Neural Computation

Week 1 - Introduction

Yunwen Lei

School of Computer Science, University of Birmingham

Outline

Course Information

Module Team

Jinming Duan
j.duan@bham.ac.uk
(Module Leader)

Yunwen Lei
y.lei@bham.ac.uk

Kashif Rajpoot (Dubai)
k.m.rajpoot@bham.ac.uk

Office Hours

- Jinming Duan Zoom link
  - Fridays 2-3pm (UK time)
- Yunwen Lei Zoom link
  - Friday 5pm-6pm (UK time)
- Kashif Rajpoot (Dubai) Zoom link
  - Tuesday 4pm-5pm (UAE time)
  - Thursday 4pm-5pm (UAE time)

TAs

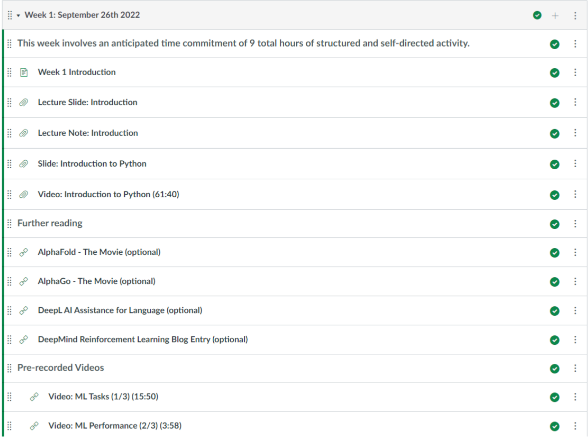
- Yang Yue yxy142@student.bham.ac.uk
- Luoying Hao lxh122@student.bham.ac.uk
- Joseph Bartlett jxb1336@student.bham.ac.uk
- Omer Eryilmaz obe851@student.bham.ac.uk
- Tianyang Zhang TXZ009@student.bham.ac.uk
- Samuel Tonks sxt118@student.bham.ac.uk

Assessment & Feedback

- Two Quiz (individual)
  - The first quiz is around middle term (10%)
  - The second quiz is around end term (10%)
- Written exam: 80% (individual), on-campus or online (tbc)
- Answers for exam and quiz will be revealed on Canvas
- For non-assessed formative assignments (eg. labs and tutorials), use Microsoft Teams or contact our Teaching Assistants (TAs)

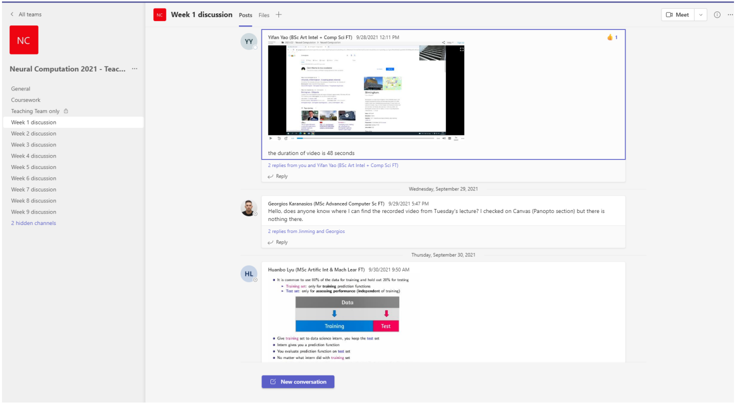
Canvas Layouts

- All learning materials are contained on the Modules page
- Each section of the modules page covers one week
- Introduction is provided for each week, summarising what we talk about, and the order in which we recommend that you study the material
- Each week covers one or more topics. Each topic will normally have slides, reading material, and a practical lab or tutorial



Canvas Layouts Cont.

- Anything that is not marked “optional” is compulsory reading and may be covered in the final exam
- The Assignments page contains links to practical labs or tutorials and quiz
- Microsoft Teams is used to answer questions about the module. It is an online service provided outside Canvas. You need to enroll onto the Neural Computation Team (Announcements)



Canvas Layouts Cont.

