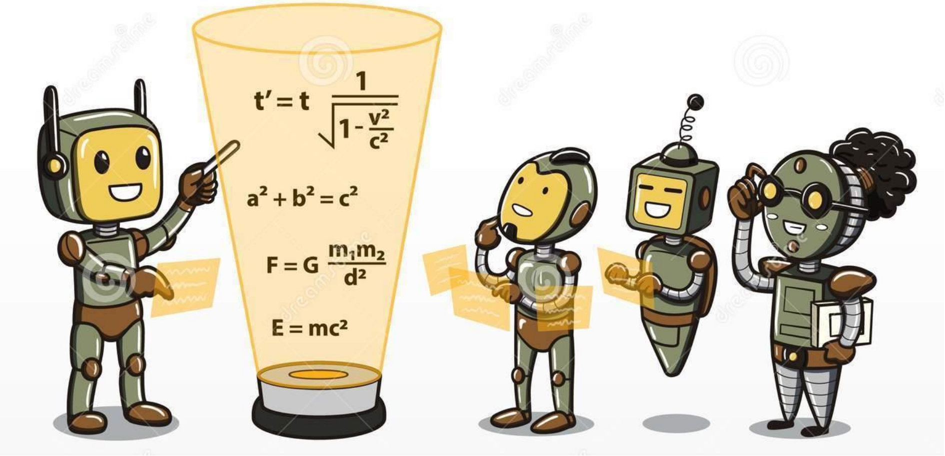
CSE 445: Machine Learning

Introduction



Instructor: Intisar Tahmid Naheen, North South University

Credit: xkcd "machine learning"

Resources

- Slides provided in course should be enough – but there is a plethora of fantastic resources available, so use them!
- Recommended Books:
  - **Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow** by Aurelien Geron (will be followed extensively in the course with code examples from https://github.com/ageron/handson-ml)
  - **Pattern Recognition and Machine Learning** by Christopher Bishop (excellent resource for mathematical foundations)
  - **Elements of Statistical Learning** by Jerome Friedman et al (good reference)
- Additional Material:
  - Andrew Ng’s course on Machine Learning available on Coursera
  - CS 189, Berkeley
  - CS 229, Stanford

Helpful Prerequisites

- MAT361- Probability & Statistics
  - Probability distribution, Random Variable, Conditional Probability, Variance (some of the important concepts to recall to name a few)
- MAT125 – Linear Algebra
  - Matrix Multiplication, Eigenvalues, Eigenvectors
- Basic programming background in Python (an OK understanding of python syntax is all that's necessary – Geron's textbook has excellent code examples)
- None of them are compulsory – easier to grasp the material if completed

Course Project

- Groups of up to 3 members (3 is a hard maximum)
- Video Demo submission at the end of the semester, and in-person/online presentation at the end of the semester
- 4-6 page Report due at semester end, IEEE format – must include link to Github repo

- Potential Topics (few examples):
  - Covid-19
  - Computer Vision
  - Natural Language Processing
  - Reinforcement Learning
  - Speech & Music Recognition
  - Biomedical Imaging and Biosignals

What is Machine Learning?

Tom Mitchell (1998): 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.

Example:

Task: Playing Checkers

Experience (data): games played by the program (with itself)

Performance measure: winning rate

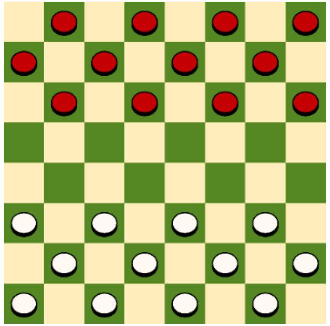


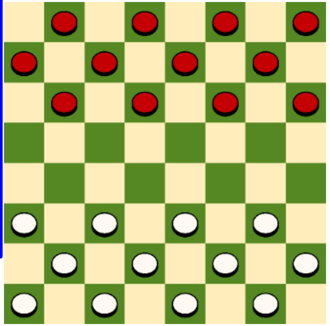
Image from Tom Mitchell's homepage

Definition of Machine Learning

Arthur Samuel (1959): Machine Learning is the field of study that gives the computer the **ability to learn** without being **explicitly programmed**.

