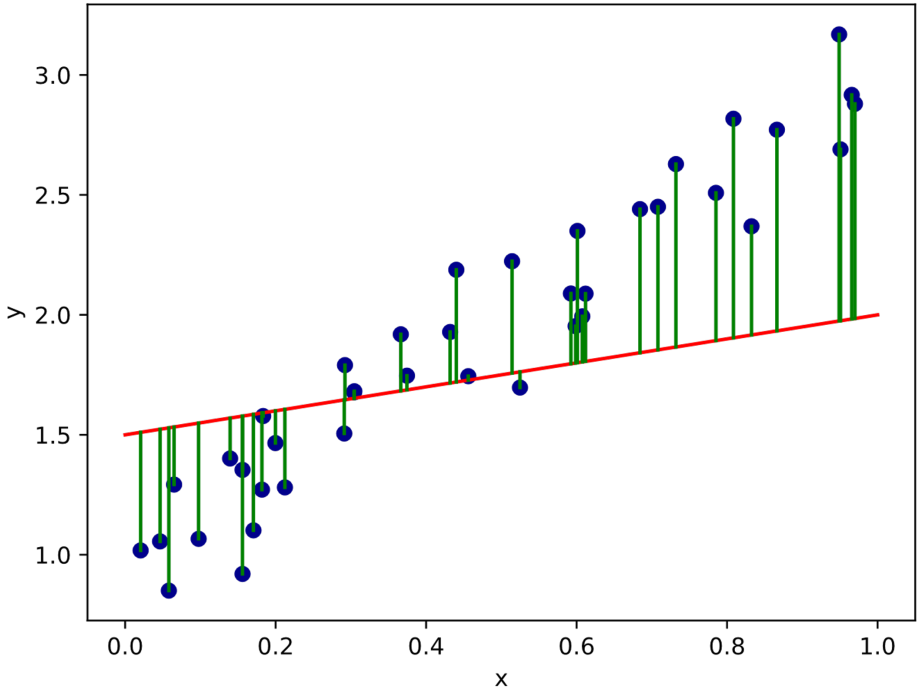
Motivating example



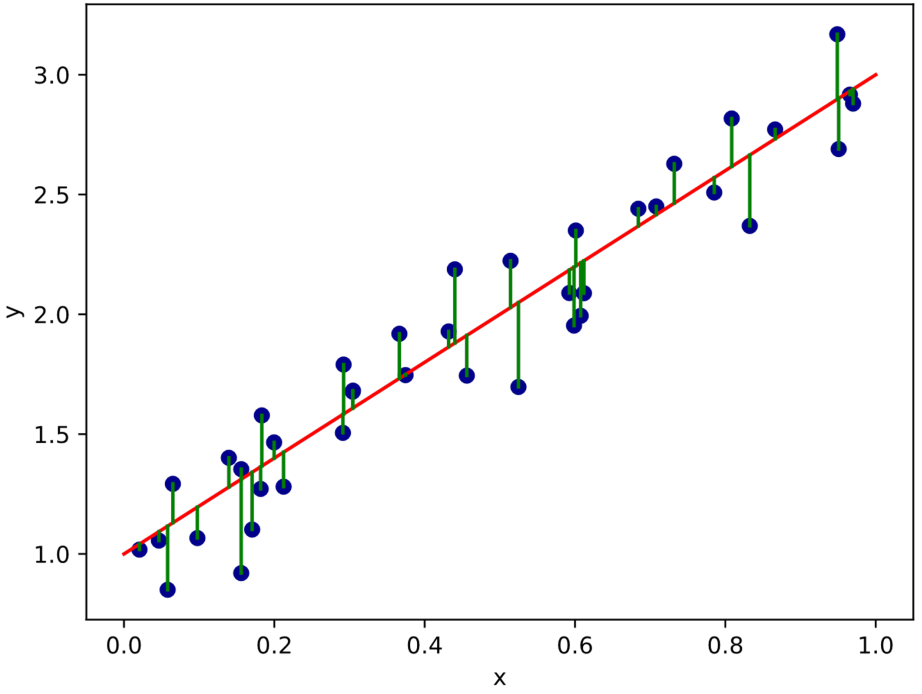
- There are large differences between the line (predicted $\hat{y}$) and the actual y
- NOT a good model of the data

Linear Regression

Motivating example



- There are large differences between the line (predicted $\hat{y}$) and the actual y
- NOT a good model of the data



- A better model of the data

Linear Regression

What do we need

- A way of expressing *linear functions*
- A way to measure *how well the line fits the data* (called a loss function)
- A way to find the line with the *smallest loss* given the data (through optimisation)

