Machine Learning and Deep Learning

CIVE AI Workshop: CIVE UDOM Tanzania.

Anthony Faustine

PhD machine learning researcher
(IDLab research group-Ghent University)

05^{th} January 2019

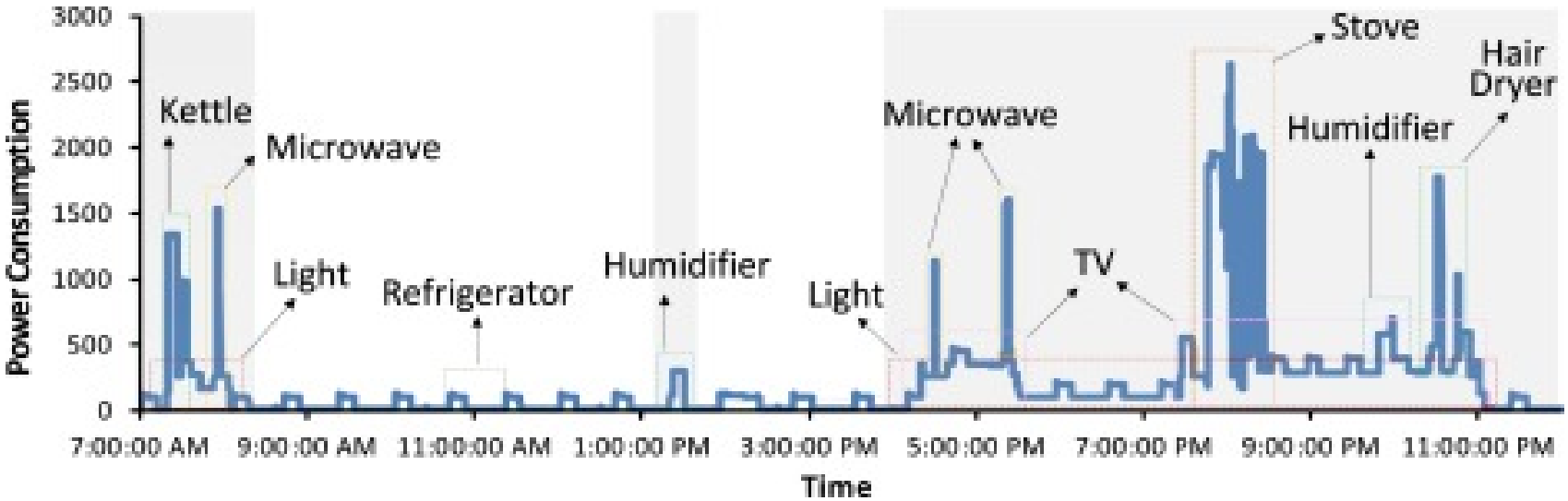
GHENT UNIVERSITY

IDLab
INTERNET & DATA LAB

UNIVERSITEIT GENT

Bio

Non-Intrusive Load Monitoring



Research interest: Stastical Machine Learning, Deep learning, Bayesian Inference, Signal processing.

Learning goal

- Understand the basics of Machine learning, motivation and its applications.
- Learn how to formulate learning problem, the challenge and how to evaluate ML problem.
- Understand Regression and Classification ML problem.
- Understand the basic building block of deep learning model, modern deep learning architectures and their application.
- Learn different techniques used in practise to build ML with focus on deeplearning.
- Explore opportunities and research direction of ML.

Outline

Introduction to Machine Learning

Machine learning (ML): the science (and art) of programming computers so they can learn from data.

ML is about learning, reasoning and acting based on data.

It gives computers the ability to learn without being explicitly programmed for the task at hand.

Learning and inference

- Automatically detect patterns in data and
- Build models that explain the world
- Use the uncovered pattern to understand what is happening (inference) and to predict what will happen(prediction).

Introduction to Machine Learning

Machine learning (ML): the science (and art) of programming computers so they can learn from data.

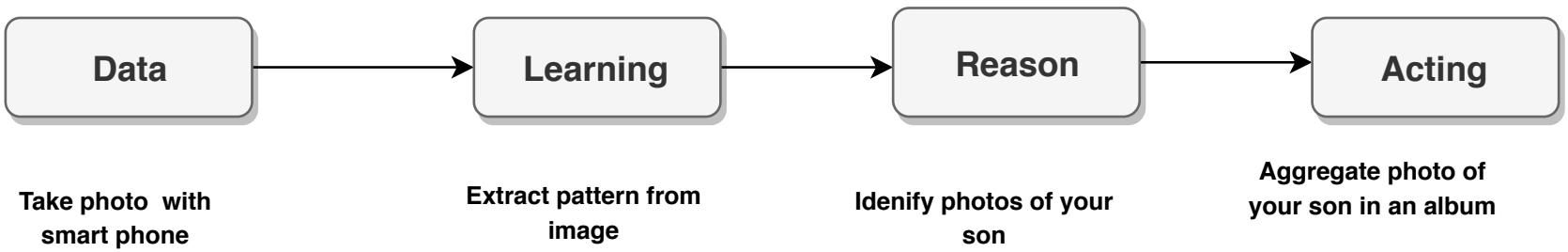
ML is about learning, reasoning and acting based on data.

It gives computers the ability to learn without being explicitly programmed for the task at hand.

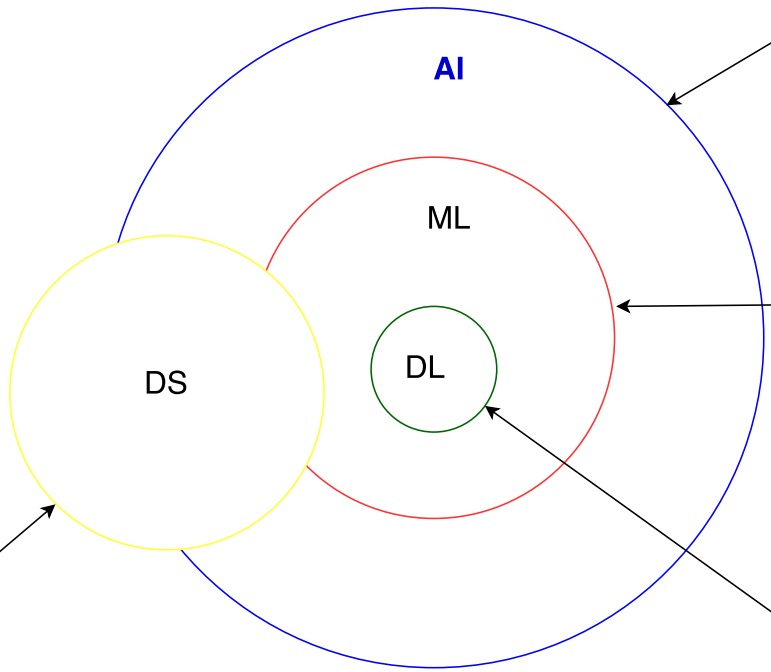
Learning and inference

- Automatically detect patterns in data and
- Build models that explain the world
- Use the uncovered pattern to understand what is happening (inference) and to predict what will happen(prediction).

Introduction to Machine Learning



AI vs ML vs Deep learning Vs Data science



Artificial Intelligence

- Enable computer to do think
- Old school AI (Simple rule systems) not used.

Machine learning

- Subset of AI that enable computer to learn from data.

Data Science

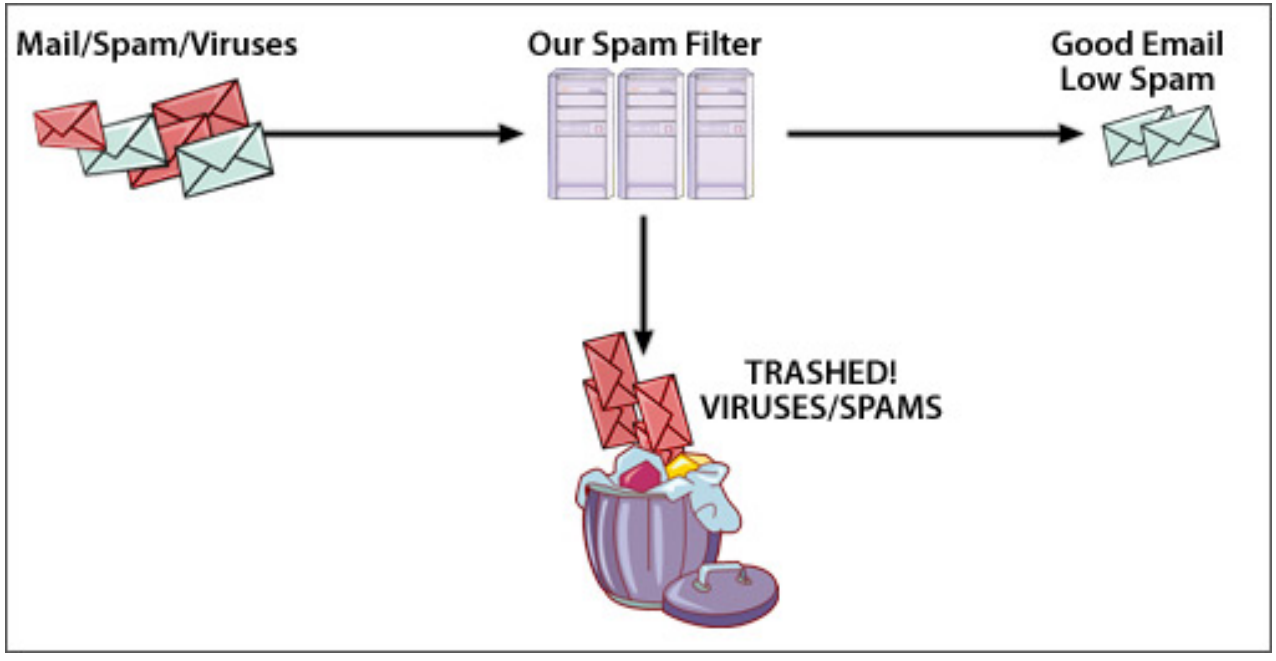
- Understating or making sense of data
- Visualisation, Data ingineering, Data products etc

Deep learning

- Subset of ML that use a cascade of multiple layers of nonlinear processing for pattern recognition and for feature learning.

Why use ML?

Consider how you would write a spam filter using traditional programming techniques.



Why use ML?

- Hard problems in high dimensions, like many modern CV or NLP problems require complex models $\Rightarrow$ difcult to program the correct behavior by hand.
- Machines can discover hidden, non-obvious patterns.
- A system might need to adapt to a changing environment.
- A learning algorithm might be able to perform better than its human programmers.

The scientific field of Machine Learning

There are many related terms, e.g. Pattern Recognition, Statistical Modelling, Statistical Learning, Data Mining, Adaptive Control, Data Analytics, Data Science, Artificial Intelligence, and Machine Learning.

Learning is clearly multidisciplinary, view from different fields

- Computer Science: Artificial Intelligence, information retrieval
- Statistics: Learning theory, data mining, learning and inference from data
- Engineering: Signal processing, system identification, adaptive and optimal control, computer vision/image processing, information theory, robotics, . . .
- Cognitive Science and Psychology: perception, mathematical psychology, computational linguistics, . . .
- Economics: Decision theory, game theory, operational research,

Machine Learning Vs Statistics

It's similar to statistics...

- Both fields try to uncover patterns in data.
- Both fields draw heavily on calculus, probability, and linear algebra, and share many of the same core algorithms.

But it's not statistics!

- Stats is more concerned with helping scientists and policymakers draw good conclusions; ML is more concerned with building autonomous agents.
- Stats puts more emphasis on interpretability and mathematical rigor; ML puts more emphasis on predictive performance, scalability, and autonomy.

