



# Introduction

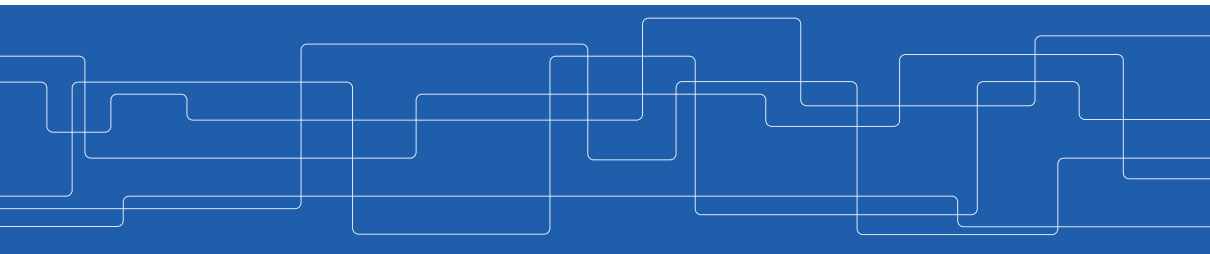
Jim Dowling

[jdowling@kth.se](mailto:jdowling@kth.se)

Course Assistants: Fabien Schmidt, Tianze Wang

2022-10-31

Slides by Amir H. Payberah and Jim Dowling





# Course Information



## Course Objective

- ▶ This course has a **system-based** focus.
- ▶ Learn the **theory** of **machine learning and deep learning**.
- ▶ Learn the **practical aspects** of building **machine learning and deep learning** algorithms using data **parallel programming platforms**, such as **TensorFlow**.





# Topics Covered in the Course

- ▶ **Part 1: large scale deep learning**
  - TensorFlow
  - Deep Neural Networks (DNN)
  - Different DNN architectures, e.g., CNNs, RNNs, Autoencoders, GAN
  - Distributed learning
  
- ▶ **Part 2: large scale machine learning**
  - Serverless Machine Learning
  - MLOps (machine learning operations)
  - Feature Stores for Machine Learning
  - Distributed Training of DNNs



## Intended Learning Outcomes (ILOs)

- ▶ **ILO1:** explain the **principles** of **ML/DL algorithms** and apply their techniques to solve problems.
- ▶ **ILO2:** demonstrate an ability to design **DNN architectures** and explain challenges to scaling their training and inference with increasing compute and data.
- ▶ **ILO3:** explain the **principles** of **scaling machine learning systems**.
- ▶ **ILO4:** implement **scalable ML/DL systems**.



# The Course Assessment

- ▶ **Task1**: the **Examination** on 16th January 2023(A-F)
- ▶ **Task2**: the **lab** assignments (A-F)
- ▶ **Task3**: the final **project** (P/F)



## How Each ILO is Assessed?

	Task1	Task2	Task3
ILO1	x		
ILO2	x	x	x
ILO3	x	x	
ILO4		x	x





## Task1: The Exam (A-F)

- ▶ Questions about the [lectures](#) and the labs.
- ▶ Some sample questions for the exam will be distributed in mid December.
- ▶ The examination is [graded \(A-F\)](#).



## Task2: The Lab Assignments (A-F)

- ▶ Two lab assignments: **source code** and **oral presentation**.
- ▶ E: source code
- ▶ D: source code + half questions (basic)
- ▶ C: source code + all questions (basic)
- ▶ B: source code + half questions (basic and advanced)
- ▶ A: source code + all questions (basic and advanced)



## Task3: The Final Project (A-F)

- ▶ One final project: **source code** and **oral presentation**.
- ▶ Proposed by students and confirmed by the teacher:
- ▶ Source code and documentation as a **README.md** by the students of the project
- ▶ 5 minute Presentation of the project by the students.
- ▶ 20% of total grade bonus for projects graded as excellent.



## The Final Grade

- ▶ The **final grade** is the **weighted average** of the **Exam** (0.3), **two labs** (0.15 each). A bonus of .09 points will be added for projects graded 'excellent'.
- ▶ To compute it, map **A-E to 5-1**, and take the average.
- ▶ The floating values are **rounded up**, if they are **more than half**, otherwise they are **rounded down**.
  - E.g., **3.6** will be rounded to **4**, and **4.5** will be rounded to **4**.
- ▶ A **late** submission will **reduce your grade level by one**. That is, A will become B, B will become C, and so on.
- ▶ To pass the course, you need to take **at least E** in **all the assignments**.

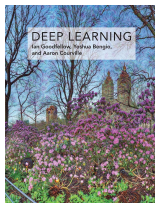
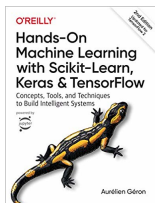
# How do you submit the Assignments?

- ▶ Through the [Canvas](#) site.
- ▶ Students will work in **groups of two** on [Task 2](#) and [Task 3](#).