Linear Regression

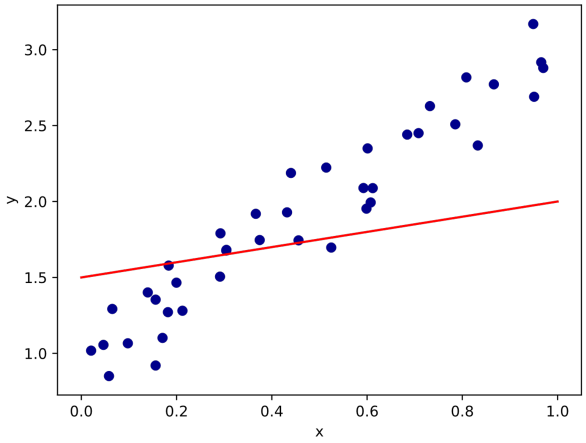
Expressing linear functions

For *one feature* we have

$$\hat{y} = wx + b$$

where

- w is the slope of the line (parameter)
- x is the feature (input)
- b is the bias (parameter)
- $\hat{y}$ is the prediction of the target variable (output)



For a linear function with *multiple features*

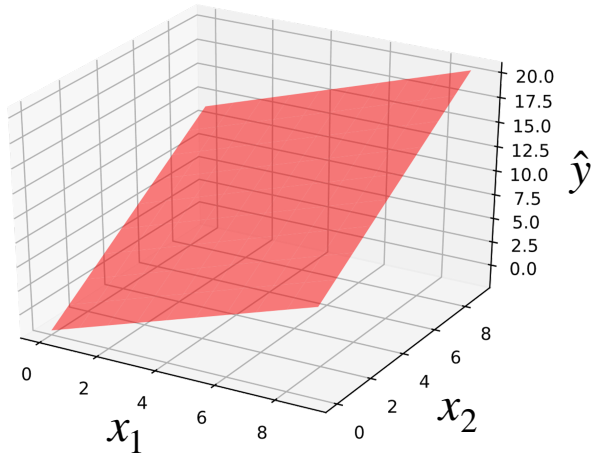
$$\hat{y} = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

Written more compactly for d features

$$\hat{y} = b + \sum_{i=1}^{d} w_i x_i$$

Written in vector form with vectors $\mathbf{w}$ and $\mathbf{x}$

$$\hat{y} = b + \mathbf{w}^\top \mathbf{x}$$



Linear Regression

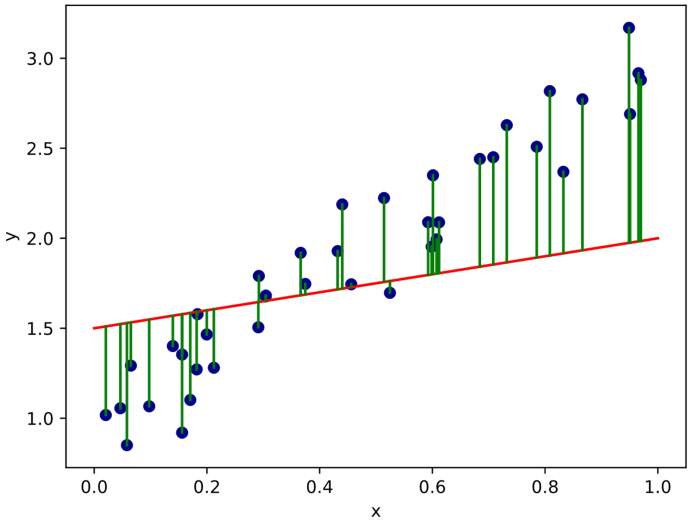
Loss function

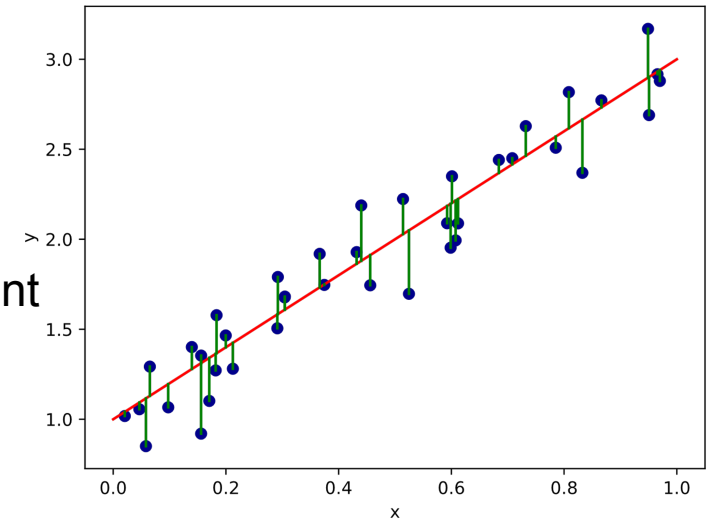
- A loss function measures how well the model fits the data points
- **Quadratic loss** (**squared error loss**): The squared difference between the prediction and true target for a data point

$$L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w}, b)$$
$$= (\hat{y}^{(i)} - y^{(i)})^2$$
$$= (\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)})^2$$

where

- $\mathbf{w}$ and b are the model parameters
- $\mathbf{x}^{(i)}$ is the feature vector of the i-th data point
- $y^{(i)}$ is the ground truth target of the i-th data point
- $\hat{y}^{(i)} = \mathbf{w}^\top \mathbf{x}^{(i)} + b$ is the predicted target for the i-th data point