- The Announcements section contains important messages from the module staff, such as changes to deadlines etc. Stay alert
- For each week, there will be two timetabled (Monday and Thursday) sessions in a lecture hall on campus.
  - Monday 14:00, Y3 G31 (Engineering building)
  - Thursday 13:00 Jeffreys LT (135) (Education building, R19)

Module FAQs

- Microsoft Teams for most questions
  - You can follow instruction from the link to join the Microsoft Teams
- Office hours via Zoom or on campus in CS
- Welfare issues (cswelfare@contacts.bham.ac.uk)

Module Overview

Weeks	Date	Lecturer	Content
w1	26 Sep	Yunwen	Introduction
w2	3 Oct	Yunwen	Linear Regression
w3	10 Oct	Yunwen	Gradient Descent
w4	17 Oct	Yunwen	Perceptron, Multilayer Perceptron
w5	24 Oct	Yunwen	Computational Graph, Backward Propagation
w6	31 Oct	Yunwen	Advanced Optimisation
w7	7 Nov	Jinming	Convolutional Neural Networks, Segmentation
w8	14 Nov	Jinming	Recurrent Neural Networks
w9	21 Nov	Jinming	Statistical Interpretation of NNs, Autoencoders
w10	28 Nov	Jinming	Variational Autoencoders
w11	5 Dec	Jinming	Generative Adversarial Networks
w12	12 Oct		Revision
	Jan		Written Exam

Education Aims

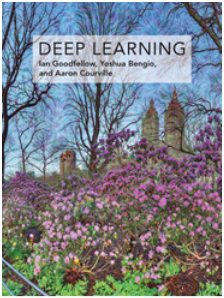
- Introduce some of fundamental techniques and principles of neural networks
- Investigate some common neural network models and their applications
- Present neural networks in the larger context of state-of-the-art techniques of automated learning

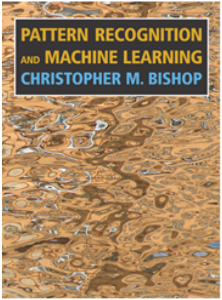
Teaching Environment

- Python programming language (PyTorch)
- Jupyter notebook IDE (all students)
- School's machines in UG04 or LG04 (on-campus students)

Recommended Textbooks

- Deep learning, Goodfellow et al., MIT Press, 2016 (online and in School Library)
- Pattern Recognition and Machine Learning, Bishop, Springer, 2007 (in Main Library)





Deep Learning

Free Machine Learning Books

You can click it to get the link

- Pattern Recognition and Machine Learning
- The Element of Statistical Learning
- Gaussian Process for Machine Learning

Free Online Courses

- Summer School
  - Deep Learning & Reinforcement Learning Summer School http://videolectures.net/DLRLsummerschool2018_toronto/
  - Machine Learning Summer School (MPI) https://www.youtube.com/playlist?list=PLqJm7Rc5-EXFv6RXaPZzzlzo93Hl0v91E
- Module
  - DeepLearning.ai (Andrew Ng) https://www.youtube.com/channel/UCcIXc5mJsHVYTZR1maL5l9w/playlists
  - CS231n (Stanford) https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv
  - CS224d (Stanford) https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6
  - Fast.ai (Fast.ai) https://www.youtube.com/playlist?list=PLCdvEQLhYkYmKTKWTrH7bHtQ1CsKZaQBl

Prerequisite

The course requires a solid maths background

- Linear Algebra: vector/matrix manipulations
  - MIT Open Course Linear Algebra
- Calculus: partial derivative, chain rule
  - MIT Open Course Multivariable Calculus
- Probability (recommended)
- Python (recommended): Learn NumPy + SciPy + Matplotlib

Definition of Machine Learning

What is Machine Learning

Definition by Tom Mitchell (1997)

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.