# The Course Material

- ▶ **Hands-on machine learning with Scikit-Learn and TensorFlow, 2nd Edition**, A. Geron, O'Reilly Media, 2019
- ▶ **Deep learning**, I. Goodfellow et al., Cambridge: MIT press, 2016





## The Course Web Page

<https://id2223kth.github.io>



# Supervised Machine Learning 101



## Features in Machine Learning

**Features are properties of things** you want to use to make predictions  
*For example, predict if a fruit is either an apple or an orange based on its color/weight.*



**feature** color: green  
**feature** weight: 70-250gr



**feature** color: orange  
**feature** weight: 60-300gr

# Supervised Machine Learning (ML)

Each example has both the features (the fruit's color) and a label that is either "apple" or "orange". The **label** is the 'target' we are trying to predict, using the input **features** (the fruit's color).



Examples of Apples



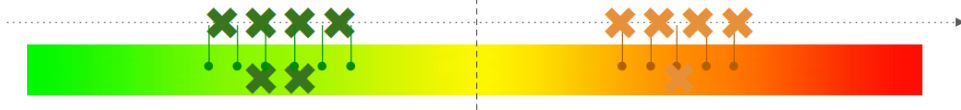
Example of Oranges



# A Linear Model can classify the Fruit (Classifier)

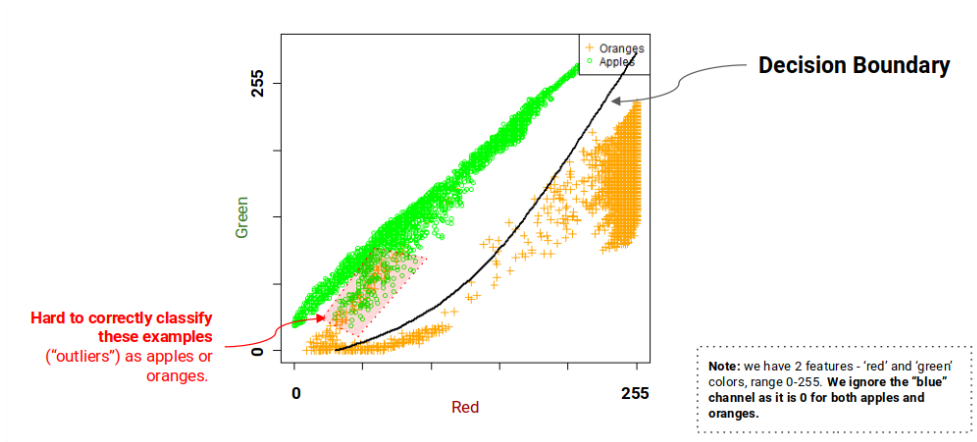
## Decision Boundary

$$\text{RGB}(0,128,0) < \text{RGB}(255,165,0)$$



## The Decision Boundary

For a large enough data set, even with a decision tree classifier and 2 features, **it's still not 100% accurate**





## Source code for a small sample of data with a decision tree classifier

```
import sklearn
from sklearn import tree
# 4 examples of features with [red-color, green-color]
features = [[0,120], [0, 110], [250, 150], [255, 163]]
# green apples == 0; oranges == 1
labels = [0, 0, 1, 1]

clf = tree.DecisionTreeClassifier()
clf = clf.fit(features, labels)

test_fruits = [[0,128], [249, 155]]
test_labels = [0, 1]
pred_labels = clf.predict(test_fruits)
print(pred_labels)
```

## But wait, apples can also be red!

There is no straight-line decision boundary based on 2 color channels (red and green)

RGB(0,128,0)



RGB(255,165,0)

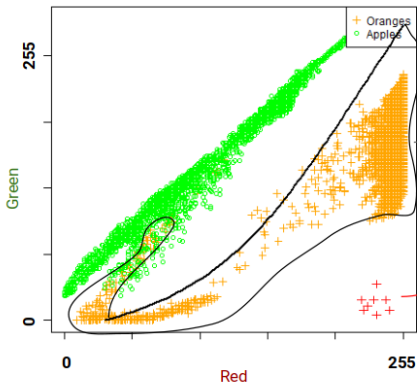


RGB(255,0,0)



It's harder to separate Green Apples, Oranges, and Red Apples with just 2 colors (red and green)

**We need a non-linear classifier** to learn to separate apples and oranges based on our 2 color channels (we don't need the blue channel)



Good decision boundary, but it's harder to learn

Red apples

**Note:** we have 2 features - 'red' and 'green' colors; 'blue' color is the same (0) for both apples and oranges.

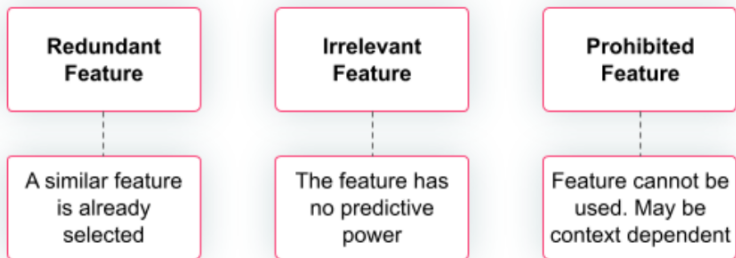


## From two dimensions to millions of dimensions

- ▶ Can we just add more features? Yes. If you add weight and smoothness, you can separate apples and oranges. We could plot our fruit in 3d or even 4d and find a plane that separates apples and oranges
- ▶ You can add many more features (dimensions). With the caveat that too many dimensions can lead to overfitting. Overfitting means your model is not good at generalizing to correctly predict new fruit examples (it would work well for the training data, but not unseen (new) examples)
- ▶ In image classification, each pixel is a feature. That's millions of features for a single HD image. Deep learning can be used to train models with millions of features.