Linear Regression

Optimisation

- Cost function for linear regression: Mean squared error (MSE)

$$J(\mathbf{w}, b) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}^{(i)}, y^{(i)}, \mathbf{w}, b)$$

$$= \frac{1}{N}\sum_{i=1}^{N} (\mathbf{w}^\top \mathbf{x}^{(i)} + b - y^{(i)})^2$$

  where N is the number the data points in the training set.

- To find the best linear function that models the data, solve this optimisation problem:

$$\underset{\mathbf{w}, b}{\text{argmin}} \; J(\mathbf{w}, b)$$

- There are many different algorithms for solving optimisation problems, **gradient descent** is a popular one when everything is differentiable.

Linear Regression

Gradient descent

- A greedy strategy for optimisation
- Initialisation: picking random values for $\mathbf{w}$ and b, e.g. by sampling from the standard normal distribution $\mathcal{N}(0, 1)$
- Compute the gradients

$$\frac{\mathrm{d}\, J(\mathbf{w}, b)}{\mathrm{d}\, \mathbf{w}} = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, \cdots, \frac{\partial J}{\partial w_d}\right] \text{ and } \frac{\mathrm{d}\, J(\mathbf{w}, b)}{\mathrm{d}\, b} = \frac{\partial J}{\partial b}$$

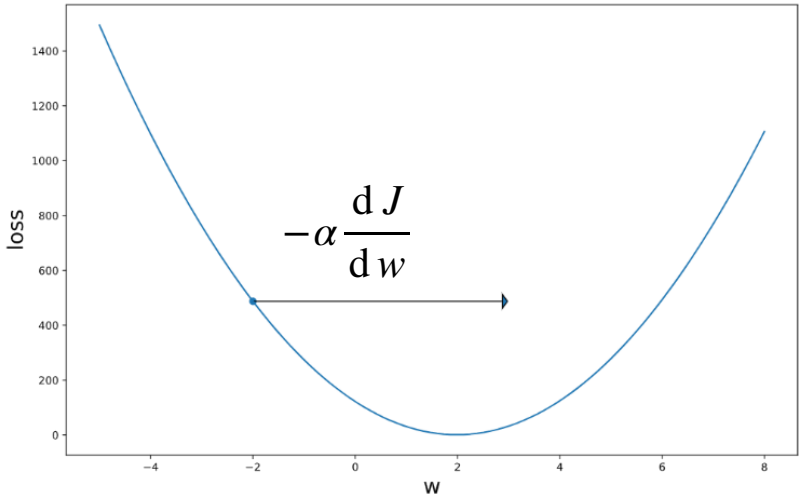
- Update

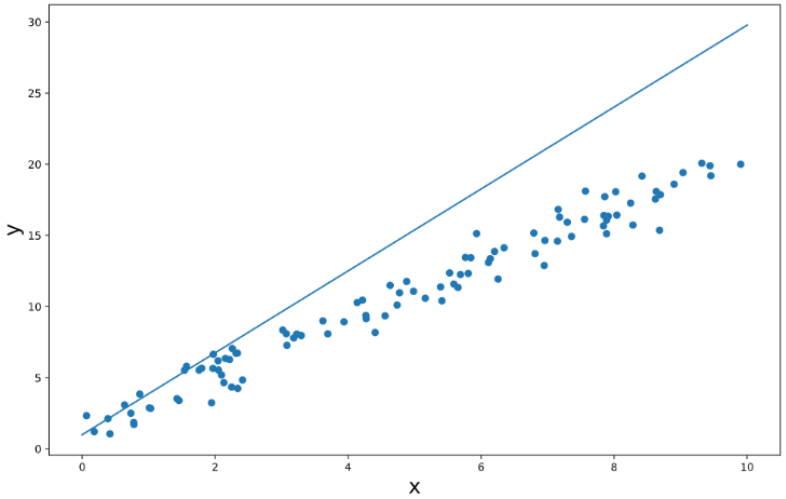
$$\mathbf{w} := \mathbf{w} - \alpha \frac{\mathrm{d}\, J}{\mathrm{d}\, \mathbf{w}} \text{ and } b := b - \alpha \frac{\mathrm{d}\, J}{\mathrm{d}\, b}$$

- Keep doing the updates until $\mathbf{w}$ and b stop changing: got a (locally) optimal solution
- Learning rate α controls how much the values change at each step
  - α is a hyper-parameter of the model: up to you to choose a good value
  - Too small α: need to take many steps
  - Too large α: may not converge