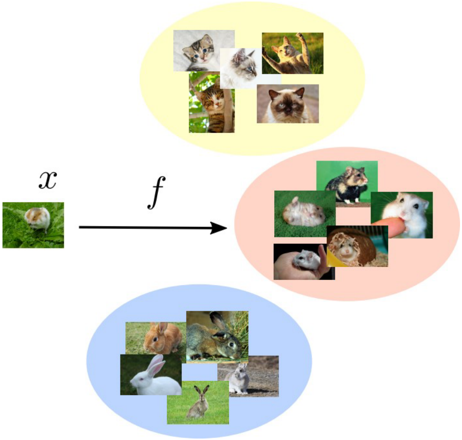
Toy block building

- E: physical world
- T: building a tower with toy block
- P: how tall the tower is

Machine Learning Tasks T

- Classification
- Regression
- Transcription
- Machine translation
- Synthesis and sampling
- . . .

Classification



- Construct a function

$$f : \mathbb{R}^d \mapsto \{1, \ldots, k\}$$

such that if an object with features $\mathbf{x} \in \mathbb{R}^d$ belongs to class y then

$$f(\mathbf{x}) = y$$

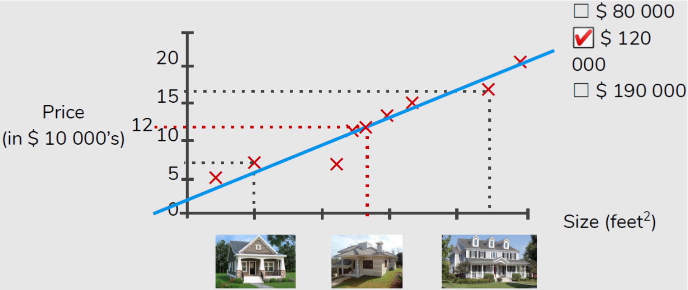
- Alternatively, construct a function which given features returns the probability of each class

Regression

- Predict a numerical output given some input, i.e., a function

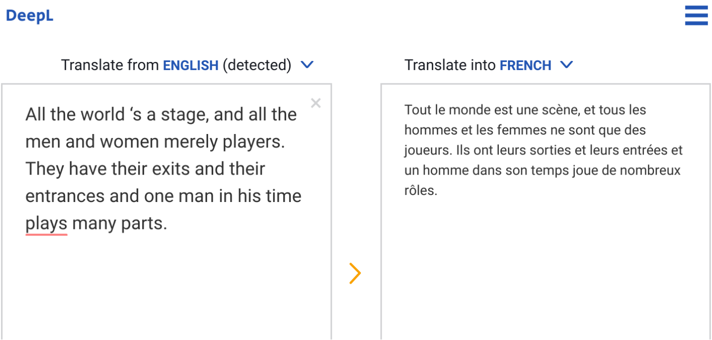
$$f : \mathbb{R}^d \mapsto \mathbb{R}.$$

- Example: house price prediction
  - **Input**: Information of House (living size, lot size, location, # floors)
  - **Output**: Price

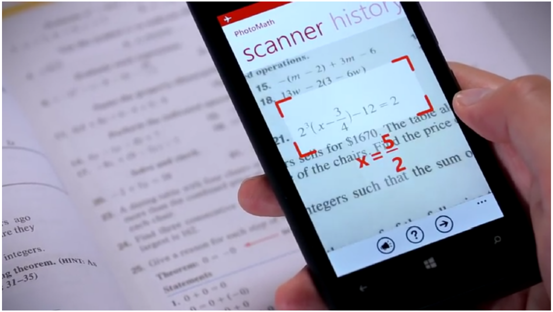


Machine Translation

- Translation from a source language to a target language
  - **Input**: sequence of characters (e.g., English text)
  - **Output**: sequence of characters (e.g., French text)



Photomath



Photomath `https://photomath.net/`

Which technique is used here?

Experience E and Performance P

Experience E: we get experience by observing a dataset from nature

- unsupervised learning: , , , ...