## Not all properties with predictive power should be features



With [current inadequate AI legislation/norms](#), you are personally responsible to build [ethical AI](#) systems

# Lots of features: deep learning for apple classification works

[Ayaz et al](#) show how deep learning and image classifiers **can identify rotten/blotched/scab apples**, shown here.

Deep learning models as **non-linear classifiers**

To work well, they need **lots of labelled training data** and **GPUs** to train models on

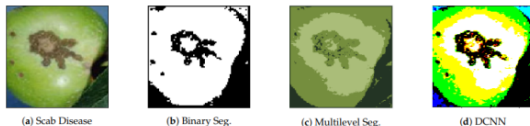


Figure 6. Scab apple process through binary thresholding, clustering and DCNN.

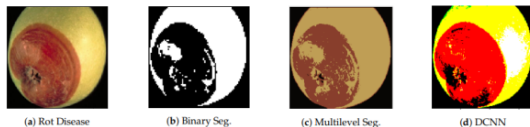


Figure 7. Rot apple process through binary thresholding, clustering and DCNN.

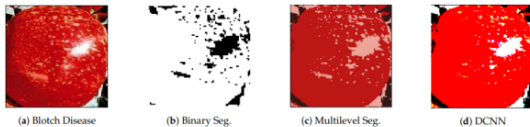


Figure 8. Blotch apple process through binary thresholding, clustering and DCNN.

# A model can also be trained to predict a number (regression)

## **A regression problem:**

train an ML model to estimate the weight of an apple given its dimensions (diameter) and color.



(red, 9cm) -> 120gr



(dark green, 11cm) -> 180gr

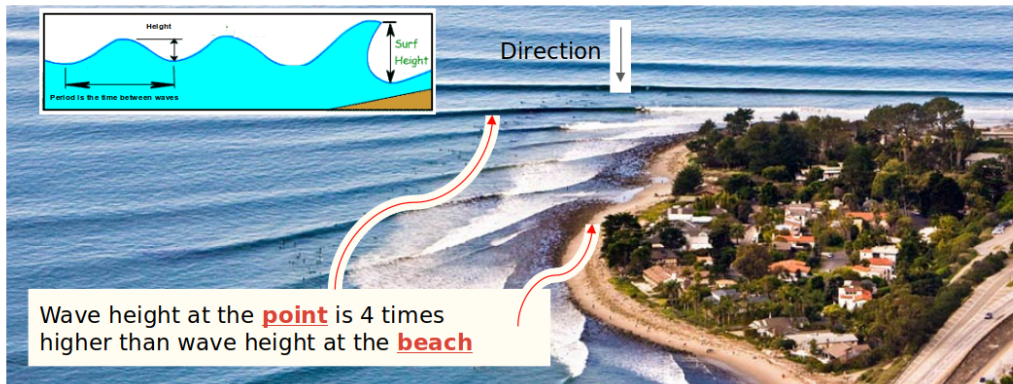


(light green, 13cm) -> 160gr

# Predicting Surf Height at a beach – Classification or Regression?

## Classification or regression?

Problem: predict the height of surf at a beach (see example system on the course website), three features that are useful are: (**swell height, period, direction**) -> **Surf Height in feet**





## What is Supervised Machine Learning, then?

- ▶ With our Apple/Orange classifier, we used (features, label) examples to train a model to find a decision boundary.
- ▶ Then when a new fruit arrived, we could use the model to predict if the new piece of fruit is an apple or an orange.
- ▶ We can generalize to say that supervised machine learning is concerned with:
  - ▶ extracting a pattern from labeled data (features) to a model
  - ▶ using that model to make predictions for new unlabeled data (features)



## Traditional ML courses vs ID2223

- ▶ Static Datasets, where Features for ML are correct and unbiased
  - ▶ The goal is to optimize your model with a model evaluation metric (accuracy) to communicate the value of your model
- 
- ▶ Data never stops coming and it comes from heterogeneous data sources
  - ▶ Communicate the value of your model as a Prediction Service - one that can be scaled and deployed using MLOps (Machine Learning Operations) best practices (versioning, automated testing/deployment)

# Feature Engineering is often treated like this “helpful” guide to drawing a Barn Owl

## How to Draw an Owl:



1. Draw Circles



2. Now, “simply” draw the rest of the Owl

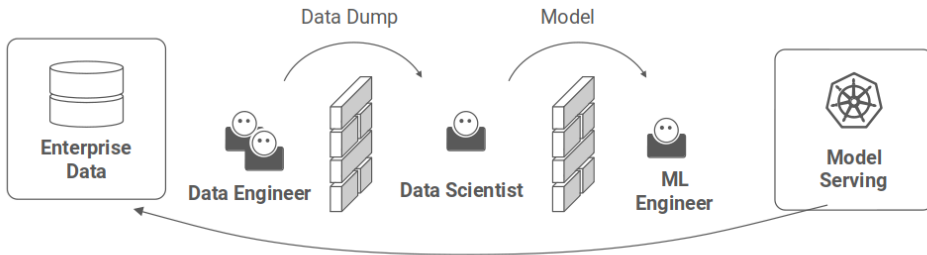


Extract the features from the input data. We will study feature engineering at scale.