A. L. Samuel*

Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress



Photos from Wikipedia

Traditional Programming

- Traditional Programming: writing a set of **RULES** to find **ANSWERS** from **DATA**

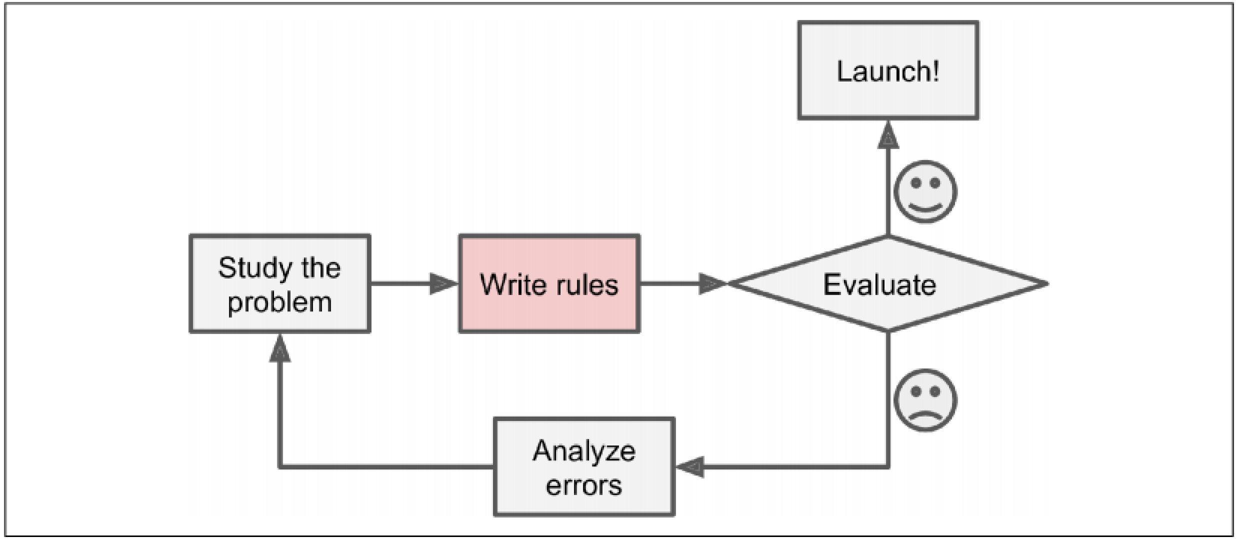


Figure 1-1. The traditional approach

The ML Approach

Machine Learning: Use **DATA** and **ANSWERS** to learn the underlying set of **RULES**