Machine Learning Application

Automatic Colorization

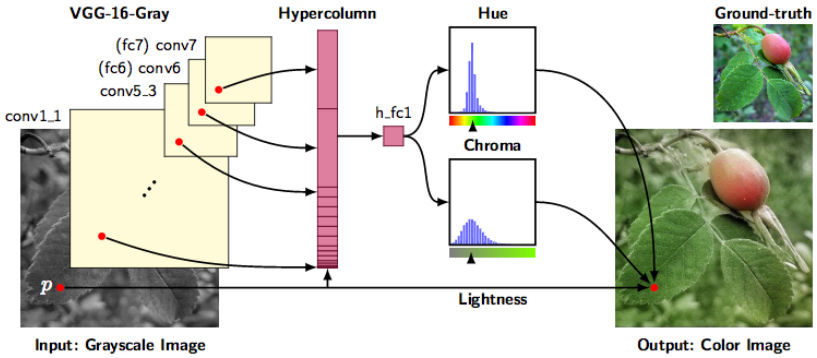


Figure 1: Automatic colorization

Image Captioning

Object Classification and Detection

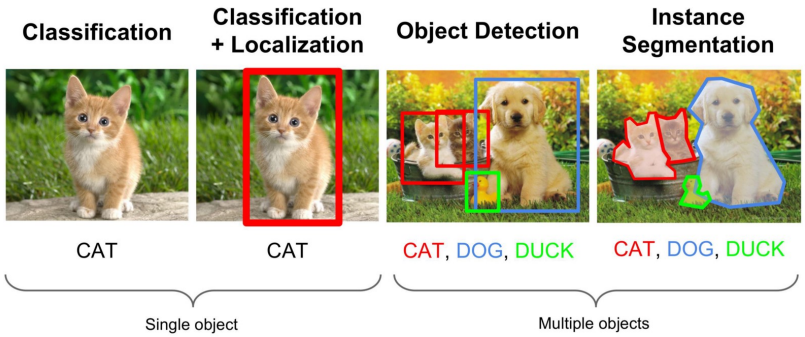
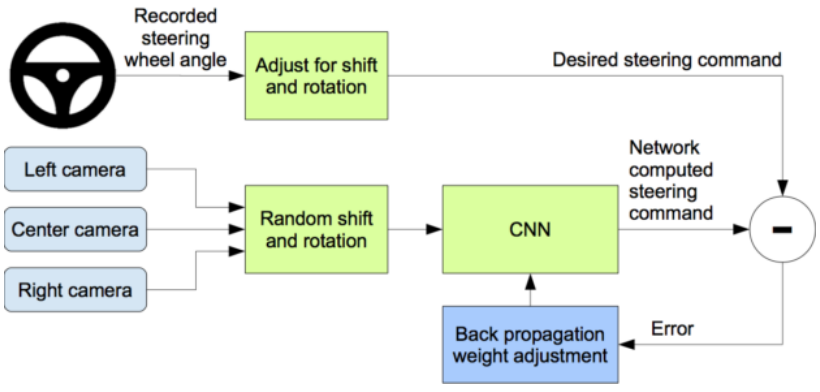


Figure 2: Object recognition

Image Style Transfer

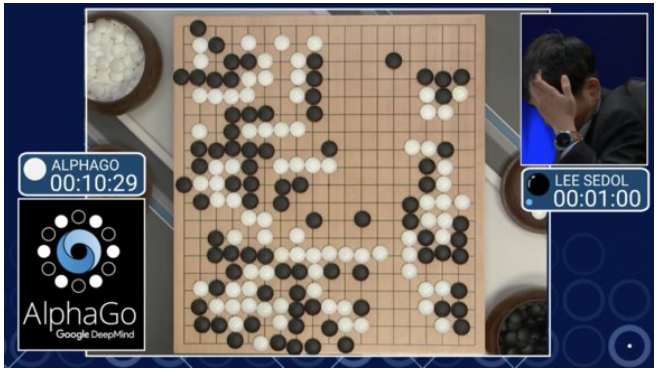
Machine Learning Application

Self driving car

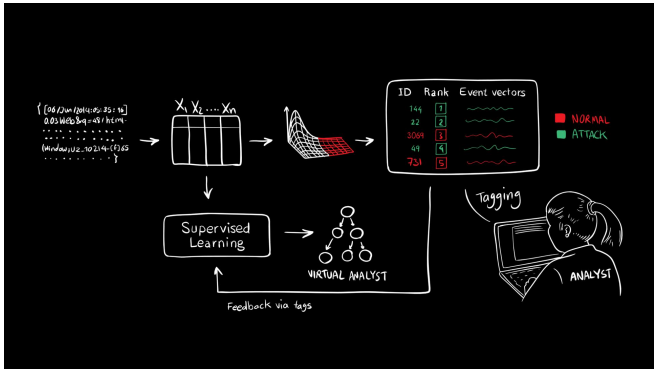


Drones

Game

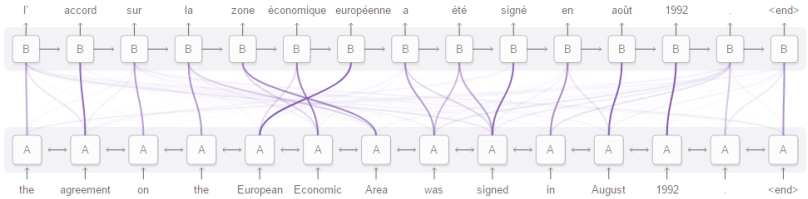


Cyber attack prediction

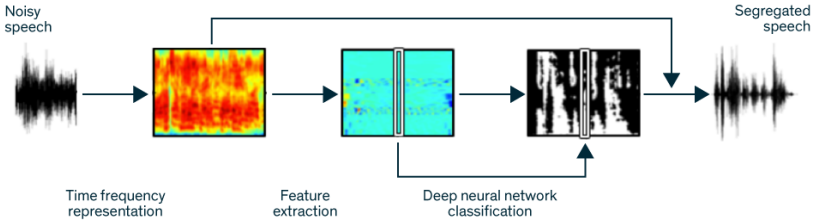


Machine Learning Application

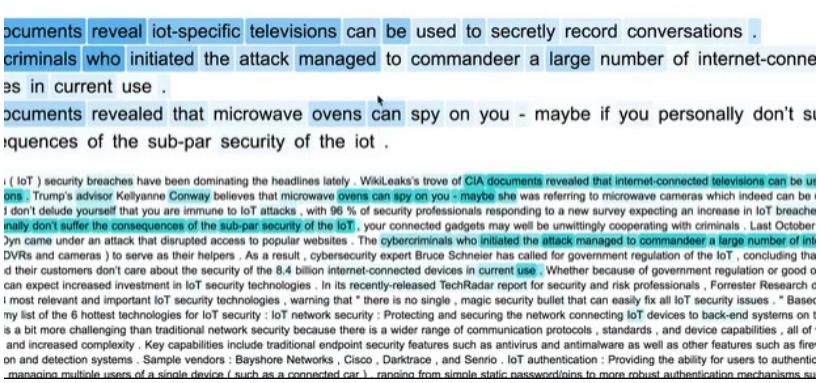
Machine translation



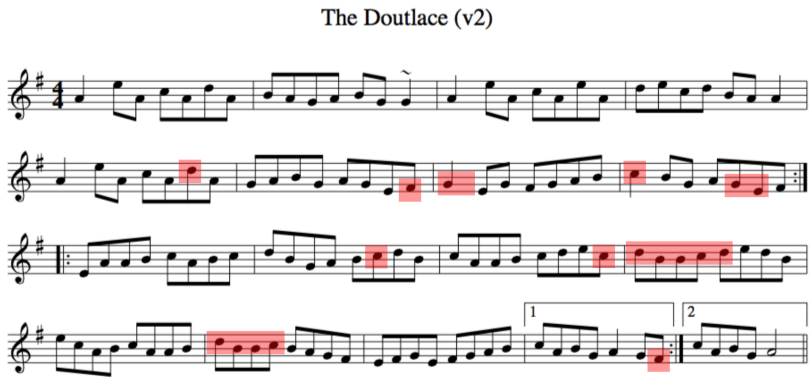
Speach Processing



Automatic Text Generation

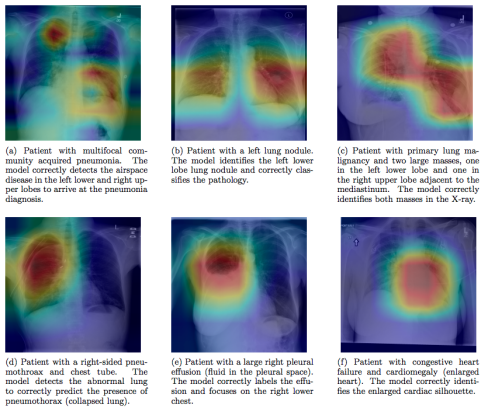


Music composition



Machine Learning Application

Pneumonia Detection on Chest X-Rays



(a) Patient with multifocal community acquired pneumonia. The model correctly detects the airspace disease in the left lower and right upper lobes to arrive at the pneumonia diagnosis.

(b) Patient with a left lung nodule. The model identifies the left lower lobe lung nodule and correctly classifies the pathology.

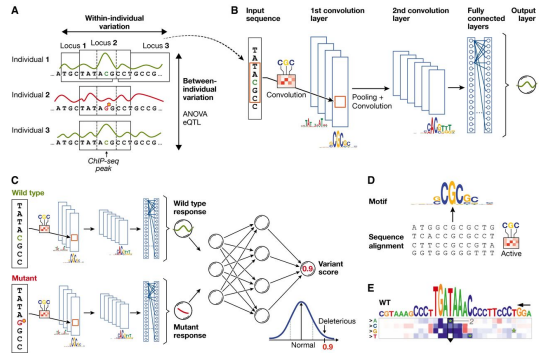
(c) Patient with primary lung malignancy and two large masses, one in the left lower lobe and one in the right upper lobe adjacent to the mediastinum. The model correctly identifies both masses in the X-ray.

(d) Patient with a right-sided pneumothroax and chest tube. The model detects the abnormal lung to correctly predict the presence of pneumothorax (collapsed lung).

(e) Patient with a large right pleural effusion (fluid in the pleural space). The model correctly labels the effusion and focuses on the right lower chest.

(f) Patient with congestive heart failure and cardiomegaly (enlarged heart). The model correctly identifies the enlarged cardiac silhouette.

Computational biology



Pedict heart disease risk from eye scans

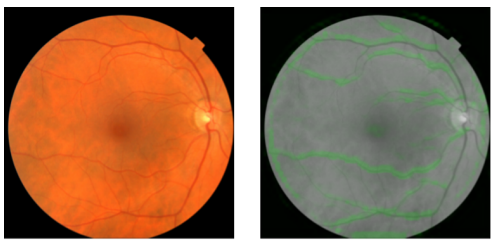
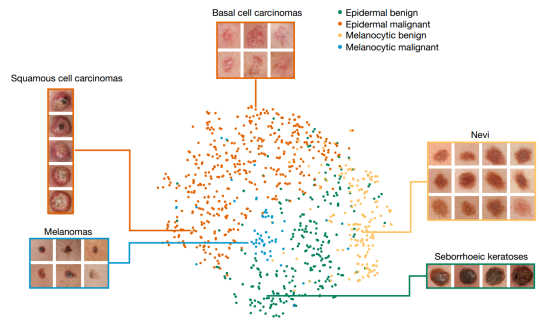


Image of retina

Blood pressure predictions focus on blood vessels

Diagnosis of Skin Cancer



More stories

Machine Learning Problem classes

Machine learning problems are usually divide into three major classes:

1. **Supervised Learning**
2. **Semi-Supervised Learning**
3. **Unsupervised Learning**
4. **Reiforcement Learning**

Machine Learning Problem classes: Supervised Learning

Supervised Learning: Learn a model from a given set of input-output pairs, in order to predict the output of new inputs.

- Further grouped into Regression and classification problems.
- Example: KNN, Logistic regression, random forest, neural network.

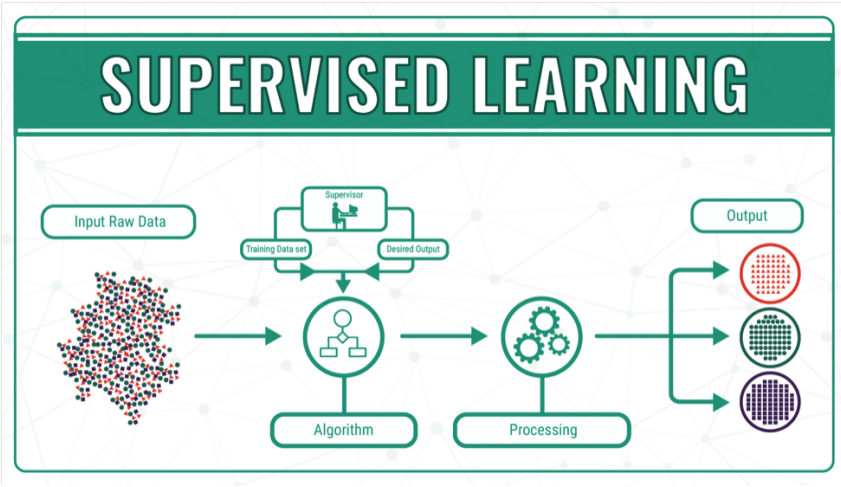


Figure 3: credit: *Shivam, pandey*

Machine Learning Problem classes: Unsupervised Learning

Unsupervised Learning: Discover patterns and learn the structure of unlabelled data.