<b>Raw Data</b>	<b>Extracted Feature</b>	<b>Method</b>
Hotel room bookings	Weekly vacancy level	Aggregation
User's web session history	Session history embedding	Dimensionality Reduction
User's date of birth		Binning
Hourly spot electricity prices	Scale into range $[0, 1]$	Normalization
User's home country	Binary number of country	One Hot Encoding

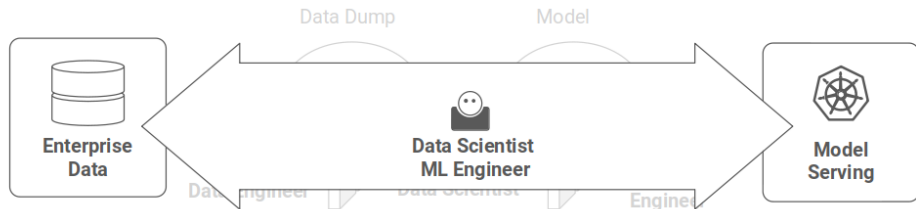


# Many Enterprises have walled gardens between teams building production ML Systems



In many large organizations,  
data scientists only build and evaluate models

With infrastructure support (Serverless Machine Learning), developers need less infrastructural skills to deploy ML Systems



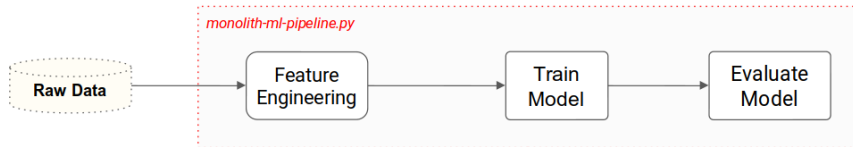
*Don't stop at training a model, build its prediction service*

Learn to deploy your models, evaluate, and debug them.

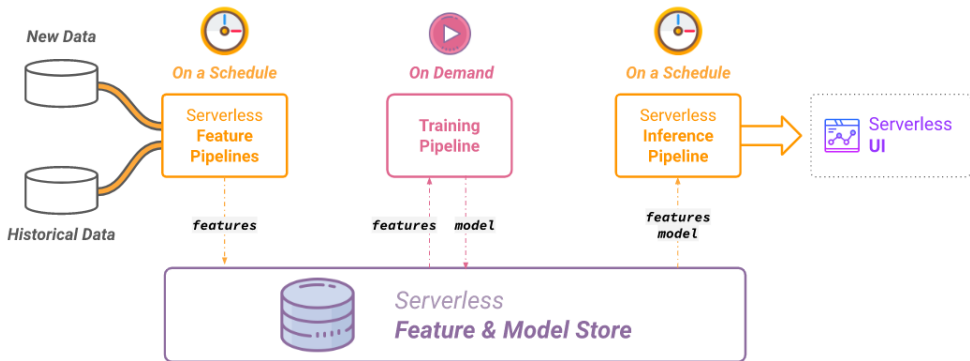
Leverage **Serverless Machine Learning** to avoid installing and managing the infrastructure needed to machine learning in production.

# Monolithic ML Pipeline

- A pipeline is a program that takes an input and produces an output
- End-to-end ML Pipelines are a single pipeline that transforms raw data into features and trains and scores the model in one single program



# Refactor the Monolithic ML Pipeline to Scale your ML Systems





# Machine Learning and Deep Learning

# Learning Algorithms

- ▶ A **ML algorithm** is an algorithm that is able to **learn from data**.
- ▶ What is **learning**?
- ▶ 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 M. Mitchell)



# Learning Algorithms - Example 1

- ▶ A **spam filter** that can learn to flag **spam** given examples of **spam emails** and examples of **regular emails**.
- ▶ **Task T**: flag spam for new emails
- ▶ **Experience E**: the training data
- ▶ **Performance measure P**: the ratio of correctly classified emails



[<https://bit.ly/2oip1YM>]

## Learning Algorithms - Example 2

- ▶ Given dataset of prices of 500 houses, how can we learn to **predict the prices** of other houses, as a **function of the size of their living areas**?
- ▶ **Task T**: predict the price
- ▶ **Experience E**: the dataset of living areas and prices
- ▶ **Performance measure P**: the difference between the predicted price and the real price



[<https://bit.ly/2MyiJUy>]



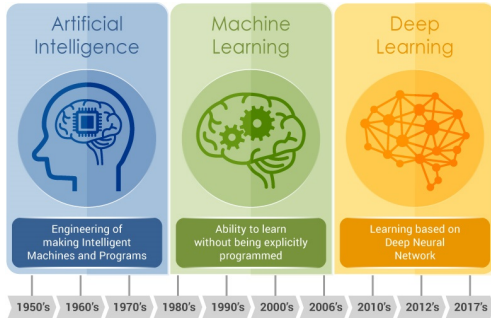
# Types of Machine Learning Algorithms

- ▶ **Supervised learning**
  - Input data is **labeled**, e.g., spam/not-spam or a stock price at a time.
  - **Regression vs. classification**
  
- ▶ **Unsupervised learning**
  - Input data is **unlabeled**.
  - Find **hidden structures** in data.