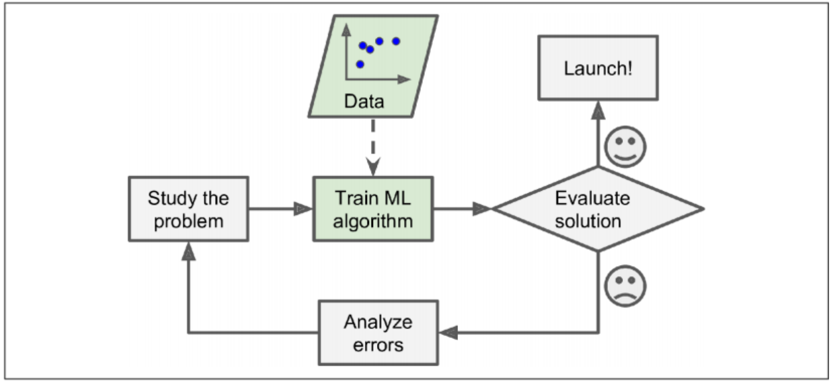


Figure 1-2. The Machine Learning approach

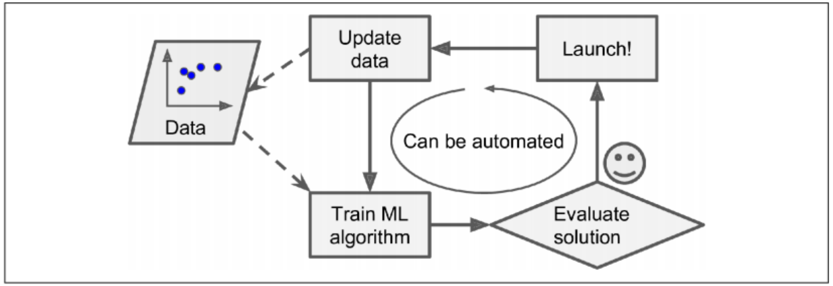


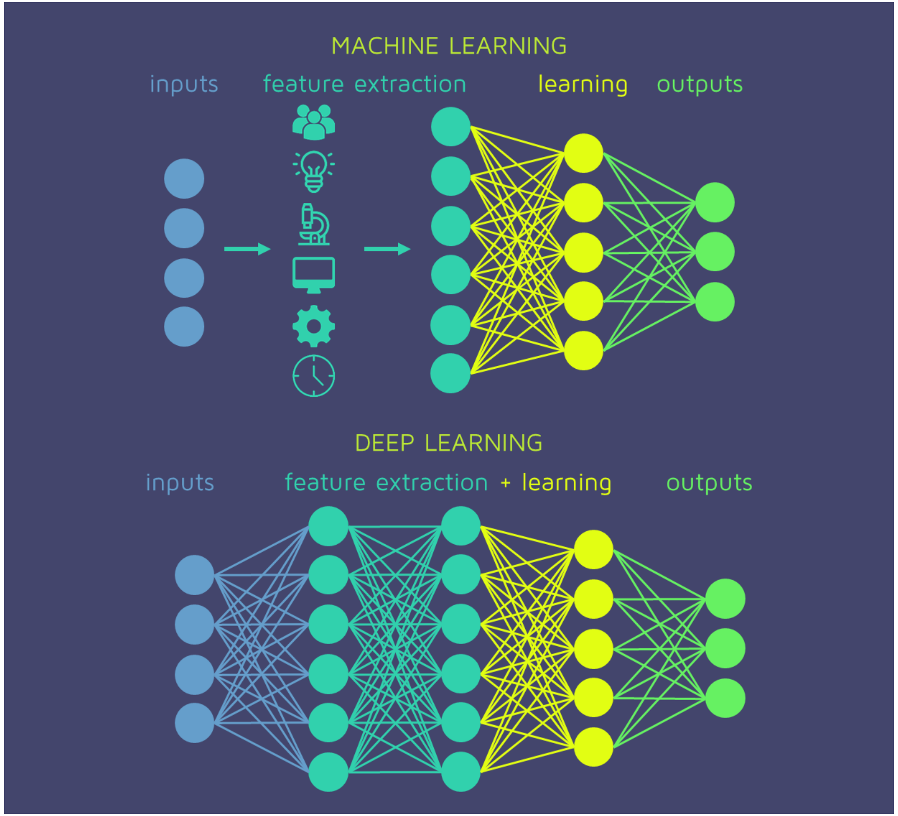
Figure 1-3. Automatically adapting to change

Great for:

- Problems that require a lot of **fine-tuning** or **long list of rules**
- **Changing environments** – ML systems can ADAPT
- Getting insights from **large amounts of data**
- **Complex** problems that yields no good solution with traditional approach