Linear Regression

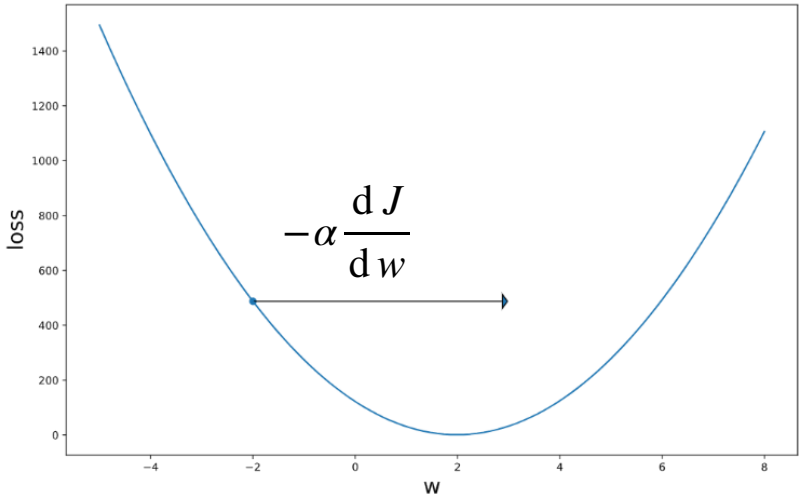
Gradient descent example: one parameter

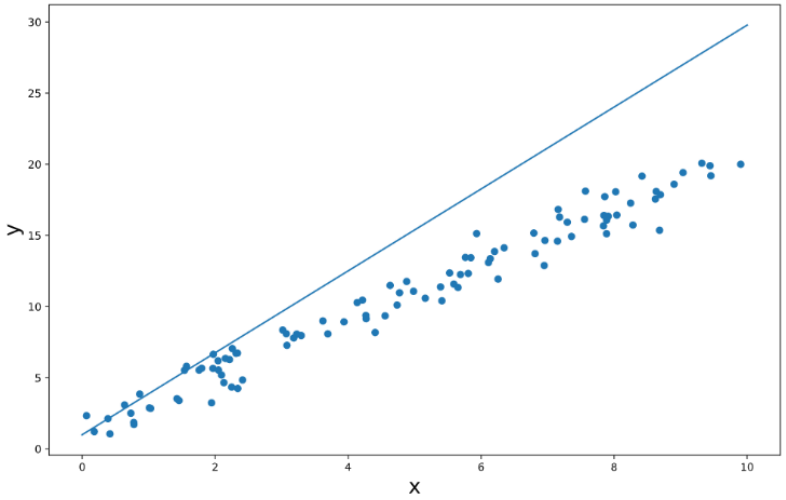


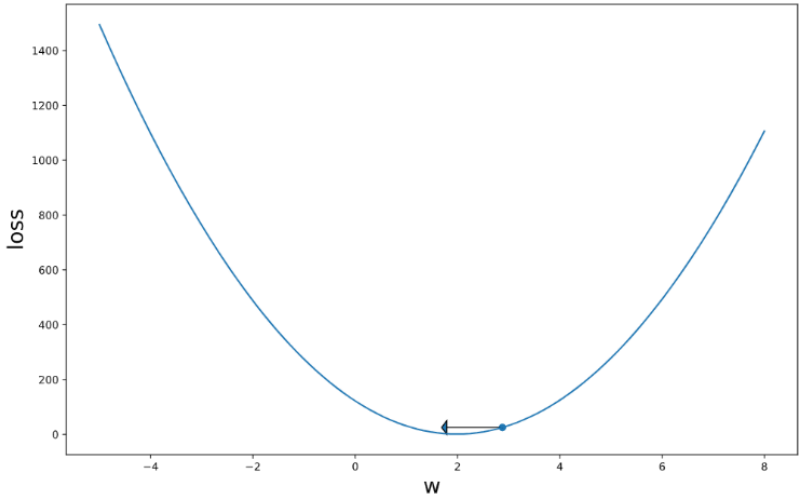


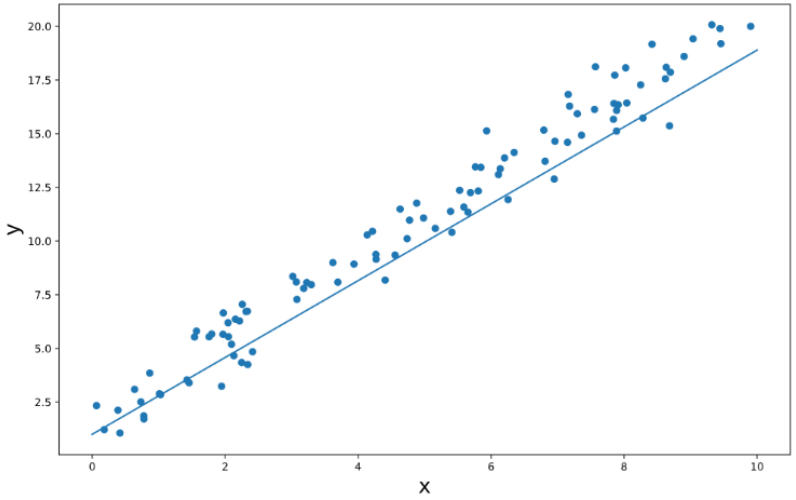
Linear Regression

Gradient descent example: one parameter



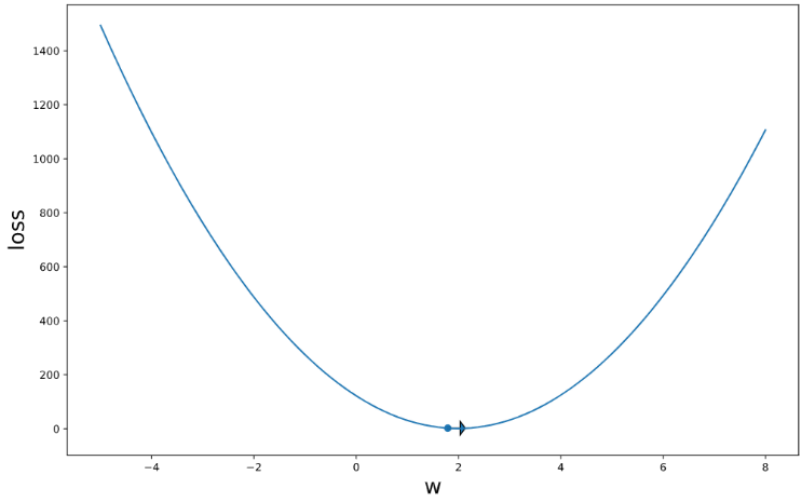


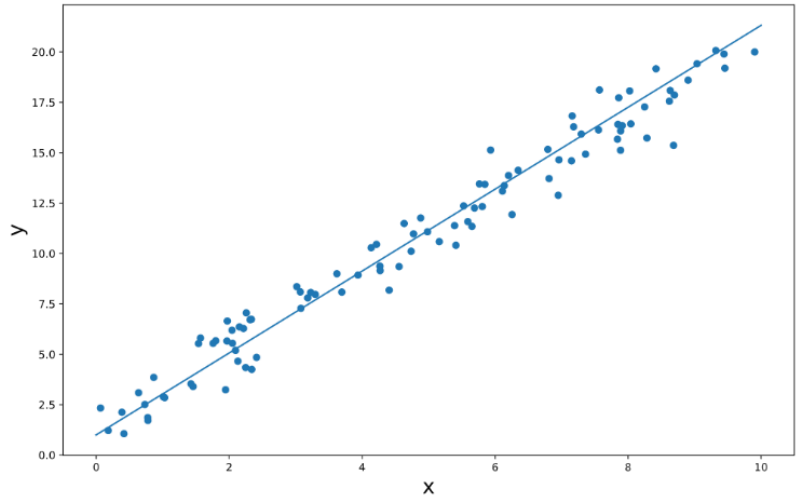


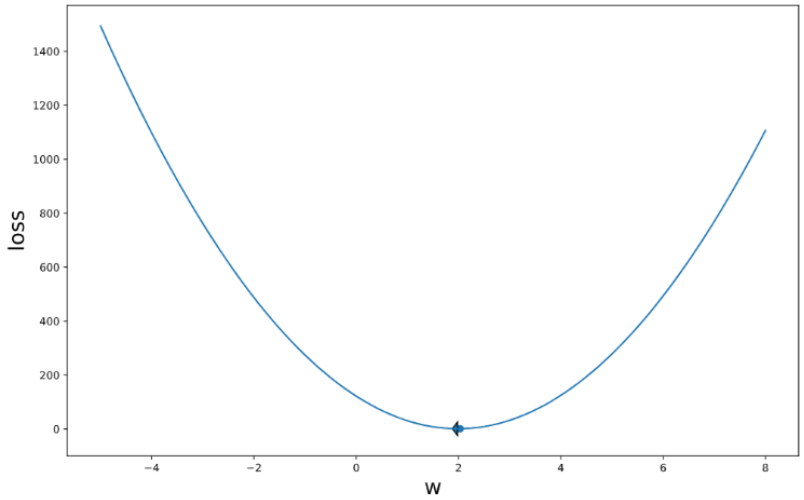


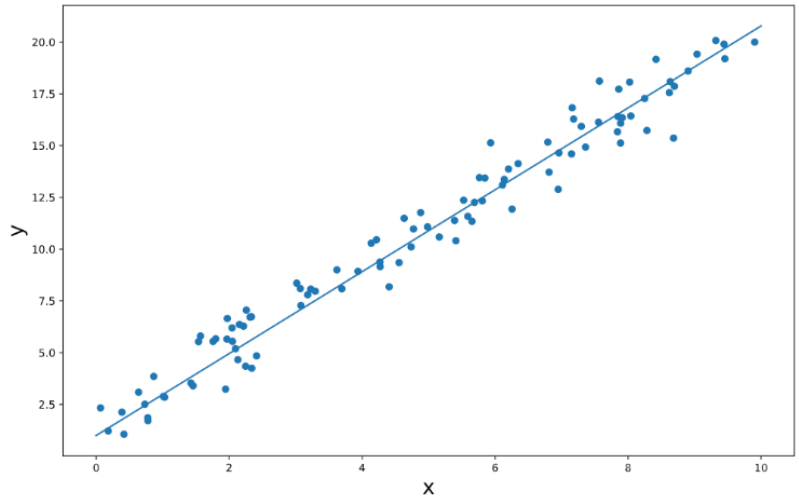
Linear Regression

Gradient descent example: one parameter



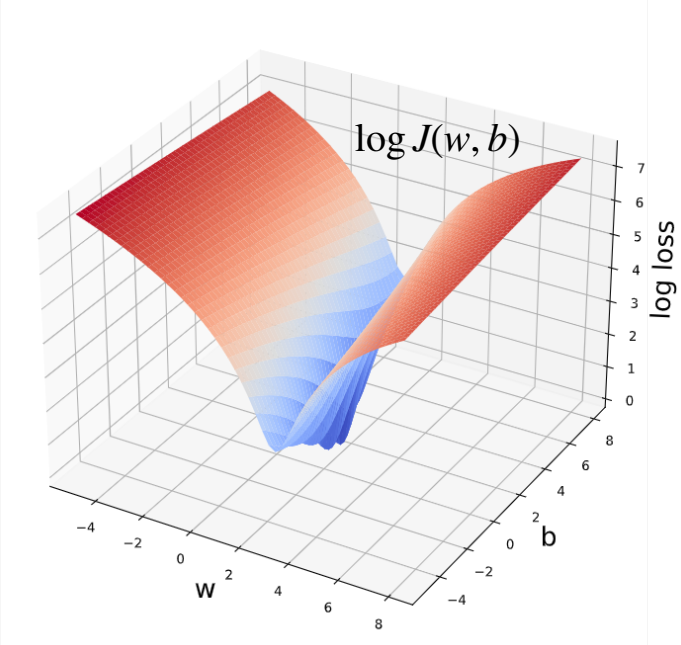