# AI Generations - Deep Learning

- ▶ For many tasks, it is difficult to know what features should be extracted
- ▶ Use machine learning to discover the mapping from representation to output



[<https://bit.ly/2woLEzs>]



# Image Classification with Deep Learning

- ▶ For image classification, where each pixel is a feature, Deep Learning can do the feature extraction as part of the learning algorithm.

# Sheepdog or Mop



# Chihuahua or Muffin

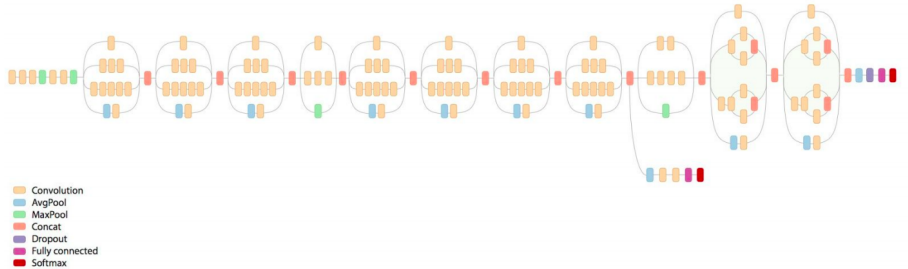


# Barn Owl or Apple



# Training Deep Neural Networks

- ▶ Computationally intensive
- ▶ Time consuming



[<https://cloud.google.com/tpu/docs/images/inceptionv3onc--oview.png>]

# Why?

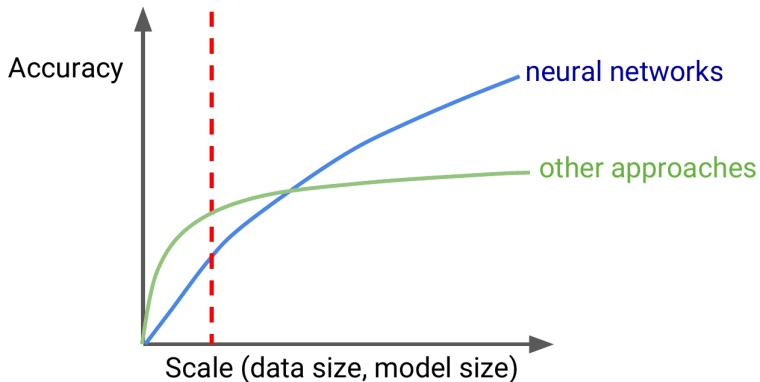
- ▶ Massive amount of training dataset
- ▶ Large number of parameters





# Accuracy vs. Data/Model Size

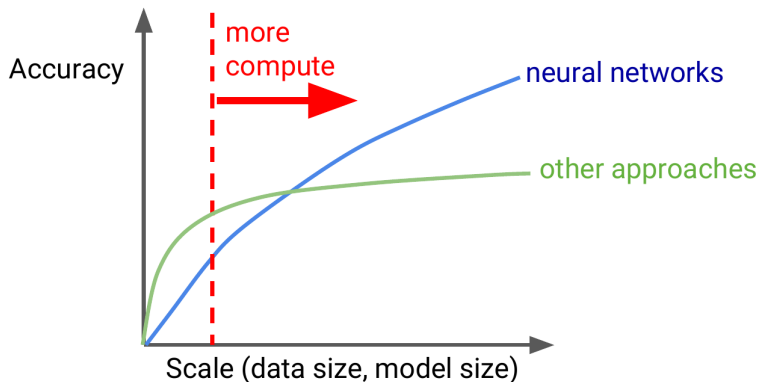
## 1980s and 1990s



[Jeff Dean at AI Frontiers: Trends and Developments in Deep Learning Research]

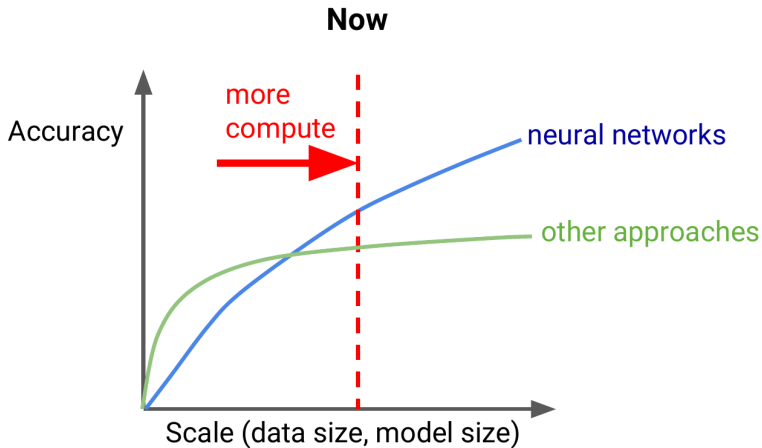
# Accuracy vs. Data/Model Size

## 1980s and 1990s



[Jeff Dean at AI Frontiers: Trends and Developments in Deep Learning Research]