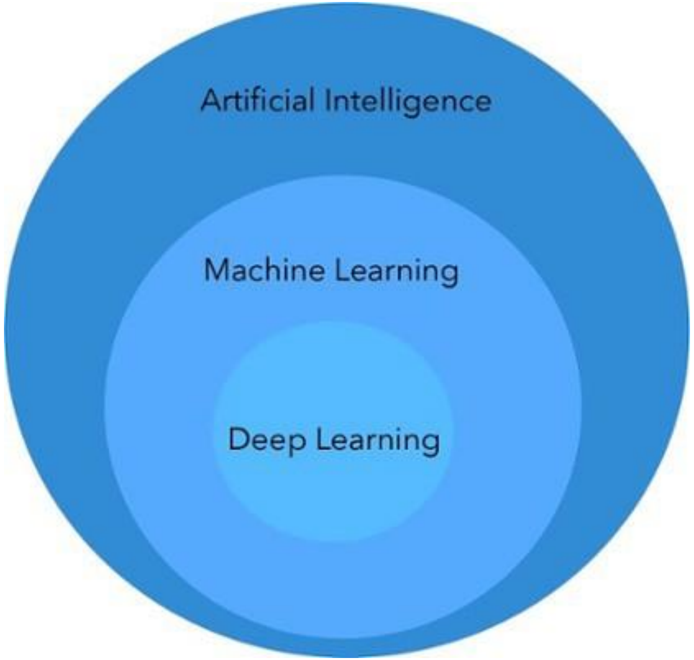
Deep Learning

- Subset of ML - loosely mimics structure/function of human brain
- Unlike traditional ML, does not require manual feature extraction
- Keeps getting better with more data (typically)
  - Computer Vision (CNN, GAN)
  - Natural Language Processing (RNN, LSTM)
  - Automatic Speech Recognition (RNN)



Summary – AI vs ML vs DL

- Subsets of each other
  - 1950 – 1990: AI in the form of Expert systems (airplane autopilot) and Games (checkers, chess)
  - 1990- : Statistical Approaches with ML, busts AI winter
  - 2010 - : Deep Learning revolutionizes CV, NLP among other applications
- Narrow AI
  - Systems can do a few defined things (such as playing chess, or driving a car) as well, or better than humans
  - Can't do EVERYTHING a human being can do – yet
- AI is not "taking over the world" anytime soon
  - Tell your uncles to relax and stop using Whatsapp



What kind of ML system is it?

- Useful to classify ML systems based on the following criteria:

1. Does it require human supervision?
   - Supervised Learning
   - Semisupervised Learning
   - Unsupervised Learning
   - Reinforcement Learning

2. Can it learn incrementally on the fly?
   - Online Learning
   - Batch Learning

3. Does the system build a predictive model?
   - Model-based Learning
   - Instance-based Learning

- These are not exclusive – can be combined
- e.g. Spam filter may learn on the fly with a deep neural network – online, model-based, supervised learning system

Supervised Learning

- Training data fed to algorithm includes the desired answers/solutions (**labels)**
- Example algorithms:
  - Linear Regression
  - Logistic Regression
  - SVM
  - Decision Tree
  - Neural Network

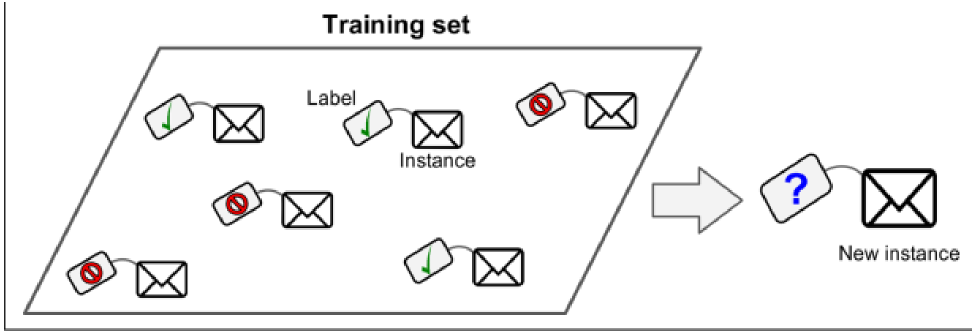


Figure 1-5. A labeled training set for spam classification (an example of supervised learning)