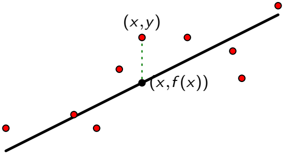
  or formally $S = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\}$

- supervised learning: (,`dog`), (,`car`), (,`airplane`), ...

  or formally $S = \left\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\right\}$

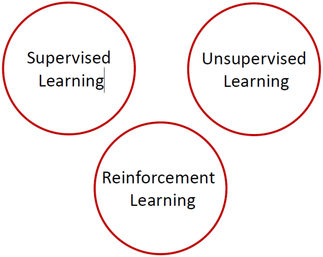
Performance P:

- **Accuracy** is a common performance measure for classification tasks
  - Proportion of correctly classified examples
- **Residual** is a common performance measure for regression tasks



Types of Learning Tasks

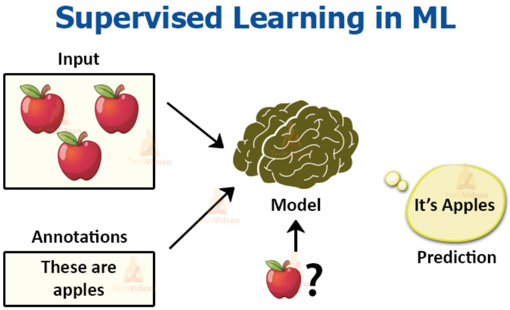
- **Supervised Learning**
  - Correct output known for each training example
  - Learn to predict output when given an input vector
- **Unsupervised learning**
  - Create an internal representation of the input, capturing regularities/structure in data
- **Reinforcement learning**
  - Learn to how to interact with environement
  - Learn action to maximize payoff



Supervised Learning

Supervised Learning

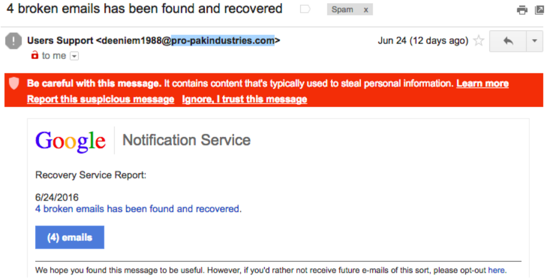
Learn a function that maps an input to an output based on examples of input-output pairs.



- Classification is used to predict discrete values (class labels).
- Regression is used to predict continuous values

Binary Classification: Spam Detection

- **Input**: Incoming email



- **Output**: “SPAM” or “’NOT SPAM’
- A binary classification problem, because only 2 possible outputs
- **Performance**: the number of messages correctly classified

Multiclass Classification: Medical Diagnosis

- **Input**: Symptoms (fever, cough, fast breathing, shaking, no smell,...)
- **Output**: Diagnosis (coronavirus, flu, common cold, pepneumonia, ...)
- A multiclass classification problem: choosing one of several [discrete] outputs
- **Performance**: the number of correct diagnosis

How to express uncertainty?

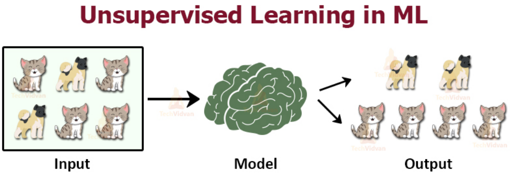
- Probabilistic classification

$$\mathbb{P}(coronavirus) = 0.7, \quad \mathbb{P}(flu) = 0.2, \quad \mathbb{P}(commoncolde) = 0.1$$

Unsupervised Learning

Unsupervised Learning

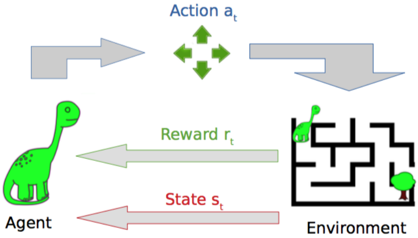
Learn interesting patterns from dataset with no labels: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n)}$



- **Clustering** algorithm tries to detect similar groups.
- **Dimensionality reduction** tries to simplify the data without losing too much information.