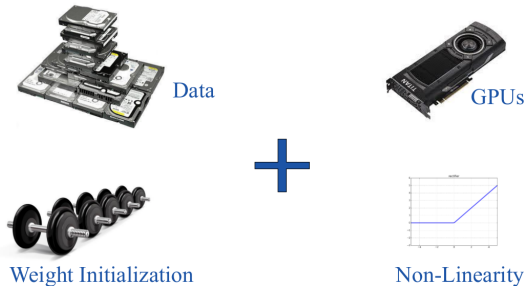
# Accuracy vs. Data/Model Size



[Jeff Dean at AI Frontiers: Trends and Developments in Deep Learning Research]

# Why Does Deep Learning Work Now?

- ▶ Huge quantity of data
- ▶ Tremendous increase in computing power
- ▶ Better training algorithms





# Linear Algebra Review



# Vector

- ▶ A **vector** is an **array of numbers**.
- ▶ **Notation:**
  - Denoted by **bold lowercase letters**, e.g.,  **$\mathbf{x}$** .
  - $\mathbf{x}_i$  denotes the  **$i$ th entry**.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$



# Matrix and Tensor

- ▶ A **matrix** is a 2-D array of numbers.
- ▶ A **tensor** is an array with more than two axes.
- ▶ **Notation:**
  - Denoted by **bold uppercase letters**, e.g., **A**.
  - $a_{ij}$  denotes the entry in  $i$ th row and  $j$ th column.
  - If **A** is  $m \times n$ , it has  $m$  rows and  $n$  columns.

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \dots & a_{m,n} \end{bmatrix}$$



# Matrix Addition and Subtraction

- ▶ The **matrices** must have the **same dimensions**.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} + \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} a + e & b + f \\ c + g & d + h \end{bmatrix}$$



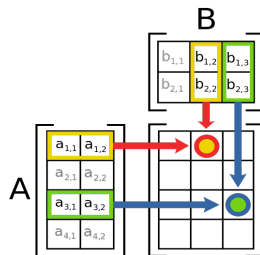
# Matrix Product

- ▶ The **matrix product** of matrices **A** and **B** is a third matrix **C**, where **C = AB**.
- ▶ If **A** is of shape  $m \times n$  and **B** is of shape  $n \times p$ , then **C** is of shape  $m \times p$ .

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

## ▶ Properties

- Associative: **(AB)C = A(BC)**
- Not commutative: **AB ≠ BA**



[[https://en.wikipedia.org/wiki/Matrix\\_multiplication](https://en.wikipedia.org/wiki/Matrix_multiplication)]



# Matrix Transpose

- ▶ Swap the **rows and columns** of a matrix.

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix} \Rightarrow \mathbf{A}^T = \begin{bmatrix} a & c & e \\ b & d & f \end{bmatrix}$$

- ▶ Properties

- $\mathbf{A}_{ij} = \mathbf{A}_{ji}^T$
- If  $\mathbf{A}$  is  $m \times n$ , then  $\mathbf{A}^T$  is  $n \times m$
- $(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T$
- $(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T$



## Inverse of a Matrix

- ▶ If  $\mathbf{A}$  is a square matrix, its inverse is called  $\mathbf{A}^{-1}$ .

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

- ▶ Where  $\mathbf{I}$ , the identity matrix, is a diagonal matrix with all 1's on the diagonal.

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



## $L^p$ Norm for Vectors

- ▶ We can measure the **size of vectors** using a **norm** function.
- ▶ Norms are functions **mapping vectors to non-negative values**.
- ▶  $L^1$  norm

$$\|\mathbf{x}\|_1 = \sum_i |x_i|$$

- ▶  $L^2$  norm

$$\|\mathbf{x}\|_2 = \left( \sum_i |x_i|^2 \right)^{\frac{1}{2}} = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$$

- ▶  $L^p$  norm

$$\|\mathbf{x}\|_p = \left( \sum_i |x_i|^p \right)^{\frac{1}{p}}$$



# Probability Review



# Random Variables

- ▶ **Random variable:** a **variable** that can take on **different values** randomly.
- ▶ Random variables may be **discrete** or **continuous**.
  - **Discrete** random variable: **finite or countably infinite** number of states
  - **Continuous** random variable: **real value**
- ▶ **Notation:**
  - Denoted by an **upper case letter**, e.g.,  $X$
  - Values of a random variable  $X$  are denoted by **lower case letters**, e.g.,  $x$  and  $y$ .



# Probability Distributions

- ▶ **Probability distribution**: how likely a random variable is to take on each of its possible states.
  - E.g., the random variable  $X$  denotes the outcome of a coin toss.
  - The probability distribution of  $X$  would take the value 0.5 for  $X = \text{head}$ , and 0.5 for  $Y = \text{tail}$  (assuming the coin is fair).
- ▶ The way we describe probability distributions depends on whether the variables are discrete or continuous.