Figure 1-6. A regression problem: predict a value, given an input feature (there are usually multiple input features, and sometimes multiple output values)

Unsupervised Learning

- Training data is unlabeled
  - System learns without direct human supervision
- Widely used in:
  - Clustering
  - Anomaly detection
  - Association mining
  - Data preprocessing
- Example algorithms:
  - K-means
  - PCA
  - SVD
  - ICA

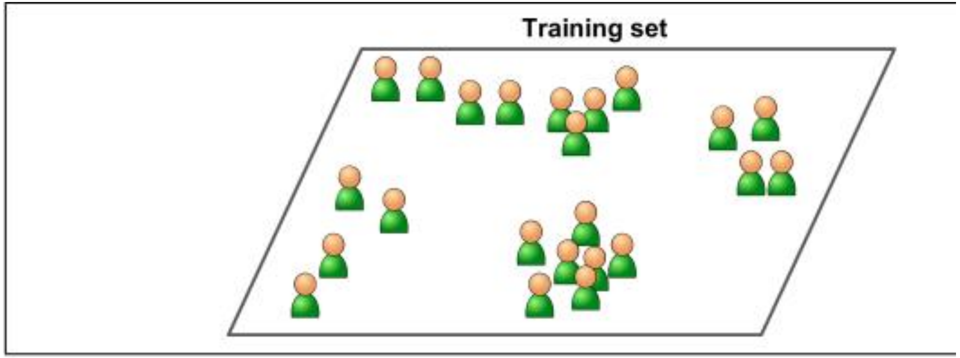


Figure 1-7. An unlabeled training set for unsupervised learning

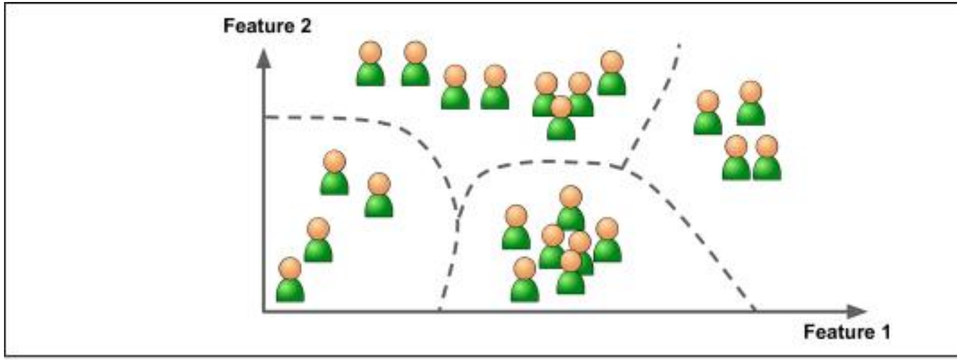
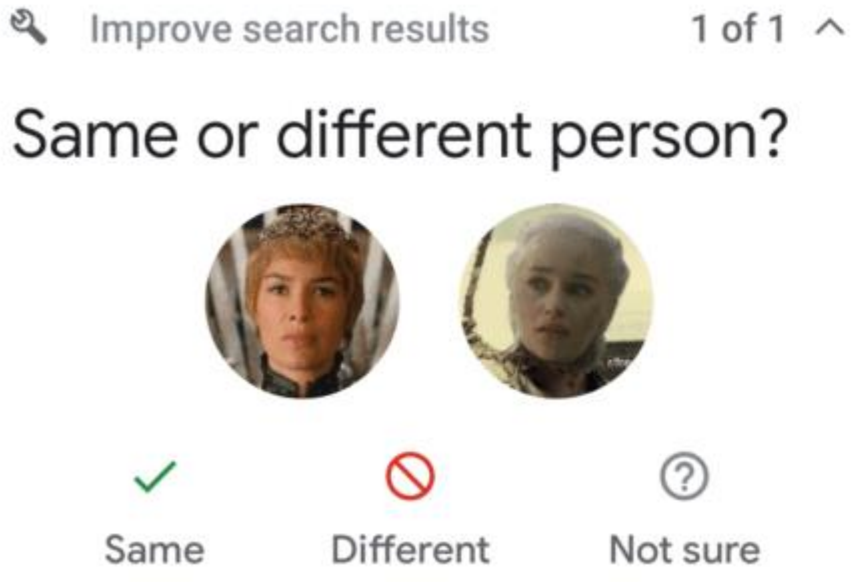