Reinforcement Learning

- An agent interacts with an environment (e.g., game of maze)
- In each time step
  - the agent receives **observations** (e.g. (x, y)) which give it information about the **state** (e.g. positions of the dinosaur)
  - the agent picks an **action** (e.g. moving direction) which affects the **state**
- The agent periodically receives a **reward** (e.g, -1 per step)
- The agent wants to learn a **policy**, or mapping from observations to actions, which maximizes its average reward over time



Training and Testing

ML: Training and Testing

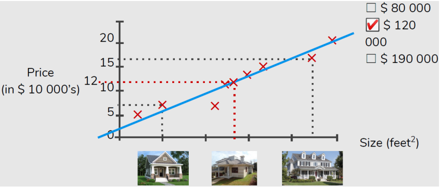
- **Input**: Information of House (living size, lot size, location, # floors)
- **Output**: Price
- **Aim**: find a model to predict price based on feature information of house
- **Training dataset**: a sequence of (features, price) pairs

\$ 70,000

\$ 160,000

- **Prediction/testing**: given information of new house, predict its price?

???



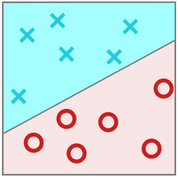
- **Performance**: difference between predicted price and true price

Training and Testing

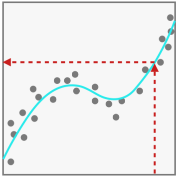
- **Training dataset**: a dataset that contains n samples

$$(\underbrace{\mathbf{x}^{(1)}}_{\text{input}}, \underbrace{y^{(1)}}_{\text{output}}), (\underbrace{\mathbf{x}^{(2)}}_{\text{input}}, \underbrace{y^{(2)}}_{\text{output}}), \ldots, (\underbrace{\mathbf{x}^{(n)}}_{\text{input}}, \underbrace{y^{(n)}}_{\text{output}})$$

  - Classification: $y \in \{-1, +1\}$
    $+1$ means positive examples
    -1 means negative examples
  - Regression: $y \in \mathbb{R}$



Classification



Regression

- Aim to find a function $f : \mathcal{X} \mapsto \mathcal{Y}$ such that

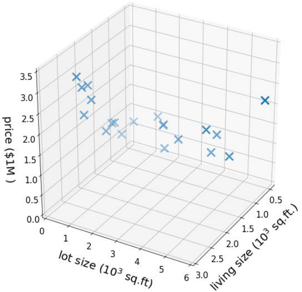
$$y \approx f(\mathbf{x})$$

- **Prediction**: given a new input $\mathbf{x}$, use f to do prediction
  - if a house has x square feet, predicting its price?

Features

- $\mathbf{x} \in \mathbb{R}^d$ for large d, e.g.,

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_d \end{pmatrix} \begin{array}{cc} - & \text{living size} \\ - & \text{lot size} \\ - & \# \text{ floors} \\ - & \text{condition} \\ - & \text{zip code} \\ & \vdots \end{array} \implies y\text{– price}$$



Loss Function

- A loss function scores how far off a prediction is from the desired "target" output
  - Loss(prediction,target) returns a number called "the loss"
  - Big Loss = Bad Error
  - Small Loss = Minor Error
  - Zero Loss = No Error
- **0-1 loss** for classification
  - Loss is 1 if prediction is wrong
  - Loss is 0 if prediction is correct
- **Square** loss for regression
  - loss = $(\text{predicted - target})^2$

Evaluating a Prediction Function

- Data science intern gives you a prediction function $f(x)$
  - "Average error on training data was 1%"
- Product manager says "we can deploy if $\leq$ 2% error"
- Deploy this prediction function?
  - No!
- Prediction function needs to do well on **new inputs**
- The only way to know how well a model will generalize to new cases is to actually try it out on new cases
  - Training set: only for **training** prediction functions
  - Test set: only for **assessing performance** (**independent** of training)

Performance on training set has a bias towards the model and cannot serve as an estimate of its performance on true environment!