# Discrete Variables

- ▶ **Probability mass function (PMF)**: the probability distribution of a discrete random variable  $X$ .
- ▶ **Notation**: denoted by a lowercase  $p$ .
  - E.g.,  $p(x) = 1$  indicates that  $X = x$  is certain
  - E.g.,  $p(x) = 0$  indicates that  $X = x$  is impossible
- ▶ **Properties**:
  - The domain  $D$  of  $p$  must be the set of all possible states of  $X$
  - $\forall x \in D(X), 0 \leq p(x) \leq 1$
  - $\sum_{x \in D(X)} p(x) = 1$





# Independence

- ▶ Two random variables  $X$  and  $Y$  are **independent**, if their **probability distribution** can be expressed as their **products**.

$$\forall x \in D(X), y \in D(Y), p(X = x, Y = y) = p(X = x)p(Y = y)$$

- ▶ E.g., if a **coin is tossed** and a single **6-sided die is rolled**, then the probability of landing on the **head** side of the coin and **rolling a 3** on the die is:

$$p(X = \text{head}, Y = 3) = p(X = \text{head})p(Y = 3) = \frac{1}{2} \times \frac{1}{6} = \frac{1}{12}$$



# Conditional Probability

- ▶ **Conditional probability:** the probability of an event given that another event has occurred.

$$p(Y = y \mid X = x) = \frac{p(Y = y, X = x)}{p(X = x)}$$

- ▶ E.g., if 60% of the class passed both labs and 80% of the class passed the first labs, then what percent of those who passed the first lab also passed the second lab?
  - E.g.,  $X$  and  $Y$  random variables for the first and the second labs, respectively.

$$p(Y = \text{lab2} \mid X = \text{lab1}) = \frac{p(Y = \text{lab2}, X = \text{lab1})}{p(X = \text{lab1})} = \frac{0.6}{0.8} = \frac{3}{4}$$



# Expectation

- ▶ The **expected value** of a random variable  $X$  with respect to a probability distribution  $p(X)$  is the **average** value that  $X$  takes on when it is drawn from  $p(X)$ .

$$E_{x \sim p}[X] = \sum_x p(x)x$$

- ▶ E.g., If  $X : \{1, 2, 3\}$ , and  $p(X = 1) = 0.3$ ,  $p(X = 2) = 0.5$ ,  $p(X = 3) = 0.2$ 
  - $E[X] = 0.3 \times 1 + 0.5 \times 2 + 0.2 \times 3 = 1.9$

# Variance and Standard Deviation

- ▶ The **variance** gives a measure of how much the **values of a random variable  $X$**  vary as we sample it from its **probability distribution  $p(X)$** .

$$\text{Var}(X) = E[(X - E[X])^2]$$
$$\text{Var}(X) = \sum_x p(x)(x - E[X])^2$$

- ▶ E.g., If  $X : \{1, 2, 3\}$ , and  $p(X = 1) = 0.3$ ,  $p(X = 2) = 0.5$ ,  $p(X = 3) = 0.2$ 
  - $E[X] = 0.3 \times 1 + 0.5 \times 2 + 0.2 \times 3 = 1.9$
  - $\text{Var}(X) = 0.3(1 - 1.9)^2 + 0.5(2 - 1.9)^2 + 0.2(3 - 1.9)^2 = 0.49$
- ▶ The **standard deviation**, shown by  $\sigma$ , is the **square root of the variance**.



## Covariance (1/2)

- ▶ The **covariance** gives some sense of **how much two values are linearly related** to each other.

$$\begin{aligned}\text{Cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\ \text{Cov}(X, Y) &= \sum \sum_{(x,y)} p(x, y)(x - E[X])(y - E[Y])\end{aligned}$$

## Covariance (2/2)

			Y		
	p(X, Y)	1	2	3	p(X)
	1	1/4	1/4	0	1/2
X	2	0	1/4	1/4	1/2
	p(Y)	1/4	1/2	1/4	1

$$E[X] = \frac{1}{2} \times 1 + \frac{1}{2} \times 2 = \frac{3}{2} \quad E[Y] = \frac{1}{4} \times 1 + \frac{1}{2} \times 2 + \frac{1}{4} \times 3 = 2$$

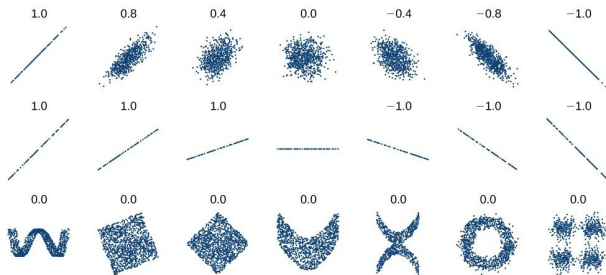
$$\text{Cov}(X, Y) = \sum_{(x,y)} p(x, y)(x - E[X])(y - E[Y])$$

$$= \frac{1}{4}(1 - \frac{3}{2})(1 - 2) + \frac{1}{4}(1 - \frac{3}{2})(2 - 2) + 0(1 - \frac{3}{2})(3 - 2) \\ + 0(2 - \frac{3}{2})(1 - 2) + \frac{1}{4}(2 - \frac{3}{2})(2 - 2) + \frac{1}{4}(2 - \frac{3}{2})(3 - 2) = \frac{1}{4}$$

# Correlation Coefficient

- ▶ The **Correlation coefficient** is a quantity that measures the **strength** of the **association** (or **dependence**) between two random variables, e.g., **X** and **Y**.