Figure 1-8. Clustering

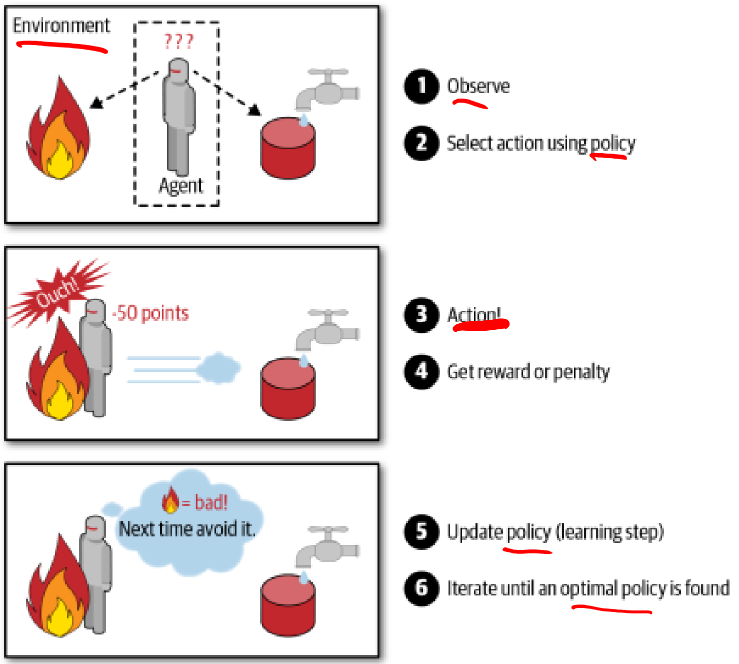
Semisupervised Learning

- Partially labeled data
  - Unsupervised learning used to cluster similar data together
  - Human input taken to label the clusters
  - e.g. Google Photos will cluster similar faces, and ask the user if they are the same person



Reinforcement Learning

- The learning system (agent) can:
  - Observe the environment
  - Select and perform an action
  - Get rewards/penalties as a result
- Learns what the best policy should be
  - Policy defines what actions should be chosen in a certain situation
- Very effective in controlled environments (such as a game of chess)
  - With the progress in deep learning, increasingly used in more complex tasks (such as driving the mars rover)



Batch Learning vs Online Learning

- Batch Learning
  - Not capable of learning after deployment
  - Must be retrained from scratch – computationally expensive!
- Online Learning
  - Can continue to learn after deployment
  - Can take advantage of parallel computing – no down time
  - Preferred choice in production

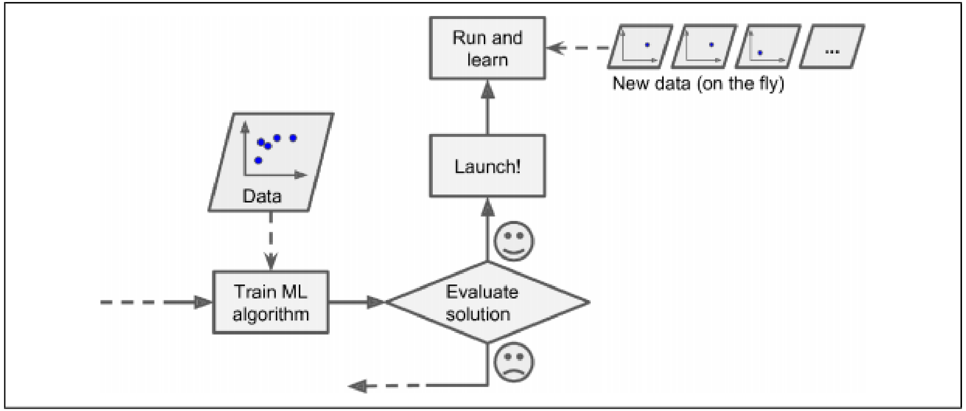


Figure 1-13. In online learning, a model is trained and launched into production, and then it keeps learning as new data comes in

Example ML Task: Does money make people happy?