Training Error and Test Error

- **Training error**: the error for using a model f to do prediction on **training data**

$$\mathrm{Err}_{train}(f) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{Loss}(f(\mathbf{x}^{(i)}), y^{(i)})$$

  - for house price prediction, this can be $\mathrm{Loss}(f(\mathbf{x}^{(i)}), y^{(i)}) = (f(\mathbf{x}^{(i)}) - y^{(i)})^2$
- **Testing error**: the error for using a model f to do prediction on **test data**
- **Error decomposition**: we decompose the test error by

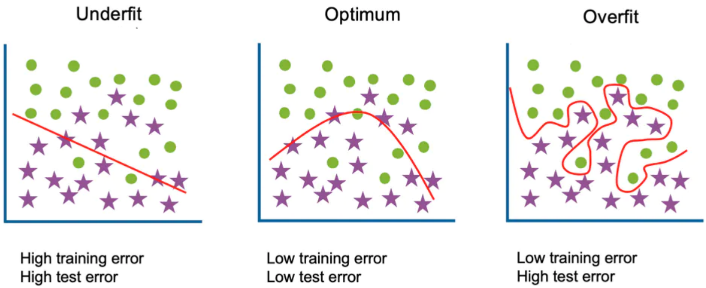
$$\mathrm{Err}_{test}(f) = \mathrm{Err}_{train}(f) + \underbrace{\mathrm{Err}_{test}(f) - \mathrm{Err}_{train}(f)}_{\mathrm{Gen}_{\mathrm{gap}}(f)}$$

  - Typically, the generalization gap $\mathrm{Gen}_{\mathrm{gap}}(\mathrm{f})$ is greater than 0
  - A simple model has a small $\mathrm{Gen}_{\mathrm{gap}}(f)$

Underfitting and Overfitting

Underfitting and Overfitting

- Loosely speaking, we say a model underfits when
  - training performance is poor
- We say a model overfits when
  - training performance is good but
  - test performance is poor

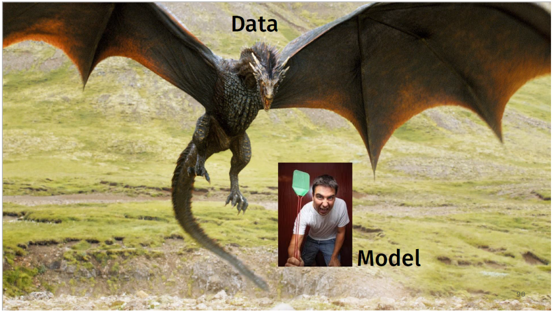


Overfitting the Training Data

- Overfitting means that the model performs well on the training data but it does not generalize to testing data
- It happens when the model is too complex relative to the amount and noisiness of the training data.

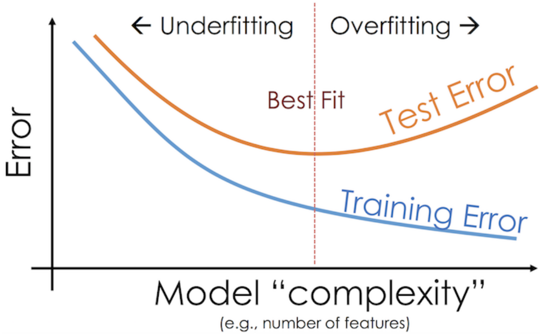
Underfitting the Training Data

- Underfitting is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data.



Underfitting and Overfitting

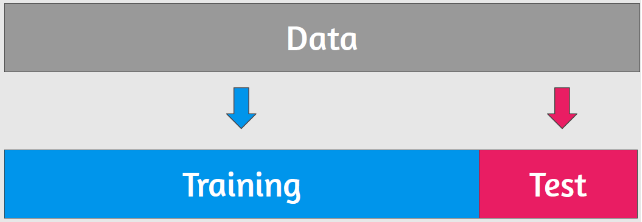
- In general, the training error decreases as we add complexity to our model with additional features or more complex prediction mechanisms.
- The test error, on the other hand, decreases up to a certain amount of complexity then increases again as the model overfits the training set.