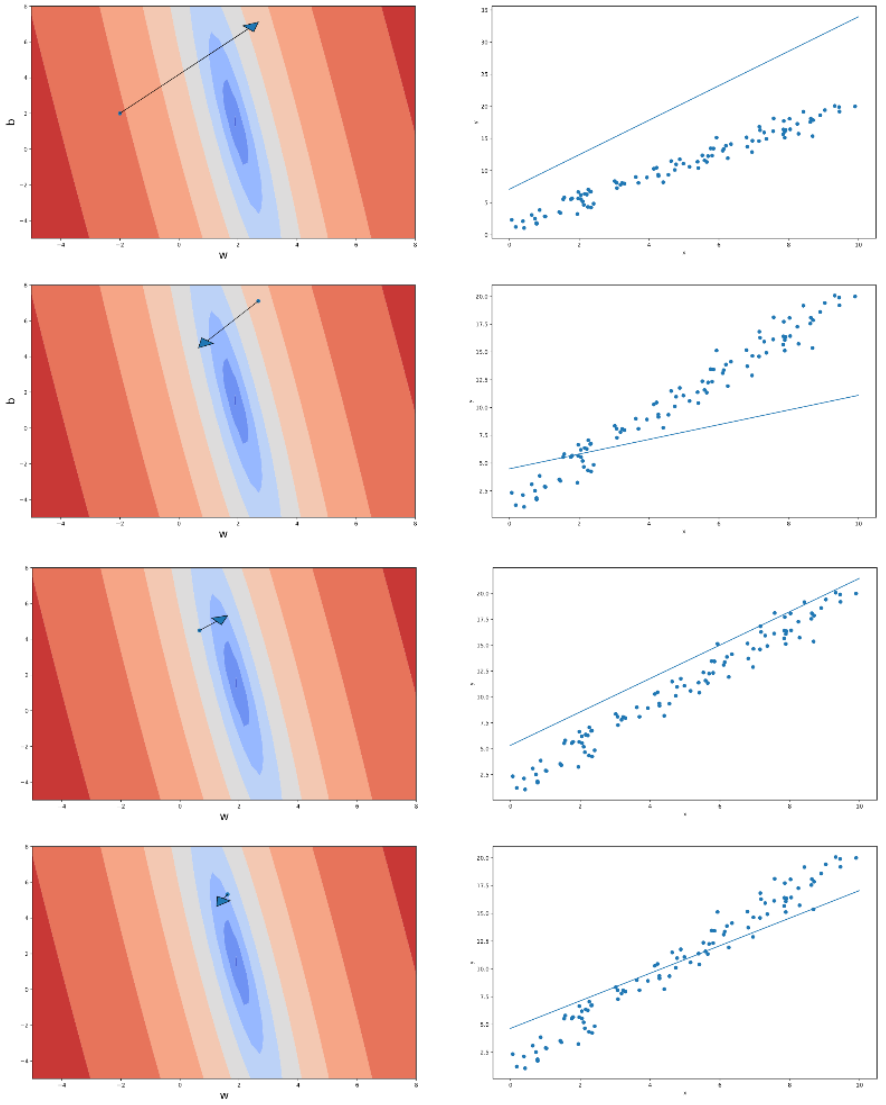




Linear Regression

Gradient descent example: two parameters





Linear Regression

Multiple linear regression

If we have more than one target **per example,** then we can use a different set of parameters for each target:

$$\hat{y}_1 = w_{1,1}x_1 + w_{1,2}x_2 + \cdots + w_{1,d}x_d + b_1$$
$$\hat{y}_2 = w_{2,1}x_1 + w_{2,2}x_2 + \cdots + w_{2,d}x_d + b_2$$
$$\hat{y}_3 = w_{3,1}x_1 + w_{3,2}x_2 + \cdots + w_{3,d}x_d + b_3$$
$$\cdots$$
$$\hat{y}_m = w_{m,1}x_1 + w_{m,2}x_2 + \cdots + w_{m,d}x_d + b_m$$