Table 1-1. Does money make people happier?

Country	GDP per capita (USD)	Life satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
United States	55,805	7.2

- Life Satisfaction data from OECD
- GDP per capita data from IMF

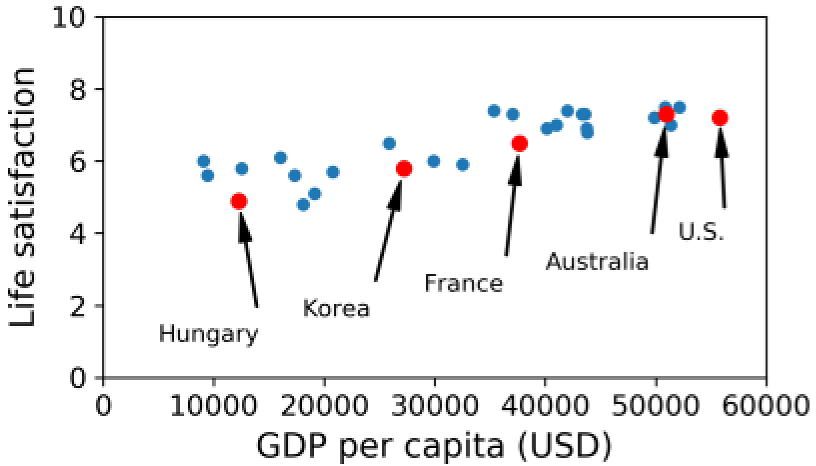


Figure 1-17. Do you see a trend here?

What relationship can we infer between life satisfaction and GDP per capita from the graph?

Problems with Machine Learning

- 3 V's of Big Data
  - **Volume, Variety, Velocity**
- Problem #1: Training data!
  - Insufficient quantity
  - Nonrepresentative data
  - Poor-quality data
- Problem #2: How "fit" is it?
  - Overfitting data
  - Underfitting data
- Problem #3: Which features should be used?
  - Deep Learning automates **feature selection**

Overfitting

- Most common problem in ML – do not overgeneralize!
  - The polynomial model is better than the linear model on training
  - How about testing?