- Clustering: analyzing and grouping data which does not include pre-labeled class or class attributes (K-means)
- Dimensionality reduction: represent data with fewer dimensions (PCA).
- Association :discovers the probability of the co-occurrence of items in a collection (Apriori).
- Example : Social Network Analysis, blog vistor segementation.

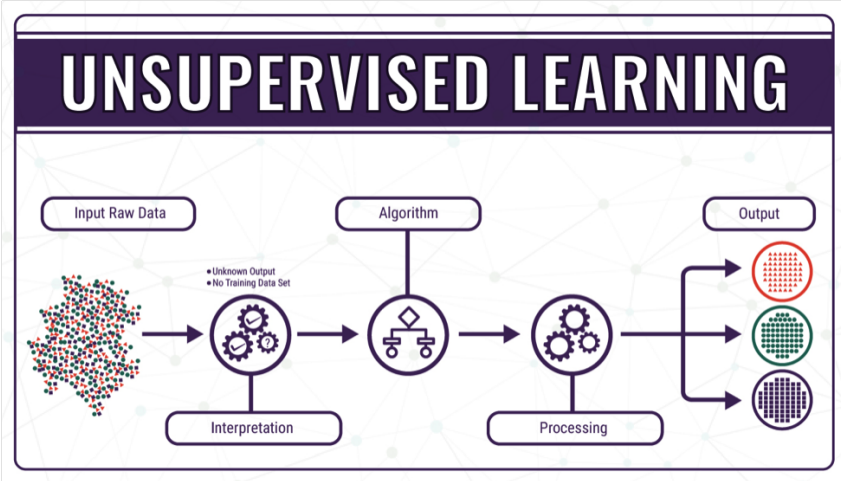
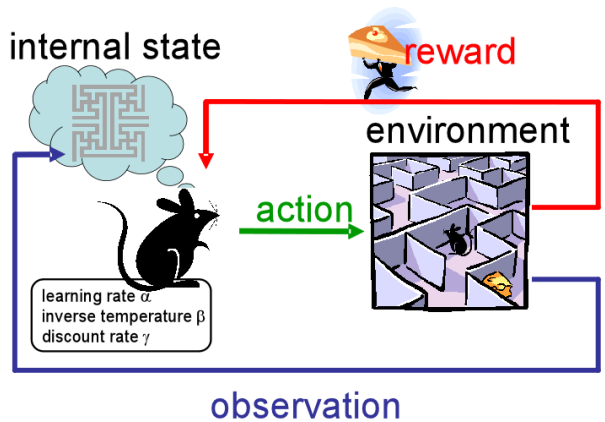


Figure 4: credit: *Shivam, pandey*

Machine Learning Problem classes: Reiforcement Learning

Reiforcement Learning: It is about taking suitable action to maximize reward in a particular situation.

- The learning system (an agent) observe the environment, select and perform actions, and get rewards (penalty) in return.
- The agent learn by itself what is the best strategy (policy) to get the most reward over time.
- A policy defines what action the agent should choose when it is in a given situation.



- Example: Traffic signal control, Robotics,

The three building blocks of ML

Machine learning is about methods allowing computers/machines to automatically make use of data to solve tasks.

- Data on its own is typically useless, it is only when we can extract knowledge/insights/pattern from the data that it becomes useful.
- A mathematical model with unknown parameters is a compact representation of the data.
- Learn the unknown parameters from the data.

The three building blocks of ML

1. Data: Typically we need lots of it.
2. Mathematical model:Compact representation of the data that cap- tures the key properties of the underlying situation $\Rightarrow$ Find and understand hidden structures from data.
3. Learning algorithm:Used to compute the unknown parameters from the observed data using the model.

The three building blocks of ML

Example: Classify plant disease from leaf image.

1. Data: Image data $\mathbf{x}$ and its corresponding label $\mathbf{y}$
2. Mathematical model: $p(\mathbf{y}|\mathbf{x}) = f_\theta(\mathbf{x};\theta)$
3. Learning algorithm: Find best value of θ that maximize $p(\mathbf{y}|\mathbf{x})$

Outline

Learning Theory: Log-likelihood function

Given model $p(\mathbf{y}|\mathbf{x};\theta) = f_\theta(\mathbf{x};\theta)$ characterized by an unknown parameter θ.

- $p(\mathbf{y}|\mathbf{x};\theta)$ is the like-hood function $\Rightarrow$ describe the ability of the parameter θ to describe the data

$$p(\mathbf{y}|\mathbf{x};\theta) = \prod_i^N p(y_n|x_n;\theta)$$

- Let define the log-likelihood function $\mathcal{L}(\theta)$: the log of the likelihood function.

$$\mathcal{L}(\theta) = \log p(\mathbf{y}|\mathbf{x};\theta) = \log \prod_i^m p(y_n|x_n;\theta)$$

$$= \sum_n^N \log p(y_n|x_n;\theta)$$

Learning Theory: Log-likelihood function

- For example for bernoulli model with i.i.d. observations

$$\mathcal{L}(\theta) = \sum_{n}^{N} \log \theta^{y_n} (1-\theta)^{1-y_n}$$

$$= \sum_{n}^{N} y_n \cdot \log \theta + (1-y_n) \log(1-\theta)$$

Parameter estimation methods

1. Maximum Likelihood Estimation (MLE)
2. Maximum A-Posteriori Estimation (MAP)

Learning Theory: MLE

Choose the parameters such that the observations are as likely as possible.

- Straightforward and natural way to learn parameters θ
- The parameter θ is selected such that it maximize $\mathcal{L}(\theta)$:

$$\theta = \arg\max_{\theta} \mathcal{L}(\theta) = \arg\min_{\theta} -\sum_{n}^{N} \log p(y_n|x_n;\theta)$$

  where θ is assumed to be fixed point (point-estimation).
- Easy to observe overfitting of parameters.

Learning Theory: MAP

MAP is the point estimation of parameter θ with some known prior distribution $p(\theta)$

- To use MAP, we need to specify two distributions.
  1. The prior distribution $p(\theta)$, which encodes our beliefs about the parameters before we observe the data.
  2. The likelihood $p(\mathbf{y}|\mathbf{x};\theta)$, same as in maximum likelihood
- With this two distributions, we define the posterior distribution $p(\theta|\mathcal{D})$ which correspond to uncertainty about θ after observing the data given by: where $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}$

$$p(\theta|\mathcal{D}) = \frac{p(\mathbf{y}|\mathbf{x};\theta) \cdot p(\theta)}{P(\mathcal{D})}$$

$$p(\theta|\mathcal{D}) \propto p(\mathbf{y}|\mathbf{x};\theta) \cdot p(\theta) = p(\theta, \mathcal{D})$$

Learning Theory: MAP

Thus MAP estimation: find the most likely parameter settings under the posterior.

$$\begin{aligned}\mathcal{L}(\theta) &= \log p(\theta, \mathcal{D}) = \log p(\mathbf{y}|\mathbf{x};\theta) \cdot p(\theta \\ &= \sum_{n}^{N} \log p(y_n|x_n;\theta) + \sum_{n}^{N} \log p(\theta) \\ &= \sum_{n}^{N} \log p(y_n|x_n;\theta) + \lambda \mathcal{R}(\theta)\end{aligned}$$

Regularisation is essential to overcome the limitations of maximum likelihood estimation.

Formulate a learning problem

To formulate learning problem mathematically, you need:

1. Data: set of labeled training examples $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\}$
2. Model (Hypothesis)
   - A model is a set of allowable functions $f_\theta(\mathbf{x}; \theta)$ that compute predictions $\hat{\mathbf{y}}$ from the inputs $\mathbf{x} \Rightarrow$ map inputs $\mathbf{x}$ to outputs $\mathbf{y}$ parameterized by θ.
   - Given hypothesis $f_\theta(\mathbf{x}; \theta)$ we define Loss function/objective function $\mathcal{L}_\theta(f(\mathbf{x}; \theta), \mathbf{y})$
   - $\mathcal{L}_\theta(f(\mathbf{x}; \theta), \mathbf{y})$ defines how well the model $f(\mathbf{x}; \theta)$ fit the data.
3. Learning algorithm
   - Find the model parameters θ that best fit the data.
   - Objective: minimize a cost function J_θ with respect to the model parameters θ

$$\arg\min_\theta J_\theta = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_\theta(\hat{y}^{(i)}, y^{(i)}) \tag{1}$$

   - A procedure to compute gradient $\frac{\partial J_\theta}{\partial \theta}$

Learning algorithm: A procedure to compute gradientt

Backpropagation: a procedure that is used to compute gradients of a loss function.

- It is based on the application of the chain rule and computationally proceeds 'backwards'.

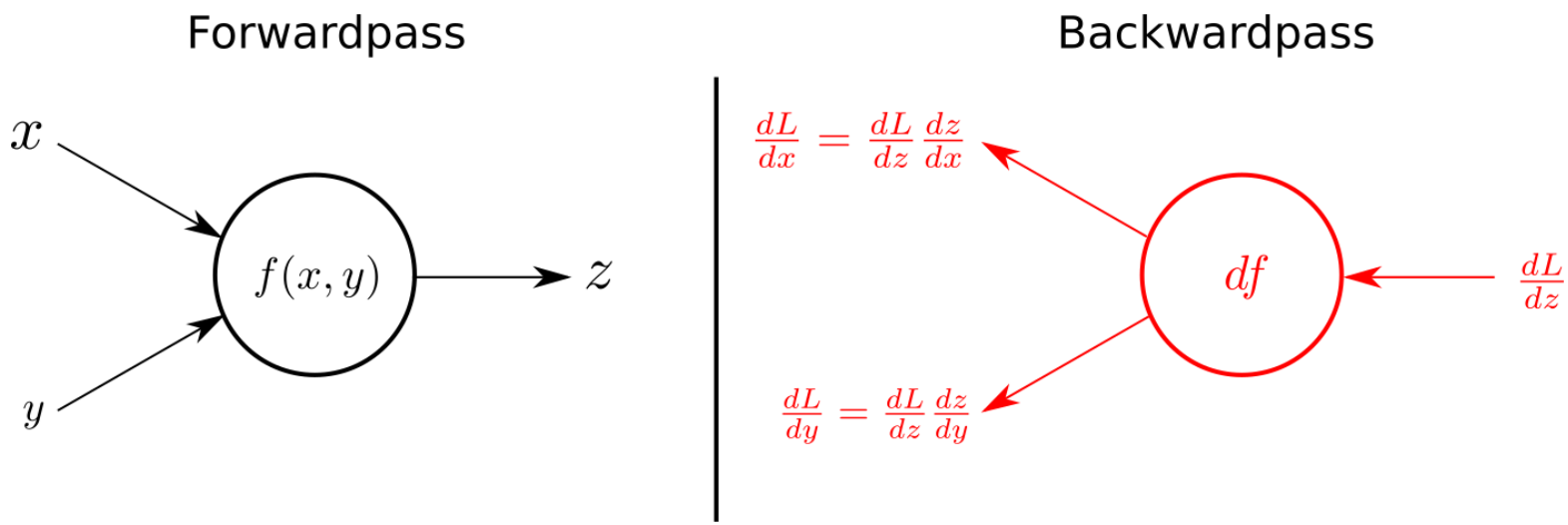
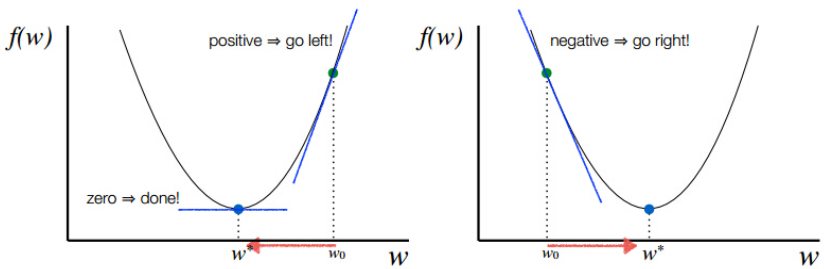


Figure 5: Back propagation: `credit: Flair of Machine Learning`

Learning algorithm: Gradient descent

Gradient descent: procedure to minimize a objective function.

- compute gradient
- take step in opposite direction



α is the learning rate $\rightarrow$ determine size of step we take to reach local minimum.

1. Initilize parameter θ,
2. Loop until converge,
   1. Compute gradient:

$$\frac{\partial J_\theta}{\partial \theta}$$

   2. Update parameters:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial J_\theta}{\partial \theta}$$

3. Retrn parameter θ

Limitation: Take time to compute

Learning algorithm: Stochastic Gradient Descent (SGD)

SGD consists of updating the model parameters θ after every sample.