$$\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma(X)\sigma(Y)}$$





## Probability and Likelihood (1/2)

- ▶ Let  $X : \{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$  be a **discrete random variable** drawn **independently** from a **distribution probability  $p$**  depending on a **parameter  $\theta$** .
  - For six tosses of a coin,  $X : \{h, t, t, t, h, t\}$ , **h**: head, and **t**: tail.
  - Suppose you have a **coin** with probability  $\theta$  to land heads and  $(1 - \theta)$  to land tails.
- ▶  $p(X | \theta = \frac{2}{3})$  is the **probability** of  $X$  given  $\theta = \frac{2}{3}$ .
- ▶  $p(X = h | \theta)$  is the **likelihood** of  $\theta$  given  $X = h$ .
- ▶ **Likelihood (L)**: a function of the **parameters ( $\theta$ )** of a probability model, given **specific observed data**, e.g.,  $X = h$ .

$$L(\theta | X) = p(X | \theta)$$





## Probability and Likelihood (2/2)

- ▶ The **likelihood** differs from that of a **probability**.
- ▶ A **probability**  $p(X | \theta)$  refers to the occurrence of **future events**.
- ▶ A **likelihood**  $L(\theta | X)$  refers to **past events** with known outcomes.

# Maximum Likelihood Estimator

- ▶ If samples in  $\mathbf{X}$  are **independent** we have:

$$\begin{aligned}L(\theta | \mathbf{X}) &= p(\mathbf{X} | \theta) = p(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(m)} | \theta) \\ &= p(\mathbf{x}^{(1)} | \theta)p(\mathbf{x}^{(2)} | \theta) \cdots p(\mathbf{x}^{(m)} | \theta) = \prod_{i=1}^m p(\mathbf{x}^{(i)} | \theta)\end{aligned}$$

- ▶ The **maximum likelihood estimator (MLE)**: what is the **most likely value** of  $\theta$  given the training set?

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} L(\theta | \mathbf{X}) = \arg \max_{\theta} \prod_{i=1}^m p(\mathbf{x}^{(i)} | \theta)$$



## Maximum Likelihood Estimator - Example

- ▶ Six tosses of a coin, with the following model:
  - Possible outcomes: **h** with probability of  $\theta$ , and **t** with probability  $(1 - \theta)$ .
  - Results of coin tosses are independent of one another.
- ▶ Data:  $X : \{h, t, t, t, h, t\}$

- ▶ The likelihood is

$$\begin{aligned}L(\theta | X) &= p(X | \theta) \\ &= p(X = h | \theta)p(X = t | \theta)p(X = t | \theta)p(X = t | \theta)p(X = h | \theta)p(X = t | \theta) \\ &= \theta(1 - \theta)(1 - \theta)(1 - \theta)\theta(1 - \theta) \\ &= \theta^2(1 - \theta)^4\end{aligned}$$

- ▶  $\hat{\theta}$  is the value of  $\theta$  that maximizes the likelihood:

$$\hat{\theta}_{MLE} = \arg \max_{\theta} L(\theta | X) = \frac{2}{2 + 4}$$

- ▶ The MLE product is prone to numerical underflow.

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} L(\theta | X) = \arg \max_{\theta} \prod_{i=1}^m p(x^{(i)} | \theta)$$

- ▶ To overcome this problem we can use the logarithm of the likelihood.
  - It does not change its arg max, but transforms a product into a sum.

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} \sum_{i=1}^m \log p(x^{(i)} | \theta)$$



## Negative Log-Likelihood

- ▶ **Likelihood:**  $L(\theta | X) = \prod_{i=1}^m p(x^{(i)} | \theta)$
- ▶ **Log-Likelihood:**  $\log L(\theta | X) = \log \prod_{i=1}^m p(x^{(i)} | \theta) = \sum_{i=1}^m \log p(x^{(i)} | \theta)$
- ▶ **Negative Log-Likelihood:**  $-\log L(\theta | X) = -\sum_{i=1}^m \log p(x^{(i)} | \theta)$
- ▶ Negative log-likelihood is also called the **cross-entropy**



# Cross-Entropy

- ▶ **Cross-entropy**: quantify the **difference (error)** between **two probability distributions**.
- ▶ **How close** is the **predicted distribution** to the **true distribution**?

$$H(p, q) = - \sum_x p(x) \log(q(x))$$

- ▶ Where **p** is the **true distribution**, and **q** the **predicted distribution**.



## Cross-Entropy - Example

- ▶ Six tosses of a coin:  $X : \{h, t, t, t, h, t\}$
- ▶ The true distribution  $p$ :  $p(h) = \frac{2}{6}$  and  $p(t) = \frac{4}{6}$
- ▶ The predicted distribution  $q$ :  $h$  with probability of  $\theta$ , and  $t$  with probability  $(1 - \theta)$ .
- ▶ Cross entropy:  $H(p, q) = -\sum_x p(x)\log(q(x))$   
 $= -p(h)\log(q(h)) - p(t)\log(q(t)) = -\frac{2}{6}\log(\theta) - \frac{4}{6}\log(1 - \theta)$
- ▶ Likelihood:  $\theta^2(1 - \theta)^4$
- ▶ Negative log likelihood:  $-\log(\theta^2(1 - \theta)^4) = -2\log(\theta) - 4\log(1 - \theta)$



## References

- ▶ Ian Goodfellow et al., Deep Learning (Ch. 1, 2, 3)



# Questions?

## Acknowledgements

Some of the pictures were copied from the book Hands-On Machine Learning with Scikit-Learn and TensorFlow, Aurelien Geron, O'Reilly Media, 2017.