In matrix / vector form (for matrix W, and vectors $\hat{\mathbf{y}}$, $\mathbf{x}$, $\mathbf{b}$):

$$\hat{\mathbf{y}} = W\mathbf{x} + \mathbf{b}$$

Australian National University

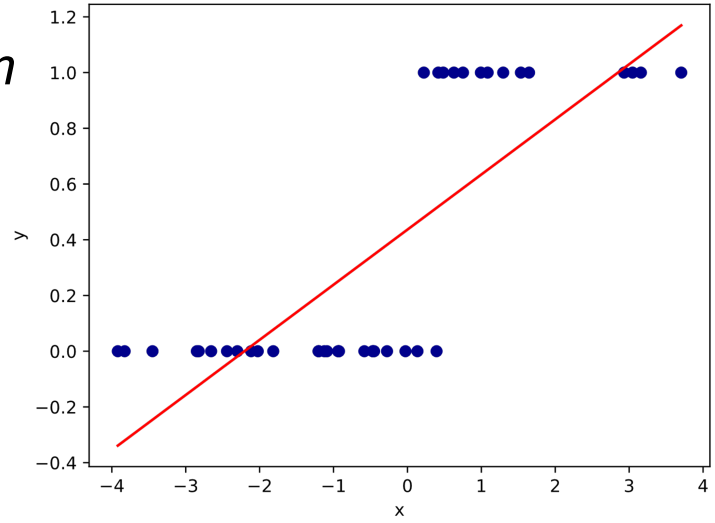
Outline

- Machine Learning Overview
  - Motivation, terminology, and notation
- Linear Regression
  - Motivating example
  - Expressing linear functions
  - Loss function for linear regression
  - Optimisation, gradient descent
  - Multiple linear regression
- Logistic Regression
  - Classification problem
  - The softmax function
  - Multinomial logistic regression, cross entropy loss

Logistic Regression

Classification

- What if the target to predict is **discrete**? e.g. classifying emails as *spam* vs *not spam*
- This is not a regression problem, so our linear model needs some changes
- To use a linear model $W\mathbf{x} + \mathbf{b}$ (multiple linear regression), we can modify the task to predict the **probability** that an example belongs to each class



- Classification example with 2 classes ($y \in \{0, 1\}$)
- Linear regression does not make sense for this task

x	$P(y = 0 \mid x)$	$P(y = 1 \mid x)$
-3.34	0.9999	0.0001
0.01	0.55	0.45
1.00	0.021	0.979
….	….	….