SGD

Initialize θ randomly.

For each training example:

- Compute gradients: $\frac{\partial J_{i\theta}}{\partial \theta}$
- Update parameters θ with update rule:

$$\theta^{(t+1)} := \theta^{(t)} - \alpha \frac{\partial J_{i\theta}}{\partial \theta}$$

Stop when reaching criterion

Easy to compute $\frac{\partial J_{i\theta}}{\partial \theta}$ but very noise.

Learning algorithm: Mini-batch SGD training

Make update based on a min-batch B of example instead of single example i

Mini-batch SGD

1. Initialize θ randomly.
2. For each mini-batch B:
   - Compute gradients: $\frac{\partial J_{B\theta}}{\partial \theta} = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial J_{k(\theta)}}{\partial \theta}$
   - Update parameters θ with update rule: $\theta^{(t+1)} := \theta^{(t)} - \alpha \frac{\partial J_{B\theta}}{\partial \theta}$
3. Stop when reaching criterion

Fast to compute $\frac{\partial J_{\theta}}{\partial \theta} = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial J_{k(\theta)}}{\partial \theta}$ and much better estimate of the true gradient.

Standard procedure for training machine learning.

Formulate a learning problem: Learning algorithm

Setting the learning rate α

- Small learning rate: Converges slowly and gets stuck in false local minima.
- Large learning rate: Overshoot became unstable and diverge.
- Stable learning rate: Converges smoothly and avoid local minima.

How to deal with this ?

1. Try lots of different learning rates and see what works for you.
2. Use an adaptive learning rate that adapts to the landscape of your objective function. $\Rightarrow$ Adam, SGD with momentum.

$$v^{(t+1)} := \rho v^{(t)} - \frac{\partial J_{B\theta}}{\partial \theta}$$
$$\theta^{(t+1)} := \theta^{(t)} - \alpha v^{(t+1)}$$

Outline

Data: Irrelevant Features

Consider relevant features (inputs) → features that are correlated with prediction.

- ML systems will only learn efficiently if the training data contain enough relevant features.

The process of identifying relevant feature from data is called Feature engineering.

Feature Engineering

Involves:

- Feature selection: selecting the most useful features to train on among existing features
- Feature extraction: combining existing features to produce a more useful one (e.g Dimension reduction)

Overfitting and Underfitting

Central ML challenge: ML algorithm must perform well on new unseen inputs ⇒ Generalization.

- When training the ML model on training set we measure training error.
- When testing the ML model on test set we measure test error (generalization error) ⇒ should be low as possible.

The performance of ML models depends on these two factors:

1. generation error ⇒ small is better.
2. the gap between generalization error and train error.

Overfitting and Underfitting

Overfitting (variance): Occur when the gap between training error and test error is too large.

- The model performs well on the training data, but it does not generalize well.

Underfitting (bias): Occur when the model is not able to obtain sufficiently low error value on training set.

- Excessively simple model

Both underfitting and overfitting lead to poor predictions on new data and they do not generalize well.

Overfitting and Underfitting: Variance-Bias Tradeoff

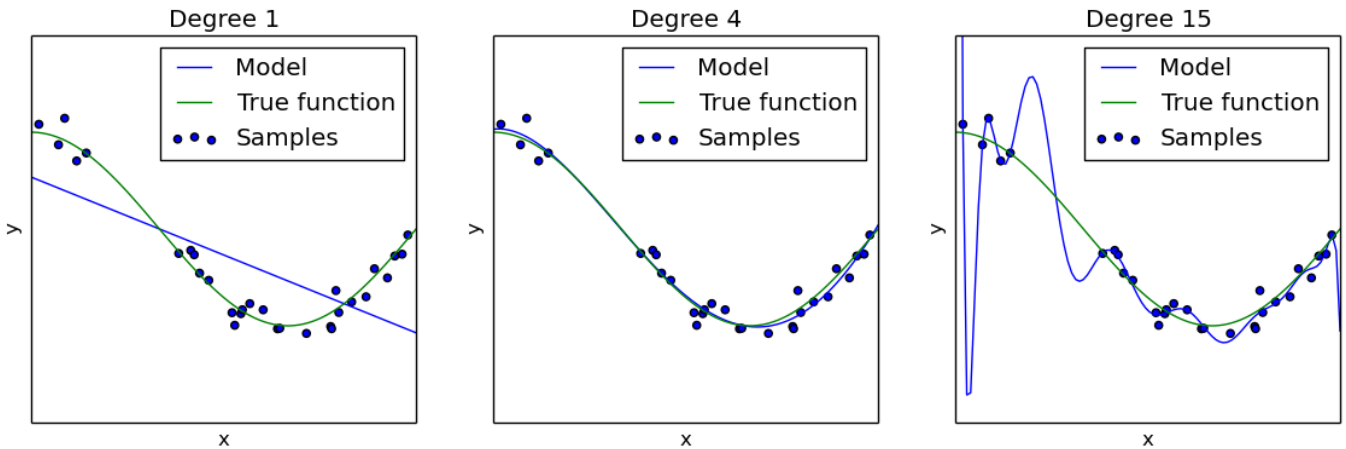


Figure 6: Overfitting vs Underfitting: `credit: scikit-learn.org`

Overfitting and Underfitting

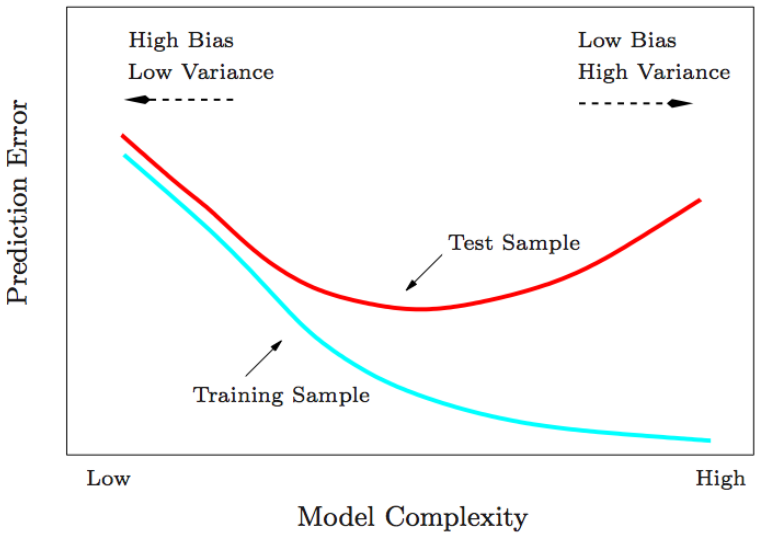


Figure 7: Model complexity: `credit: Gerardnico`

Variance-Bias Tradeoff

- For high bias, we have a very simple model.
- For the case of high variance, the model become complex.

To control bias and variance

1. Reduce number of features
2. Alter the capacity of the model (regularization).

Overfitting and Underfitting: Regularization

Regularization: Reduces overfitting by adding a complexity penalty to the loss function.

$$\mathcal{L}_{reg}(\hat{y}, y) = \mathcal{L}(\hat{y}, y) + \lambda \mathcal{R}(\mathbf{w})$$

where:

- $\lambda \geq 0$ is regularization strength.
- $\mathcal{R}(\mathbf{w})$ is the regularization functions which is defined as:

Simple example

$\mathcal{R}(\mathbf{w}) = \mathbf{w}^2$ for L2 regulization

$\mathcal{R}(\mathbf{w}) = |\mathbf{w}|$ for L1 regulization

More complex

Dropout, Batch normalization, Layer normalization etc

Outline

Evaluation protocols

Learning algorithms require the tuning of many meta-parameters (Hyper-parameters).

Hyper-parameters

Hyper-parameter: a parameter of a model that is not trained (specified before training)

- Have a strong impact on the performance, resulting in over-fitting through experiments.
- We must be extra careful with performance estimation.
- The process of choosing the best hyper-parameters is called Model selection.

Evaluation protocols

The best practise is to spilit your data into three disjoint sets.

