Research and Innovation Center of Excellence

MSc on Intelligent Critical Infrastructure Systems

Machine Learning
Lecture 14

Christos Kyrkou
Research Lecturer
KIOS Research and Innovation Center of Excellence
University of Cyprus

Imperial College London

University of Cyprus

funded by:

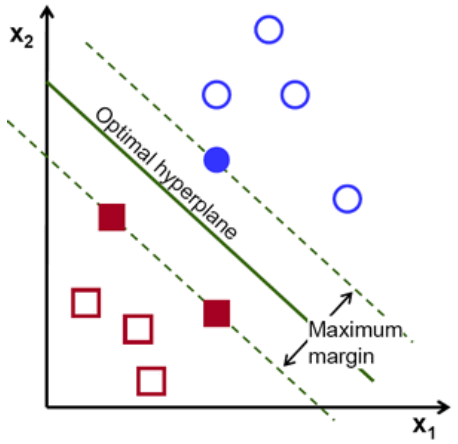
Machine learning tasks

- Supervised learning
  - regression: predict numerical values
  - classification: predict categorical values, i.e., labels
- Unsupervised learning
  - clustering: group data according to "distance"
  - association: find frequent co-occurrences
  - link prediction: discover relationships in data
  - data reduction: project features to fewer features
- Reinforcement learning
- Recently: Self-Supervised Learning

Machine learning algorithms

- Regression:
  Ridge regression, Support Vector Machines, Random Forest, Multilayer Neural Networks, **Deep Neural Networks**, ...

- Classification:
  Naive Base, , Support Vector Machines, Random Forest, Multilayer Neural Networks, **Deep Neural Networks**, ...



- Clustering:
  k-Means, Hierarchical Clustering, ...

ML in a nutshell

- Tens of thousands of machine learning algorithms
  - Hundreds new/variations every year

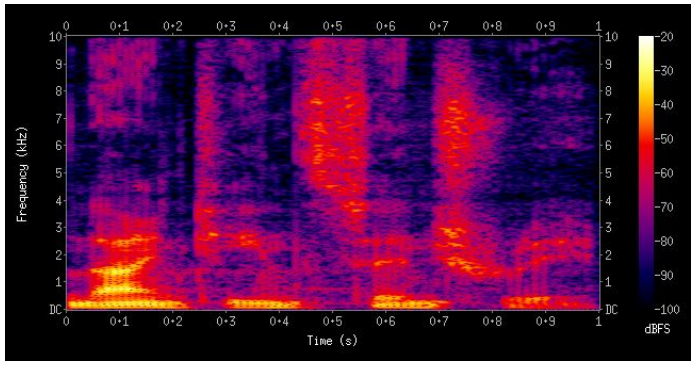
- **Objective function:** encodes the right loss for the problem
- **Parameterization:** makes assumptions that fit the problem
- **Regularization:** right level of regularization for amount of training data
- **Training algorithm:** can find parameters that maximize objective on training set
- **Inference algorithm:** can solve for objective function in evaluation

Various Data Types

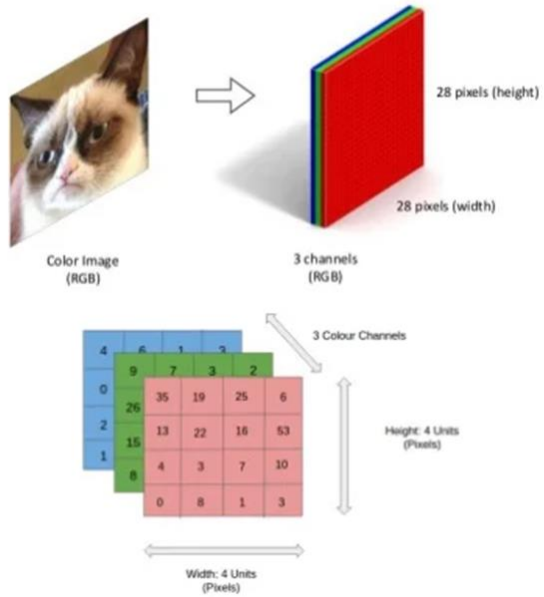
	living being	feline	human	gender	royalty	verb	plural
cat →	0.6	0.9	0.1	0.4	−0.7	−0.3	−0.2
kitten →	0.5	0.8	−0.1	0.2	−0.6	−0.5	−0.1
dog →	0.7	−0.1	0.4	0.3	−0.4	−0.1	−0.3
houses →	−0.8	−0.4	−0.5	0.1	−0.9	0.3	0.8
man →	0.6	−0.2	0.8	0.9	−0.1	−0.9	−0.7
woman →	0.7	0.3	0.9	−0.7	0.1	−0.5	−0.4
king →	0.5	−0.4	0.7	0.8	0.9	−0.7	−0.6
queen →	0.8	−0.1	0.8	−0.9	0.8	−0.5	−0.9