Example of predicting probabilities

Logistic Regression

Classification

- Suppose that each data point can be labelled with one of m different classes
- Linear model $\bar{P}(\mathbf{y} \mid \mathbf{x}) = W\mathbf{x} + \mathbf{b}$ outputs a vector of size m
- $\bar{P}(\mathbf{y} \mid \mathbf{x})$ denotes un-normalised probabilities (often called logits) which may be negative

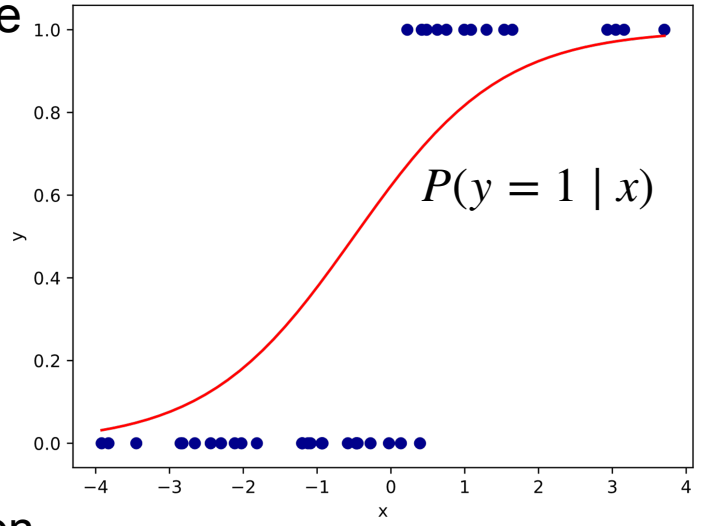
x	$\bar{P}(y = 0 \mid x)$	$\bar{P}(y = 1 \mid x)$
-3.34	13.56	3.1
0.01	12.1	11.9
1.00	-1.7	2.15
….	….	….

Example of un-normalised probabilities

Logistic Regression

Classification: from logits to probabilities

- The output values of the linear model are not probabilities:
  - Need all the output values to be non-negative
  - Need output values to sum up to 1



- Converting logits to probabilities using the **softmax** function

$$\text{softmax}(\mathbf{v})_i = \frac{\exp(v_i)}{\sum_{j=1}^{m} \exp(v_j)}$$

  - Step 1: Apply the (standard) exponential function to all values. This makes them positive.
  - Step 2: Divide each value by the sum of all values. This makes them sum to 1.
  - The result is a categorical probability distribution.
  - The softmax function is a generalisation of the logistic / sigmoid function $\sigma(v) = (1 + \exp(-v))^{-1}$ to multiple dimensions

Logistic Regression

Multinomial logistic regression

- Predicted probabilities:

$$P(\mathbf{y} \mid \mathbf{x}) = \text{softmax}(W\mathbf{x} + \mathbf{b})$$

- Cross-entropy loss (per example)

$$L_{CE}(\mathbf{x}, \mathbf{y}, W, b) = -\sum_{j=1}^{m} y_j \log P(y_j \mid \mathbf{x})$$

x_1	x_2	y	$\mathbf{y}$ (one-hot encoding)
1.24	-0.32	2	$[0, 0, 1]^\top$
-0.11	2.64	0	$[1, 0, 0]^\top$
3.17	1.55	1	$[0, 1, 0]^\top$

Example of one-hot encoding of the label. Suppose that data points are labelled with one of three possible values {0, 1, 2}.

  - $\mathbf{y}$ is the one-hot encoding of the label
  - Minimising the cross-entropy loss is the same as maximising the (log) likelihood
  - We could use the sum of squared differences as the loss function, just like for regression. But in practice, using cross-entropy as the loss function for classification gives better results.

Logistic Regression

Multinomial logistic regression

- Predicted probabilities:

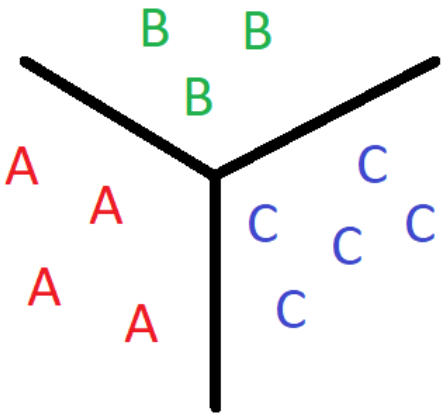
$$P(\mathbf{y} \mid \mathbf{x}) = \text{softmax}(W\mathbf{x} + \mathbf{b})$$

- Parameters (W and $\mathbf{b}$) can be learned by minimising the cost function $J(W, \mathbf{b})$ w.r.t. W and $\mathbf{b}$, e.g. using gradient descent

$$J(W, \mathbf{b}) = \frac{1}{N} \sum_{i=1}^{N} L_{CE}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}, W, \mathbf{b})$$

- To make a prediction for a new data point:
  - using the model to compute probabilities for each class
  - predict the class with the *largest* probability
  - When you predict a class in this way the decision boundarie are straight lines

  Why?
  (*Hint: Think about how the probability is calculated*)



Summary

- Machine Learning Overview
  - Motivation, terminology, and notation
- Linear Regression
  - Motivating example
  - Expressing linear functions
  - Loss function for linear regression
  - Optimisation, gradient descent
  - Multiple linear regression
- Logistic Regression
  - Classification problem
  - The softmax function
  - Multinomial logistic regression, cross entropy loss

References

- Chapter 5, Speech and Language Processing