Train set

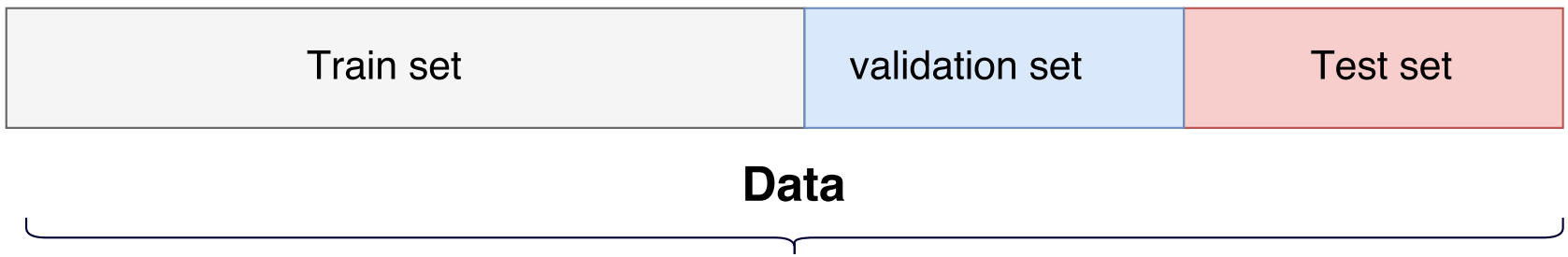
- Used for learning

Validation set

- Used for model selection

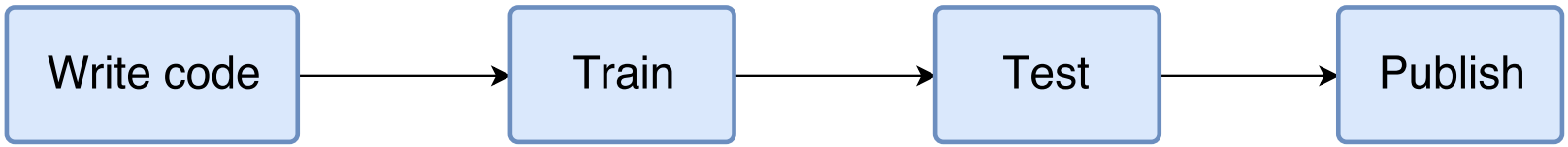
Test set

- Used for estimating generalization performance

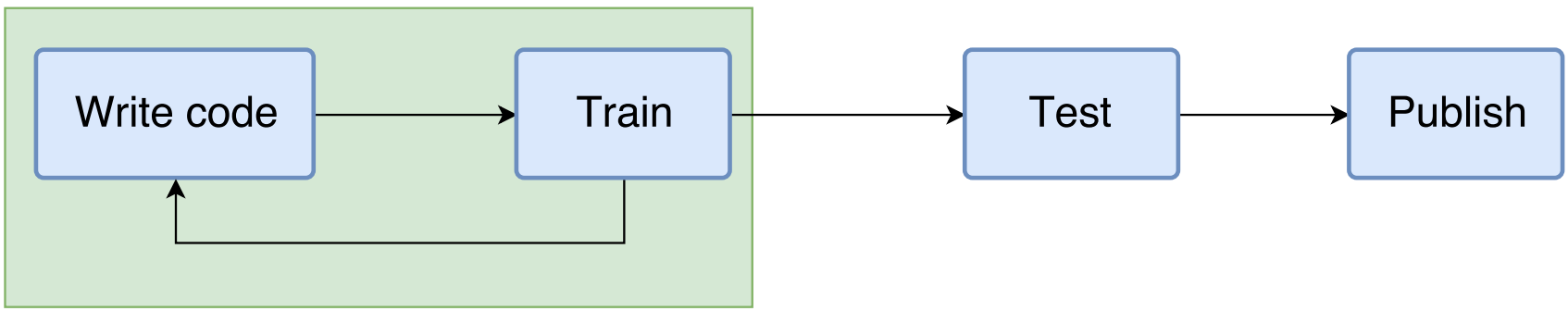


Evaluation protocols: Development cycle

The ideal development cycle is



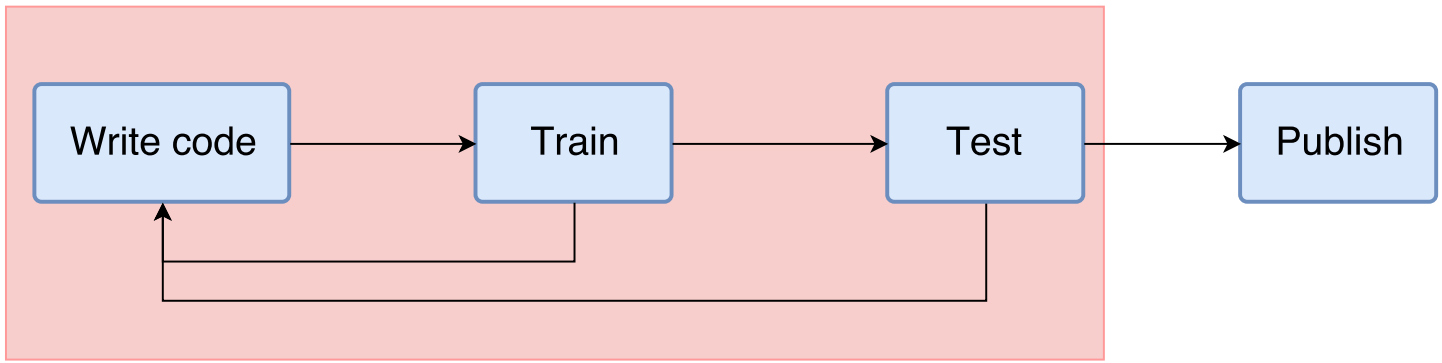
or in practice something like



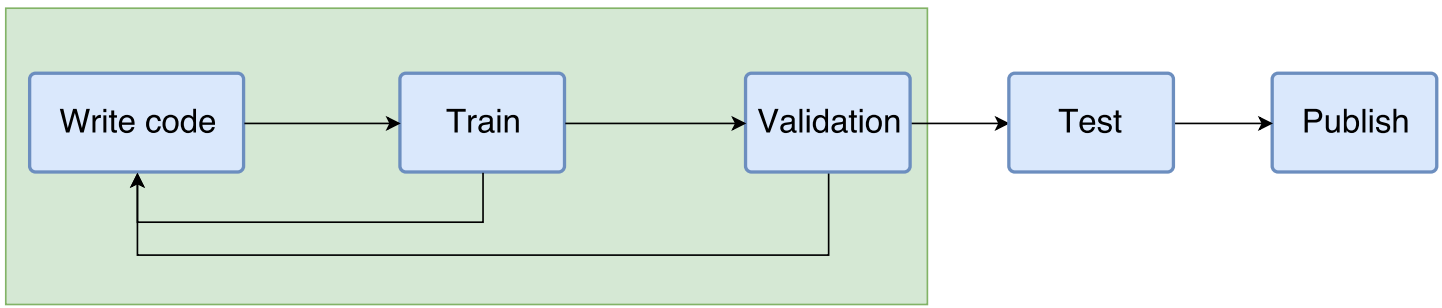
- There may be over-fitting, but it does not bias the final performance evaluation.

Evaluation protocols: Development cycle

Unfortunately, it often looks like



- This should be avoided at all costs.
- The standard strategy is to have a separate validation set for the tuning.



Evaluation protocols: Cross validation

Cross validation: Statistical method for evaluating how well a given algorithm will generalize when trained on a specific data set.

- Used to estimate the performance of learning algorithm with less variance than a single train-test set split.
- In cross validation we split the data repetedely and train a multiple models.

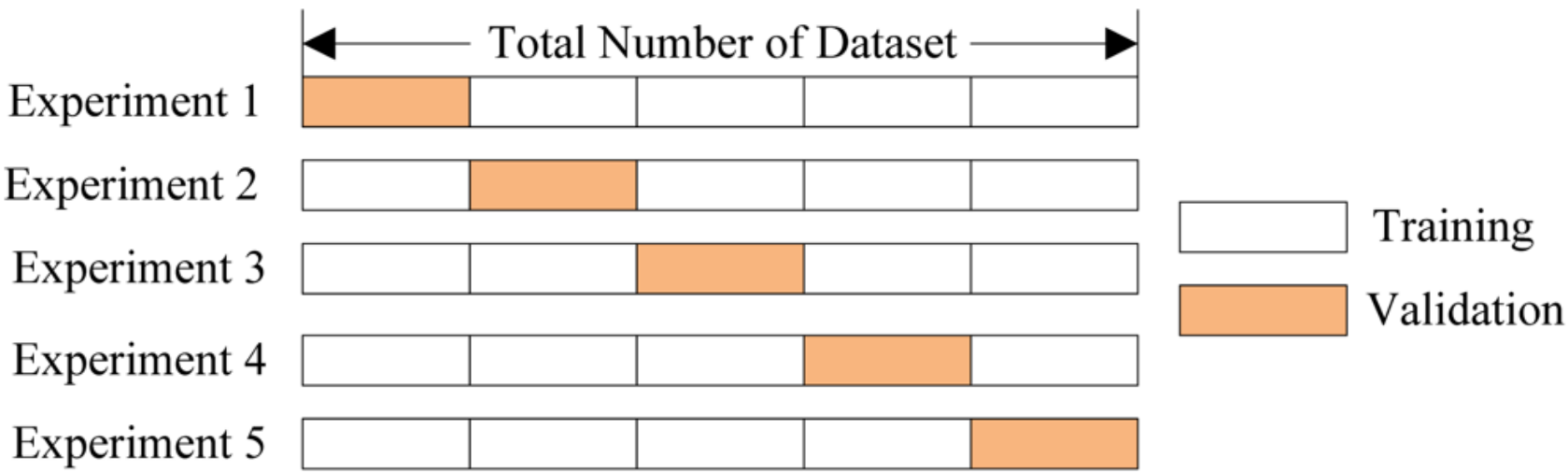


Figure 8: Cross validation: `credit: kaggle.com`

Evaluation protocols: Cross validation Types

K-fold cross validation

- It works by splitting the dataset into k-parts (e.g.k=3, k=5 or k=10).
- Each split of the data is called a fold.

Figure 9: K-Fold: `credit: Juan Buhagiar`

Startified K-fold cross validation

- The folds are selected so that the mean response value is approximately equal in all the folds.

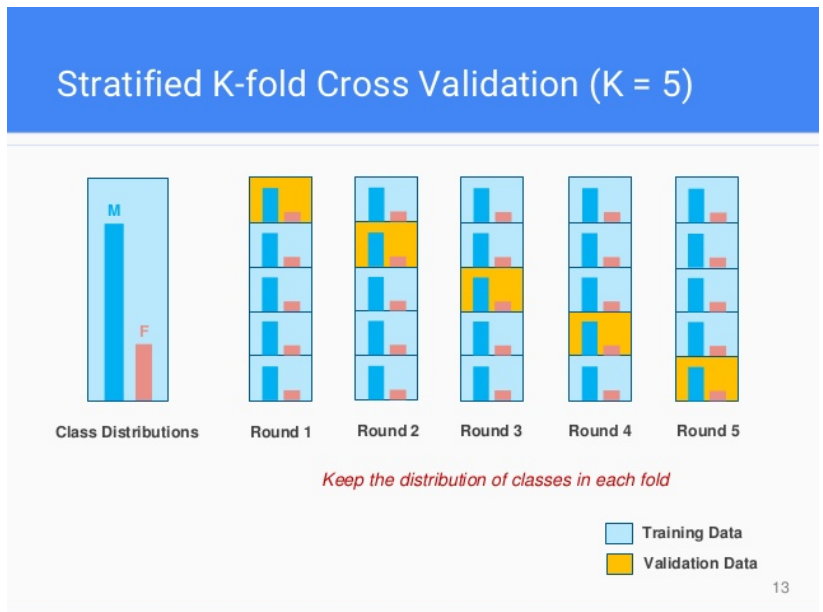


Figure 10: SK-Fold: `credit: Mark Peng`

Performance metrics

How to measure the performance of a trained model?

Many options availabe: depend on type of problem at hand.

- `Classification`: Accuracy, Precision, Recall, Confusion matrix etc.
- `Regression`: RMSE, Explained variance score, Mean absolute error etc.
- `Clustering`: Adjusted Rand index, inter-cluster density etc.

Example: scik-learn metrics

Outline

Regression

Regression: predict a scalar-valued target, such as the price of stock.

1. weather forecasting.
2. house pricing prediction.
3. student performance prediction.
4. ....
5. ....

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Figure 11: dataset.

Regression: Linear regression

In linear regression, the model consists of linear functions given by:

$$f(\mathbf{x};\theta) = \sum_j w_j x_j + b = \mathbf{w}^T\mathbf{x} + \mathbf{b}$$

where w is the weights, and b is the bias.

The loss function is given by:

$$\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

The cost function:

$$\begin{aligned} J_\theta &= \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\ &= \frac{1}{2N}\sum_{i=1}^{N}\left(\sum_j w_j x_j^{(i)} + b - y^{(i)}\right) \end{aligned}$$

Formulate a learning problem: Classification

Goal is to learn a mapping from inputs x to target y such that $y \in \{1 \dots k\}$ where k is the number of classes.

- If $k = 2$, this is called binary classification.
- If $k > 2$, this is called multiclass classification.
- If each instance of x is associated with more than one label to each instance, this is called multilabel classification.

Examples

- predict whether a patient has a disease, given the presence or absence of various symptoms
- classify e-mails as spam or non-spam
- predict whether a financial transaction is fraudulent

Classification: Logistic regression

Goal is to predict the binary target class $y \in \{0, 1\}$.

Model is given by:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

where

$$z = \mathbf{w}^T \mathbf{x} + \mathbf{b}$$

This function squashes the predictions to be between 0 and 1 such that:

$$p(y = 1 \mid x, \theta) = \sigma(z)$$

and

$$p(y = 0 \mid \mathbf{x}, \theta) = 1 - \sigma(\mathbf{z})$$

Loss function: it is called crossentropy and defined as:

$$\mathcal{L}_{CE}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$$

Multi-class Classification

What about classification tasks with more than two categories?

- Targets form a discrete set $\{1, ..., k\}$.
- It's often more convenient to represent them as indicator vectors, or a one-of-k encoding:

Model: softmax function

$$\hat{y}_k = softmax(z_1 \dots z_k) = \frac{e^{z_k}}{\sum_k e^{z_k}}$$

where

$$z_k = \sum_j w_{kj} x_j + b$$

Loss Function: cross-entropy for multiple-output case

$$\begin{aligned} \mathcal{L}_{CE}(\hat{y}, y) &= -\sum_{k=1}^{K} y_k \log \hat{y}_k \\ &= -\mathbf{y}^{\mathbf{T}} \log \hat{\mathbf{y}} \end{aligned}$$

Outline

What is Deep Learning

Deep Learning a subclass of machine learning algorithms that learn underlying features in data using multiple processing layers with multiple levels of abstarction.

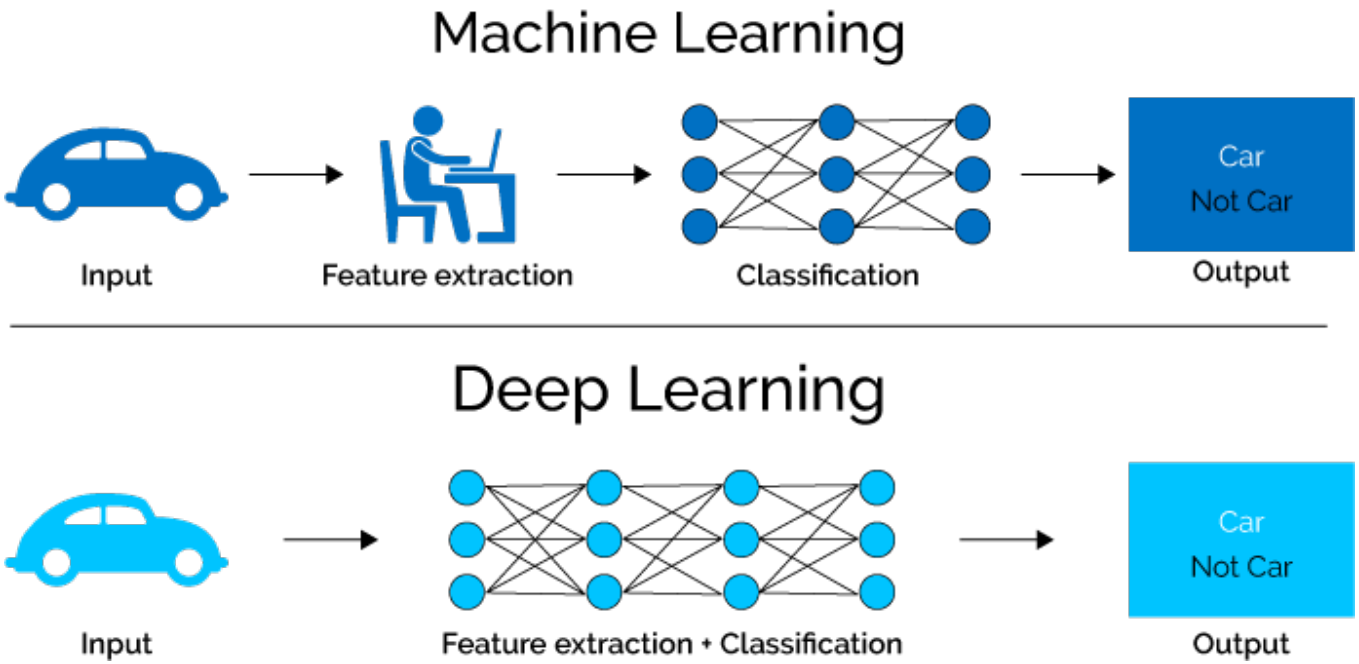


Figure 12: ML vs Deep learning: *credit:*

Why Deep Learning and why now?