Create vectors from words

Digital Temperature Sensor
Temperature is a scalar



Spectrogram – Audio
2D Matrix



Digital Image
A tensor of 3 stacked 2D matrices

	A	B	C	D	E	F	G	H	I	J	K	L
1	PassengerI	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
2	1	0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
3	2	1	1	Cumings,	female	38	1	0	PC 17599	71.2833	C85	C
4	3	1	3	Heikkinen	female	26	0	0	STON/O2.	7.925		S
5	4	1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
6	5	0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
7	6	0	3	Moran, Mr	male		0	0	330877	8.4583		Q
8	7	0	1	McCarthy,	male	54	0	0	17463	51.8625	E46	S
9	8	0	3	Palsson, M	male	2	3	1	349909	21.075		S
10	9	1	3	Johnson, (	female	27	0	2	347742	11.1333		S

Tabular Data
2D Matrix

Optimization

Optimization problem:

$F(x)$: Objective function

$\{c_i\}$: constraint functions

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}}\ F(x) \\ &\text{subject to } c_i(x) = 0 \quad i = 1,2,\ldots,m' \\ &\qquad\qquad\quad c_i(x) \geq 0 \quad i = m',m'+1,\ldots,n \end{aligned}$$

Some notation: $F : \mathbb{R}^n \mapsto \mathbb{R}$

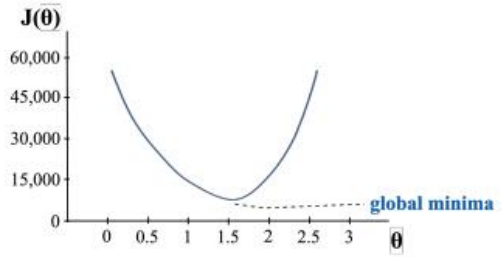
Gradient of F : $\nabla F = \left[\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \ldots, \frac{\partial F}{\partial x_n}\right]^{\cdot}$

Hessian matrix:
($n \times n$ symmetric matrix)

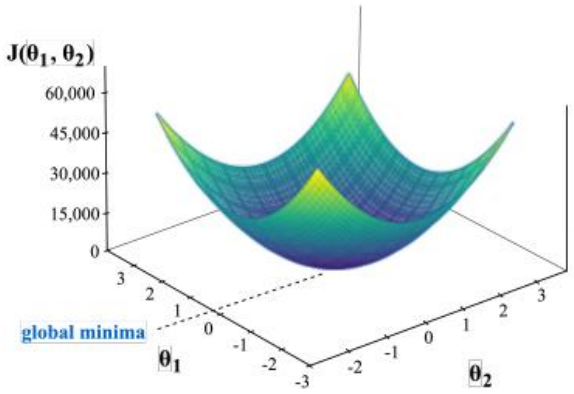
$$\nabla^2 F = \frac{\partial^2 F}{\partial x_i \partial x_j} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1^2} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial xn} \\ \vdots & & & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \cdots & \cdots & \frac{\partial^2 F}{\partial x_n^2} \end{bmatrix}^{\cdot}$$

Convex and Non-Convex Cost Functions

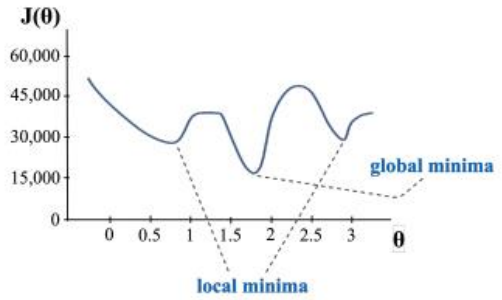
- For non-convex cost-functions with two parameters, there are three types of critical points that have gradient zero, and are thus relevant to the optimization:
  - **Saddle points** are the plateau-like regions
  - **Local minima** is the smallest values of the function within a specific range.
  - **Global minimum** is the smallest value of the function on the entire domain.



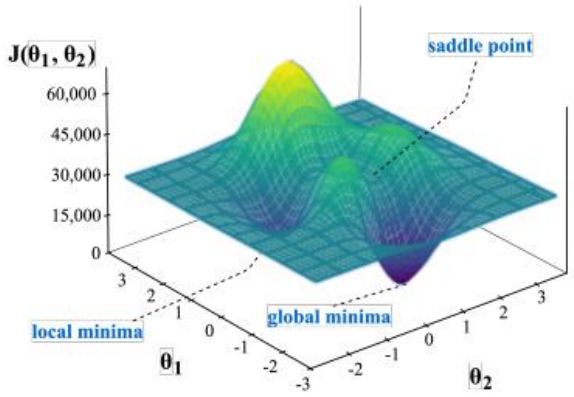
(a) Convex, one parameter.



(b) Convex, two parameters.