Basic Machine Learning Workflow

Data Partition

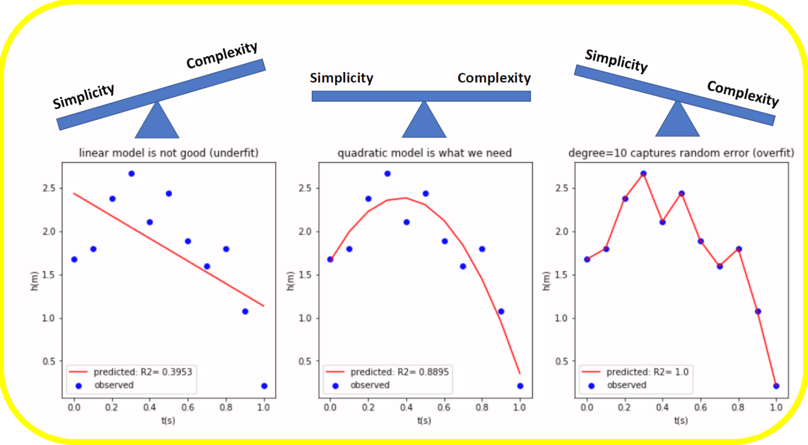
- It is common to use 80% of the data for training and hold out 20% for testing
  - Training set: only for **training** prediction functions
  - Test set: only for **assessing performance** (**independent** of training)



- Give training set to data science intern, you keep the test set
- Intern gives you a prediction function
- You evaluate prediction function on test set
- No matter what intern did with training set
  - test performance should give you good estimate of deployment performance

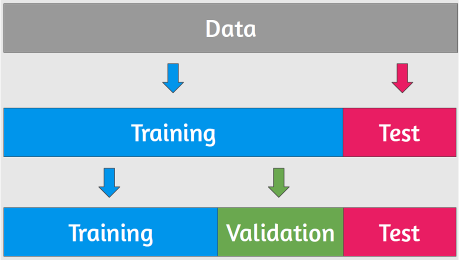
How to Find a Good Model?

Intern wants to try **many** fancy ML models: each gives a different prediction function. How should she choose one model?



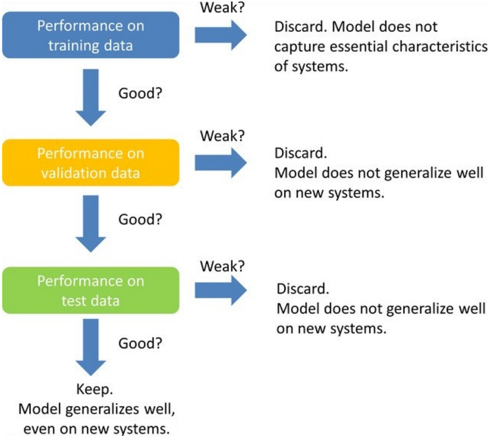
How to Find a Good Model?

- Intern needs her own test set to evaluate prediction functions.
- Intern should randomly split data again into (80/20 split)
  - training set and
  - validation set
- Validation set is like test set, but used to choose best among many prediction functions
- Test set is just used to evaluate the final chosen prediction function



Basic Machine Learning Workflow

- Split labeled data into **training**, **validation**, and **test** sets
- Repeat until happy with performance on validation set
  - Choose some ML algorithm
  - Train ML models with various complexities
  - Evaluate prediction functions on validation set
- Evaluate performance on test set



Summary

- According to Mitchell (1997), machine learning occurs when
  - performance P of algorithm at task T improves with experience E
- Types of ML
  - supervised learning: predict input-output mapping
  - unsupervised learning: find pattern among input data
  - reinforcement learning: interaction with environment
- Key concepts in ML
  - training and testing
  - overfitting and underfitting
  - basic ML workflow

Next Lecture

Linear Models