Why deep learning: Hand-Engineered Features vs. Learned features.

Traditional ML

- Use enginered feature to extract useful patterns from data.
- Complex and difficult since different data sets require different feature engineering approach

Deep learning

- Automatically discover and extract useful pattern from data.
- Allows learning complex features e.g speach and complex networks.

Why Deep Learning and why now?

Why Now?

Big data availability

- Large datasets
- Easier collection and storage

Increase in computaional power

- Modern GPU architecture.

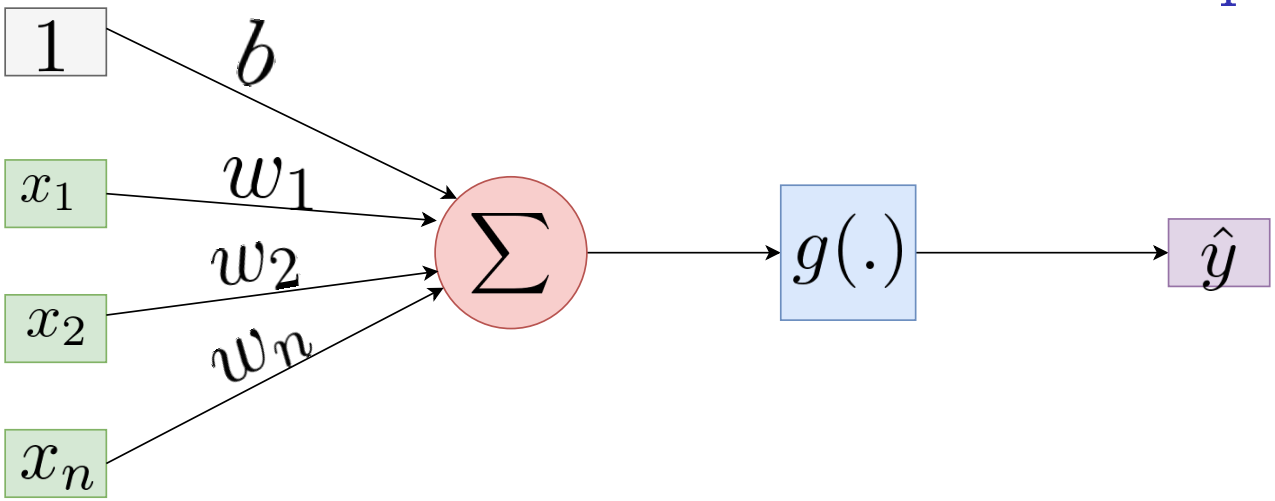
Improved techniques

- Five decades of research in machine learning.

Open source tools and models

- Tensorflow.
- Pytorch
- Keras

Generalized Linear Function: The Perceptron



$$z(x) = b + \sum_{i=1}^{n} w_i x_i$$

$$\mathbf{z} = \mathbf{w}^T \mathbf{x} + b$$

$$p(y|\mathbf{x}) = \mathbb{E}[y] = p(y|g(\mathbf{z}); \theta)$$

- The basic function $\mathbf{z}$ can be any linear function.
- $g(.)$ is an inverse link function (activation function).
- $\mathbf{z}$ is pre-activation.

The Perceptron: Activation Function

Why Activation Functions?

- Activation functions add non-linearity properties to neuro network function.
- Most real-world problems + data are non-linear.
- Activation function need to be differentiable.

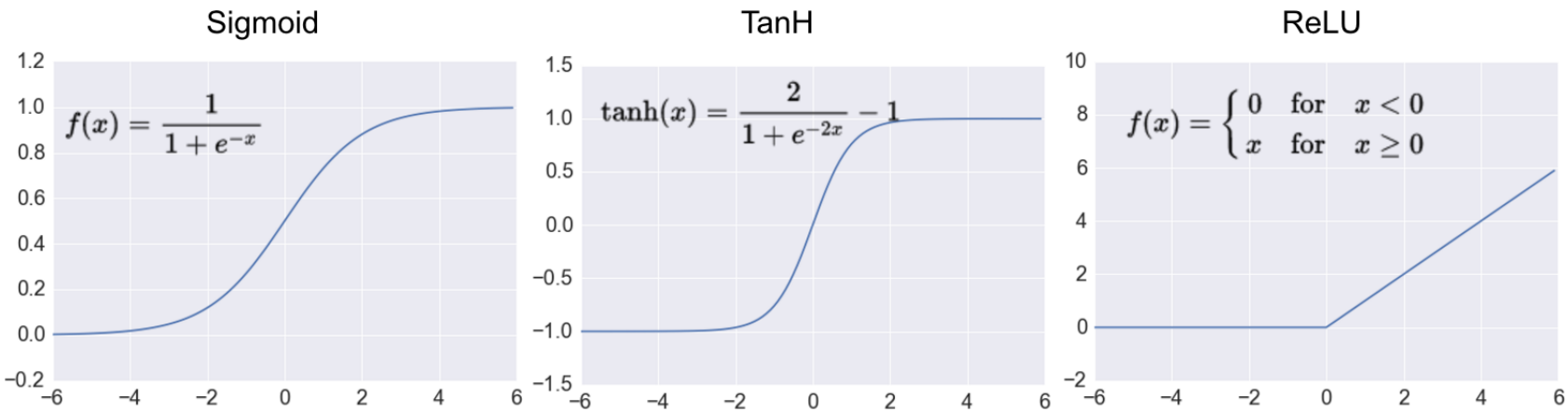
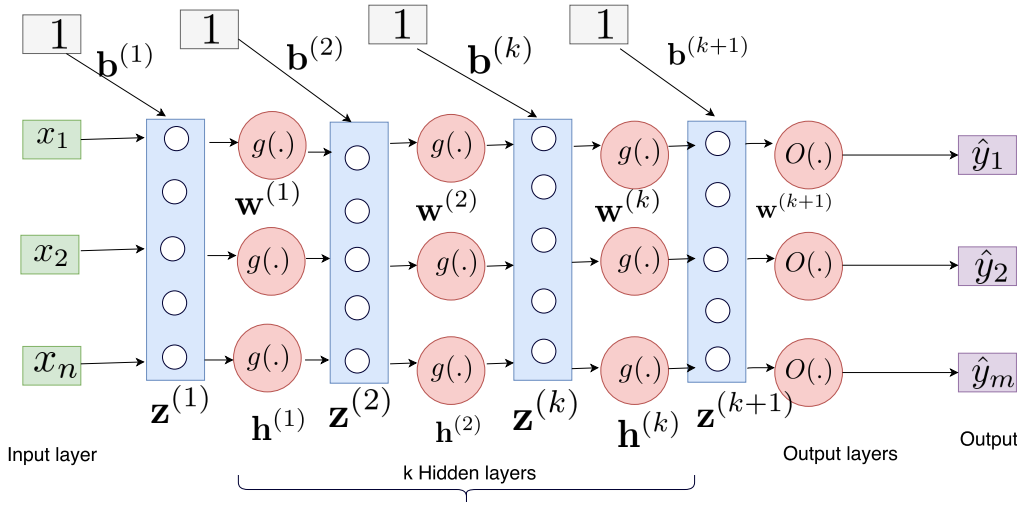


Figure 13: Activation function `credit:kdnuggets.com`

Multilayer Perceptrons (MLP)



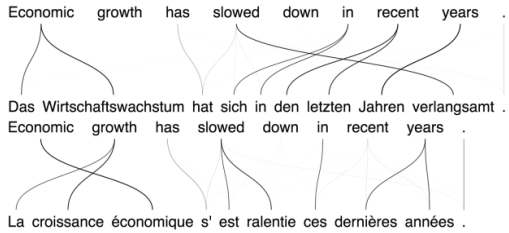
- Consists of L multiple layers $(l_1, l_2 \ldots l_L)$ of pecepron, interconnected in a feed-forward way.
- The first layer l_1 is called the input layer $\Rightarrow$ just pass the information to the next layer.
- The last layer is the ouput layer $\Rightarrow$ maps to the desired output format.
- The intermediate k layers are hidden layers $\Rightarrow$ perform computations and transfer the weights from the input layer.

Limitation of MLP

Consider the following problems.

Problem 1: Speach recognition

Problem 2:Machine translation



Problem 3: Object recognition.

(a) Image 1

(b) Image 2

Figure 14: Classify the image as zebra regardless of the orientation of zebra in the image.

Deep Neural Networks (MLP)

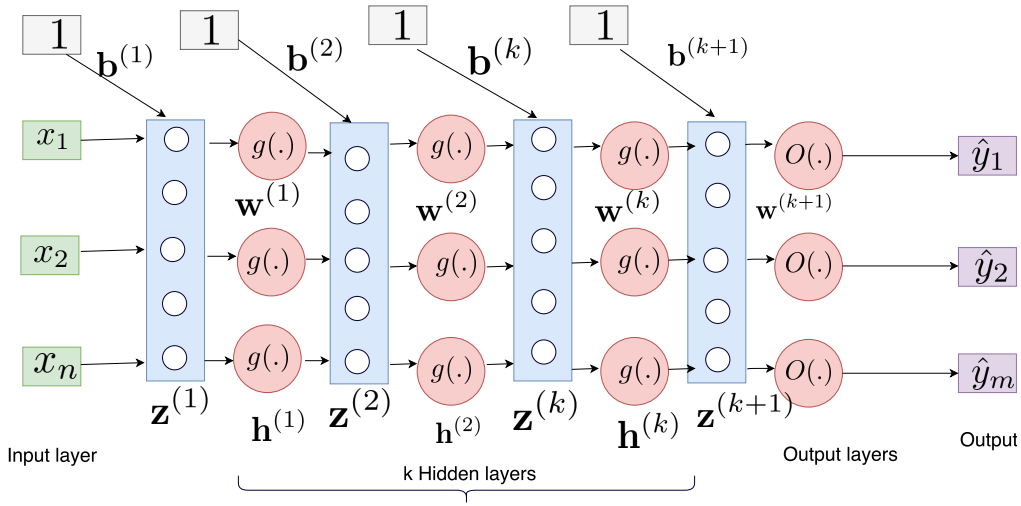
Deep networks

- Recursively compose the basic linear functions (the perceptron).
- Give a deep neural network.

$$p(y|\mathbf{x}) = \mathbb{E}[y] = p(y|\mathbf{h}^1(g(\mathbf{z})) \cdot \mathbf{h}^2(g(\mathbf{h}^1) \dots \mathbf{h}^k(g(\mathbf{h}^{k-1})); \theta)$$

- A general, flexible framework for building non-linear, parametric models.