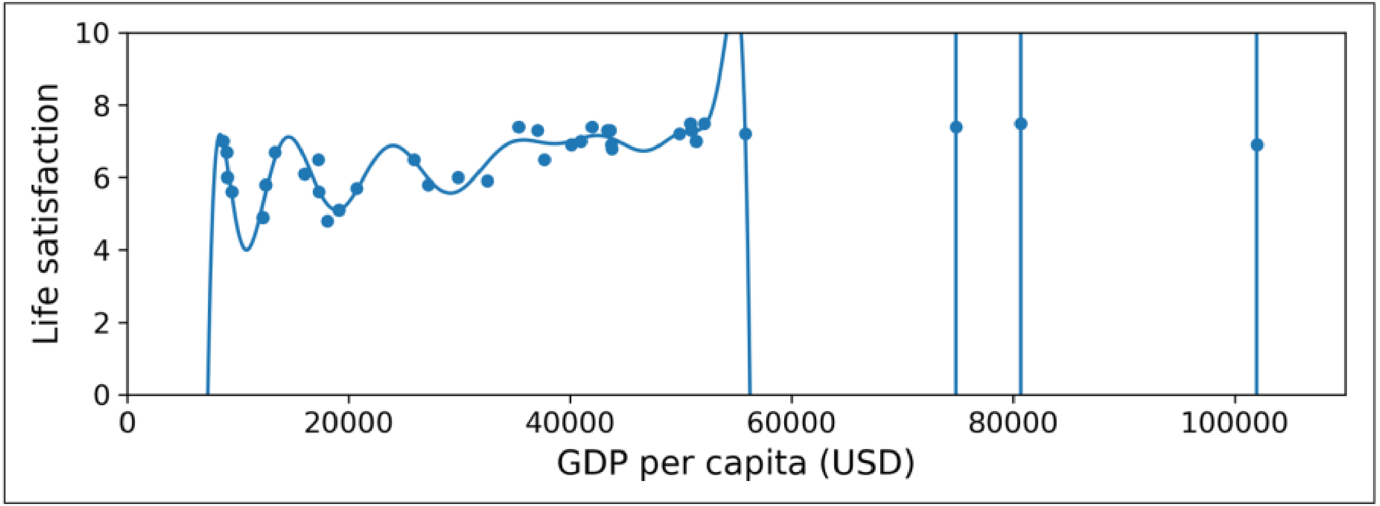


Figure 1-22. Overfitting the training data

How to avoid overfitting

- Tip #1: REGULARIZATION – USE IT
  - **Constrain** model to keep it simple – reduce risk of overfitting
  - If you can stand on one leg, you'll be able to stay balanced with two legs
  - **Hyperparameters –** control level of regularization
- Tip #2: Get more training data, and reduce noise in it

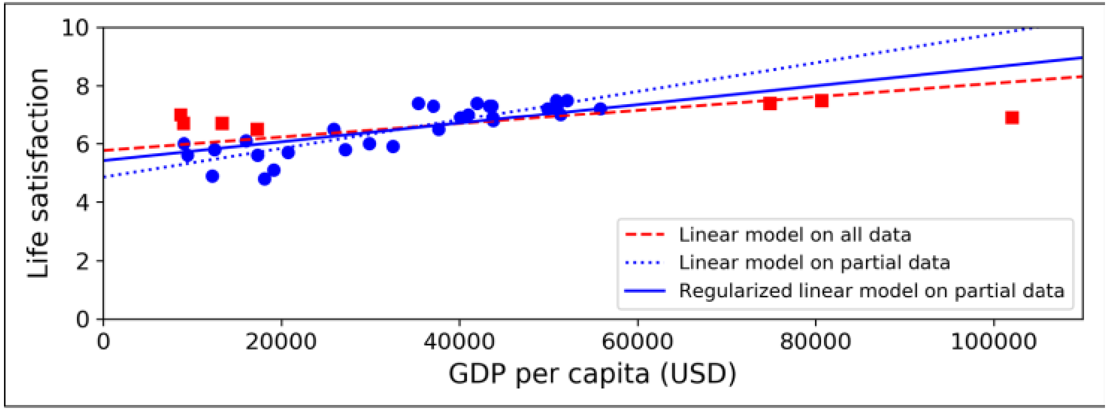


Figure 1-23. Regularization reduces the risk of overfitting

Model Evaluation

- How good is your model?
  - Test it on **new data** – data not seen by the model ever before!
  - Keep **80%** for training, set **20%** for testing
  - NEVER go below 10% test data – better model is better than better "accuracy"
- How to regularize?
  - Keep a portion of **training data** held out for **validation**
  - Alternatively, use **cross-validation** (many validation sets instead of one)
  - Pick the hyperparameters that work best on validation for your model on the test dataset

Ratios

- A great model
  - trained with 60% training data, 20% validation data, and 20% testing data
- An okay model
  - trained with 70% training data, 15% validation data, and 15% testing data
- A barely-acceptable model
  - trained with 80% training data, 10% validation data, and 10% testing data
- Models with worse ratios – hacks
  - Unless there's millions of instances in the dataset
- "No Free Lunch" theorem
  - Only way to know for sure which model works best is to evaluate them
  - Make reasonable assumptions about your data to select model