Multilayer Perceptrons (MLP)



Hidden layer 2

- Activation

$$\begin{aligned}\mathbf{h^{(2)}}(\mathbf{x}) &= g(\mathbf{z^{(2)}}(\mathbf{x})) \\ &= g(\mathbf{b^{(2)}} + \mathbf{w^{(2)}}\mathbf{h^{(1)}}(\mathbf{x}))\end{aligned}$$

- Pre-activation

$$\mathbf{z^{(3)}}(\mathbf{x}) = \mathbf{b^{(3)}} + \mathbf{w^{(3)}}\mathbf{h^{(2)}}(\mathbf{x})$$

Hidden layer k

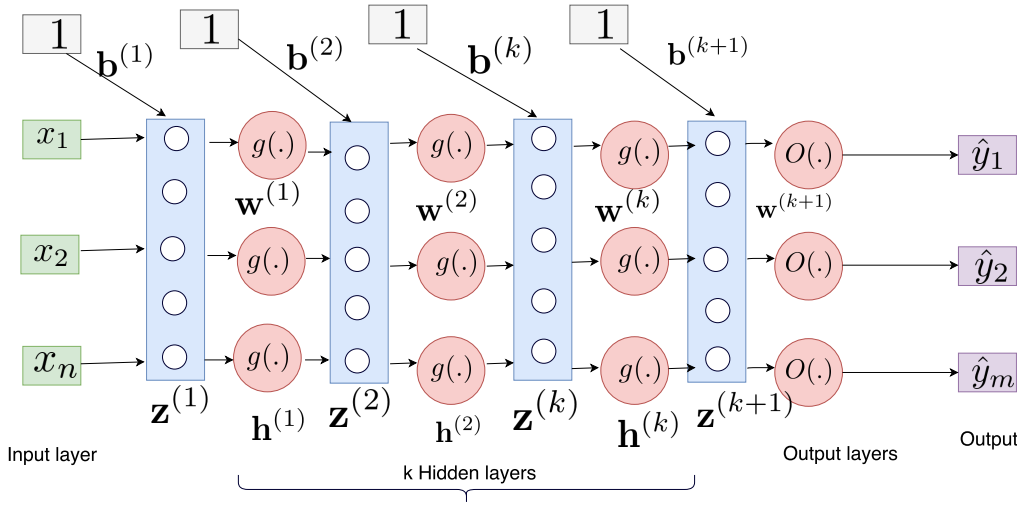
- Activation

$$\begin{aligned}\mathbf{h^{(k)}}(\mathbf{x}) &= g(\mathbf{z^{(k)}}(\mathbf{x})) \\ &= g(\mathbf{b^{(k)}} + \mathbf{w^{(k)}}\mathbf{h^{(k-1)}}(\mathbf{x}))\end{aligned}$$

- Pre-activation

$$\mathbf{z^{(k+1)}}(\mathbf{x}) = \mathbf{b^{(k+1)}} + \mathbf{w^{(k+1)}}\mathbf{h^{(k)}}(\mathbf{x})$$

Multilayer Perceptrons (MLP)



Output layer

- Activation

$$\begin{aligned}\mathbf{h^{(k+1)}}(\mathbf{x}) &= O(\mathbf{z^{(k+1)}}(\mathbf{x})) \\ &= O(\mathbf{b^{(k+1)}} + \mathbf{w^{(k+1)}}\mathbf{h^{(k)}}(\mathbf{x})) \\ &= \hat{\mathbf{y}}\end{aligned}$$

where $O(.)$ is output activation function

Output activation function

- Binary classification: $y \in \{0, 1\} \Rightarrow sigmoid$
- Multiclass classfiction:$y \in \{0, K-1\} \Rightarrow softmax$
- Regression: $y \in \mathbb{R}^n \Rightarrow$ identity sometime RELU.

Demo Playground

Limitation of MLP

MLP as universal function aproximator has limitation for such problems.

- Take fixed sized input and generates fixed sized output
- Require a very large network.
- MLPs are sensitive to the location of the pattern $\Rightarrow$ Does not share feature learned across different position.
- It treat every example independently $\Rightarrow$ does not care about what happened at previous time step

Modern Deep learning Architecture: Convolutional Neural Network

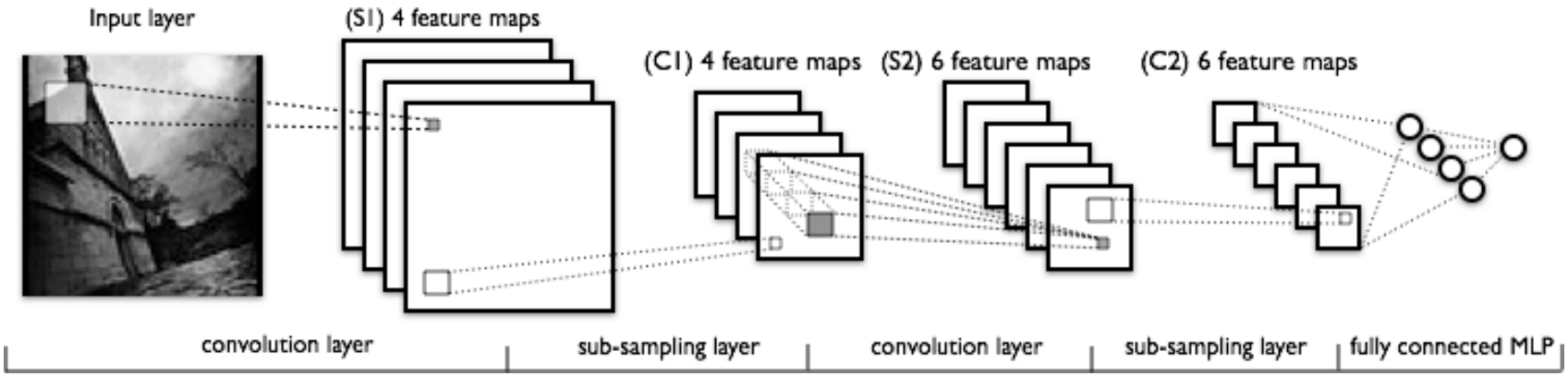


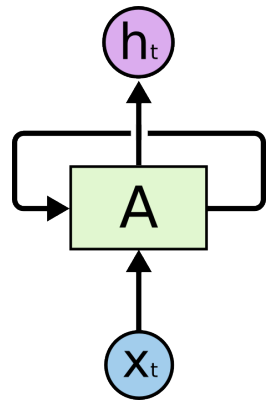
Figure 15: CNN [credit:deeplearning.net]

- Enhances the capabilities of MLP by inserting convolution layers.
- Composed of many "filters", which convolve, or slide across the data, and produce an activation at every slide position
- Suitable for spatial data, object recognition and image analysis.

Deep learning Architecture: Recurrent Neural Networks (RNN)

RNN: family of neural network for handling sequential data.

- Each output is a function of the previous output with the same update rule applied.



$$h^{(t)} = f_\theta(h^{(t-1)}, x_t)$$

- Can model a long time dimension and arbitrary sequence of events and inputs.
- Suitable for sequenced data analysis: time-series, sentiment analysis, NLP, language translation, speech recognition etc.
- Common variants: LSTM and GRUs

Deep learning Architecture: Auto-enceoder

Autoenceoder:A neural network where the input is the same as the output.

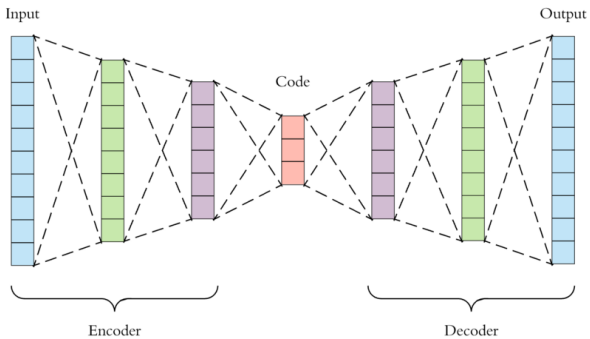


Figure 16: `credit:Arden Dertat`

- They compress the input into a lower-dimensional code and then reconstruct the output from this representation.
- It is an unsupervised ML algorithm similar to PCA.
- Several types exist: Denoising autoencoder, Sparse autoencoder.

Deep learning Architecture: Auto-enceoder

Autoencoder consists of components: encoder, code and decoder.

- The encoder compresses the input and produces the code,
- The decoder then reconstructs the input only using this code.

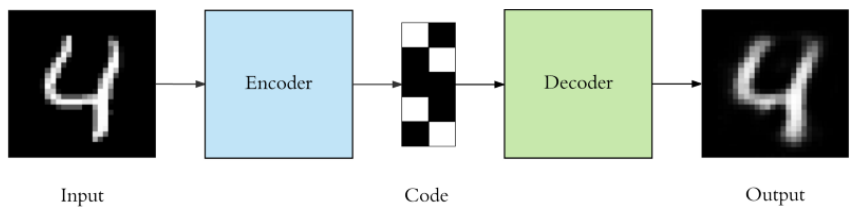


Figure 17: `credit:Arden Dertat`

Deep learning Architecture: Deep Generative models

Idea:learn to understand data through generation $\rightarrow$ replicate the data distribution that you give it.

- Can be used to generate Musics, Speach, Langauge, Image, Handwriting, Language
- Suitable for unsupervised learning as they need lesser labelled data to train.

Two types:

1. Autoregressive models: Deep NADE, PixelRNN, PixelCNN, WaveNet, ByteNet
2. Latent variable models: VAE, GAN.

Limitation

- Very data hungry (eg. often millions of examples)
- Computationally intensive to train and deploy (tractably requires GPUs)
- Poor at representing uncertainty (how do you know what the model knows?)
- Uninterpretable black boxes, difficult to trust
- Difficult to optimize: non-convex, choice of architecture, learning parameters
- Often require expert knowledge to design, fine tune architectures

Research Direction

- Transfer learning.
- Unsepervised machine learning.
- Computational efficiency.
- Add more reasoning (uncertatinity) abilities $\Rightarrow$ Bayesian Deep learning
- Many applications which are under-explored especially in developing countries.

Outline

Deep learning in Practice: Regularization

Regularization: Technique to help deep learning network perform better on unsee data.

- Constraints optimization problem to discourage complex model.

$$\arg\max_{\theta} \frac{1}{N} \sum_{i} \mathcal{L}(f(\mathbf{x}^{(i)} : \theta), \mathbf{y}^{(i)}) + \lambda \mathcal{R}(\theta)$$

- Improve generalization of deep learning model.

Deep learning in Practice: Activation Function

- Use ReLU, Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU
- Try out tanh but don't expect much
- Don't use sigmoid

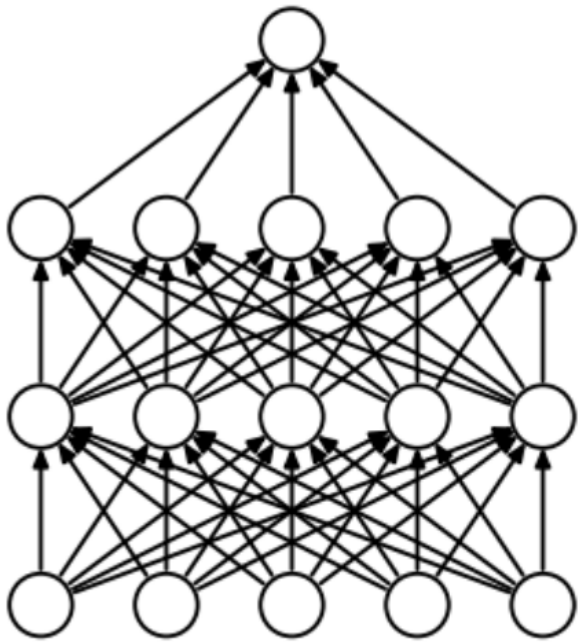
Deep learning in Practice: Optimisation

- Adam is a good default choice in many cases
- SGD+Momentum with learning rate decay often outperforms Adam by a bit, but requires more tuning

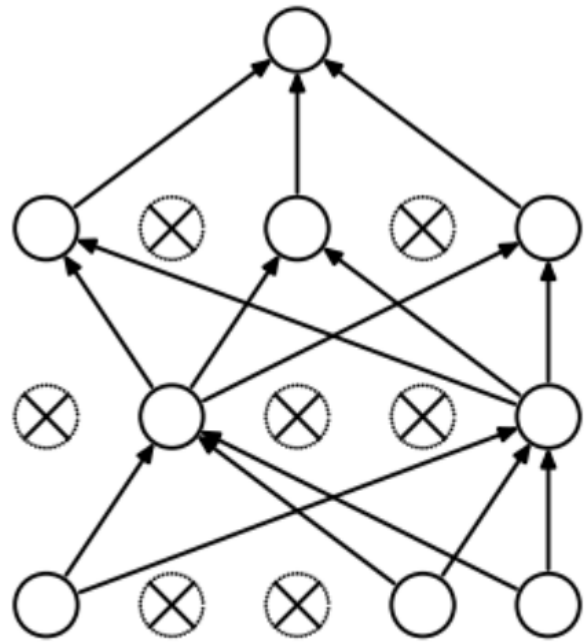
Deep learning in Practice: Regulisation-1 Dropout

Dropout: Randomly remove hidden unit from a layer during training step and put them back during test.

- Each hidden unit is set to 0 with probability p.
- Force network to not rely on any hidden node $\Rightarrow$ prevent neural net from ovefitting (improve performance).
- Any dropout probability can be used but 0.5 usually works well.

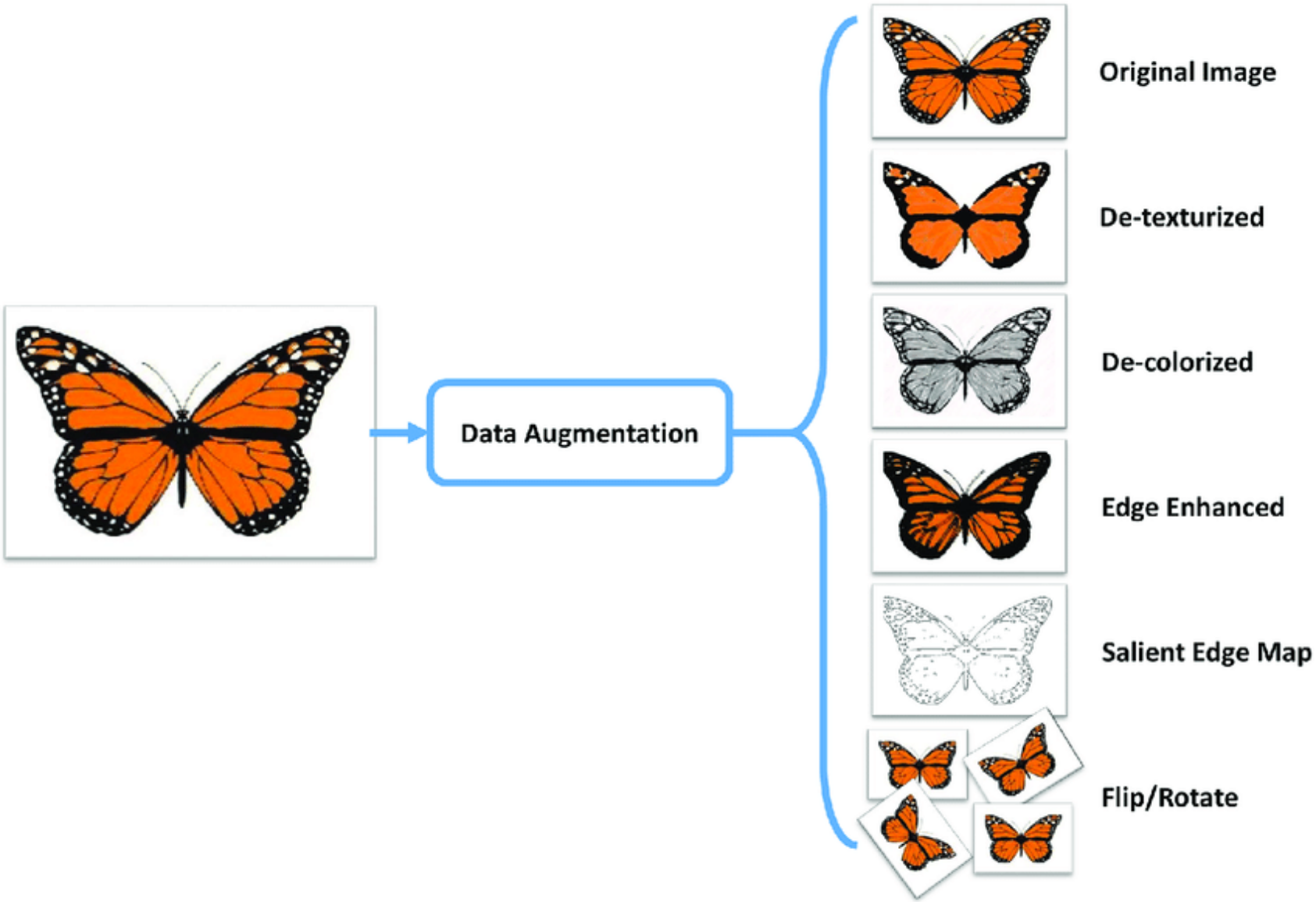


(a) Standard Neural Net



(b) After applying dropout.

Deep learning in Practice-Regulisation-2 Data Augementation



Deep learning in Practice-Regulisation-3: Early Stopping

Early Stopping: Stop training before the model overfit.

- Monitor the deep learning training process from overfiting.
- Stop training when validation error increases.

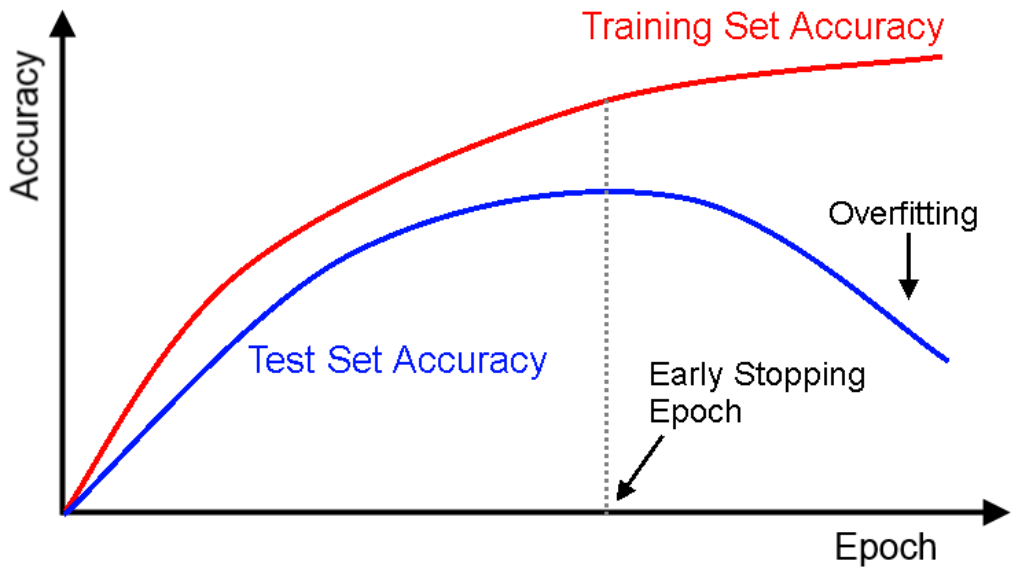


Figure 18: Early stopping: `credit: Deeplearning4j.com`

Deep learning in Practice: Batch Normalization

Batch normalisation: A technique for improving the performance and stability of deep neural networks.

Training deep neural network is complicated

- The input of each layer changes as the parameter of the previous layer change.
- This slow down the training ⇒ require low learning rate and careful parameter initilization.
- Make hard to train models with saturation non-linearity.
- This phenomena is called Covariate shift

To address covariate shift ⇒ normalise the inputs of each layer for each mini-batch (Batch normalization)

- To have a mean output activation of zero and standard deviation of one.

Deep learning in Practice: Batch Normalization

Batch normalisation: A technique for improving the performance and stability of deep neural networks.

Training deep neural network is complicated

- The input of each layer changes as the parameter of the previous layer change.
- This slow down the training ⇒ require low learning rate and careful parameter initilization.
- Make hard to train models with saturation non-linearity.
- This phenomena is called Covariate shift

To address covariate shift ⇒ normalise the inputs of each layer for each mini-batch (Batch normalization)

- To have a mean output activation of zero and standard deviation of one.

Deep learning in Practice: Batch Normalization

If $x_1, x_2, \ldots x_B$ are the sample in the batch with mean $\hat{\mu}_b$ and variance $\hat{\sigma}_b^2$.

- During training batch normalization shift and rescale each component of the input according to batch statistics to produce output y_b:

$$y_b = \gamma \odot \frac{x_b - \hat{\mu}_b}{\sqrt{\hat{\sigma}_b^2 + \epsilon}} + \beta$$

  where
- $\odot$ is the Hadamard component-wise product.
- Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

Limitation of Batch Normalisation

- Puts a lower limit on the batch size.
- Difficult to apply to recurrent connections.

Other alternative: Weight normalization and Layer normalization.

Deep learning in Practice: Weight Initilization

Before training the neural network you have to initialize its parameters.

Set all the initial weights to zero

- Every neuron in the network will computes the same output $\Rightarrow$ same gradients.
- Not recommended

Deep learning in Practice: Weight Initilization

Random Initilization

- Initilize your network to behave like zero-mean standard gausian function.

$$w_i \sim \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{1}{n}}\right)$$

$$b_i = 0$$

where n is the number of inputs.

Deep learning in Practice: Weight Initilization

Random Initilization: Xavier initilization

- Initilize your network to behave like zero-mean standard gausian function such that

$$w_i \sim \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{1}{n_{in} + n_{out}}}\right)$$

$$b_i = 0$$

where n_{in}, n_{out} are the number of units in the previous layer and the next layer respectively. where n is the number of inputs.

Deep learning in Practice: Weight Initilization

Random Initilization: Kaiming

- Random initilization that take into account ReLU activation function.

$$w_i \sim \mathcal{N}\left(\mu = 0, \sigma = \sqrt{\frac{2}{n}}\right)$$

$$b_i = 0$$

  - Recommended in practise.

Deep learning in Practice: Transfer learning

Transfer learning: ML method where a model developed for a task (source task) is reused as the starting point for a model on a second task (target task).

- Define source and target domain.
- Learn on source domain.
- Generalize on target domain $\Rightarrow$ Learned knowledge from source domain applied to a target domain.
- why it work: some low-level features can be shared for different tasks.

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

Deep learning in Practice: Transfer learning

When to use transfer learning

- source task and target task have the same input.
- There are lot of data for source task and relatively small amount of data for target task.
- Low level feature of source task could be helpful for target task.

Deep learning in Practise: Babysitting the Learning Process

Step 1: Preprocess the data

- Spend time cleaning up your data ⇒ remove errors, outliers, nose and handle missing data.
- Explore your data: visualize and identify potential correlations between inputs and outputs or between input dimensions.
- Transform non-numerical data: on-hot encoding, embending etc.

ID	Gender
1	Male
2	Female
3	Not Specified
4	Not Specified
5	Female

ID	Male	Female	Not Specified
1	1	0	0
2	0	1	0
3	0	0	1
4	0	0	1
5	0	1	0

Figure 19: on-hot encoding `credit: Adwin Jahn`

Deep learning in Practise: Babysitting the Learning Process

Step 1: Preprocess the data Normalization Make your features consistent $\Rightarrow$ easy for ML algorithm to learn.

1. Centering: Move your dataset so that it centered around the origin.

$$\mathbf{x} \leftarrow (\mathbf{x} - \mu)/\sigma$$

2. Scaling: rescale your data such that each feature has maximum absolute of 1.

$$\mathbf{x} \leftarrow \frac{\mathbf{x}}{\max |\mathbf{x}|}$$

3. scikit-learn example

Deep learning in Practise: Babysitting the Learning Process

Step 2: Choose the architecture and train your model

- Start with simple architecture.
- Double check that the loss is reasonable.
- Make sure that you can overfit very small portion of the training data.
- Start with small regularization and find learning rate that makes the loss go down.
  - Loss barely changing: Learning rate is probably too low.
  - Loss exploding: learning rate too high.
  - Rough range for learning rate we should be cross-validating is somewhere [1e-3 ... 1e-5].
- Monitor and visualize the loss curve.

Deep learning in Practise: Hyperparameter Optimization

- First stage: only a few epochs to get rough idea of what params work
- Second stage: longer running time, finer search
- ... (repeat as necessary)
- Random sample hyperparams, in log space when appropriate

$$lr = 10^{uniform(-3,-6)}$$

Hyperparameters to play with:

- network architecture
- learning rate, its decay schedule, update type
- regularization (L2/Dropout strength)

Machine Learning Work Flow

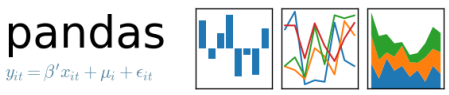
ML workflow sketch:

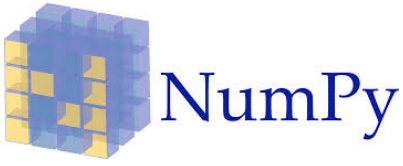
1. Should I use ML on this problem? Is there a pattern to detect? Can I solve it analytically? Do I have data?
2. Gather and organize data.
3. Preprocessing, cleaning, visualizing.
4. Establishing a baseline.
5. Choosing a model, loss, regularization, ...
6. Optimization (could be simple, could be a Phd...).
7. Hyperparameter search.
8. Analyze performance and mistakes, and iterate back to step 5 (or 3)

Implementing machine learning systems

- You will often need to derive an algorithm (with pencil and paper), and then translate the math into code.
- Array processing
  - vectorize computations (express them in terms of matrix/vector operations) to exploit hardware efficiency.

Python ML learning libraries





References I

- Intro to Neural Networks and Machine Learning:csc321 2018, Toronto University.
- Deep learning in Pytorch, Francois Fleurent: EPFL 2018.
- Machine Learning course by Andrew Ng: Coursera.