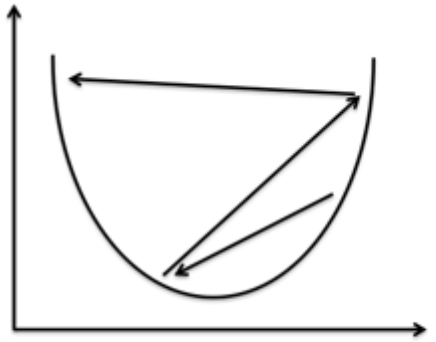
(c) Non-convex, one parameter; note that there is no saddle point in function of 1 parameter (saddle point occurs when gradient is zero, but point is minimum in one direction and maximum in another).



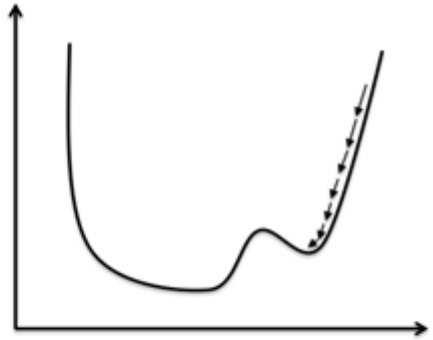
(d) Non-convex, two parameters

Optimization

Intuition for Steepest Descent



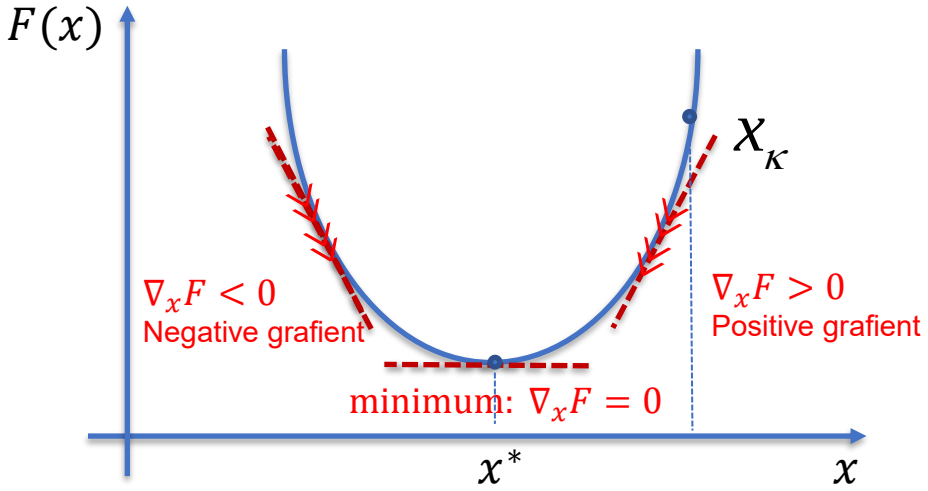
Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

1-dimentional

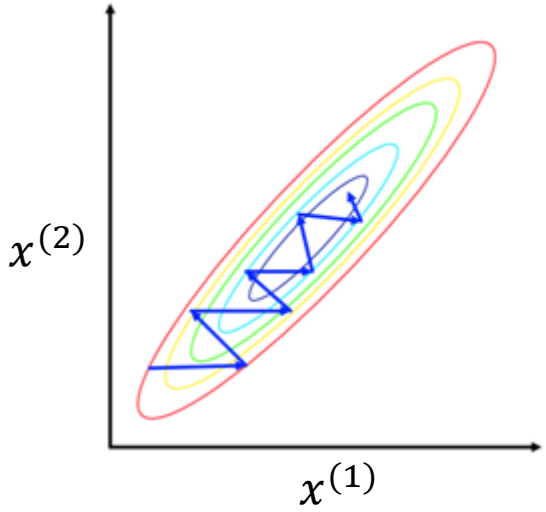
$x \in \mathbb{R}$

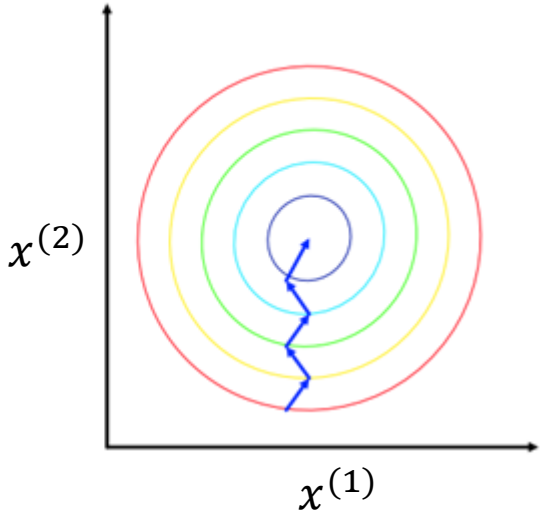


$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \lambda \nabla F(\boldsymbol{x}_k)$$

2-dimentional

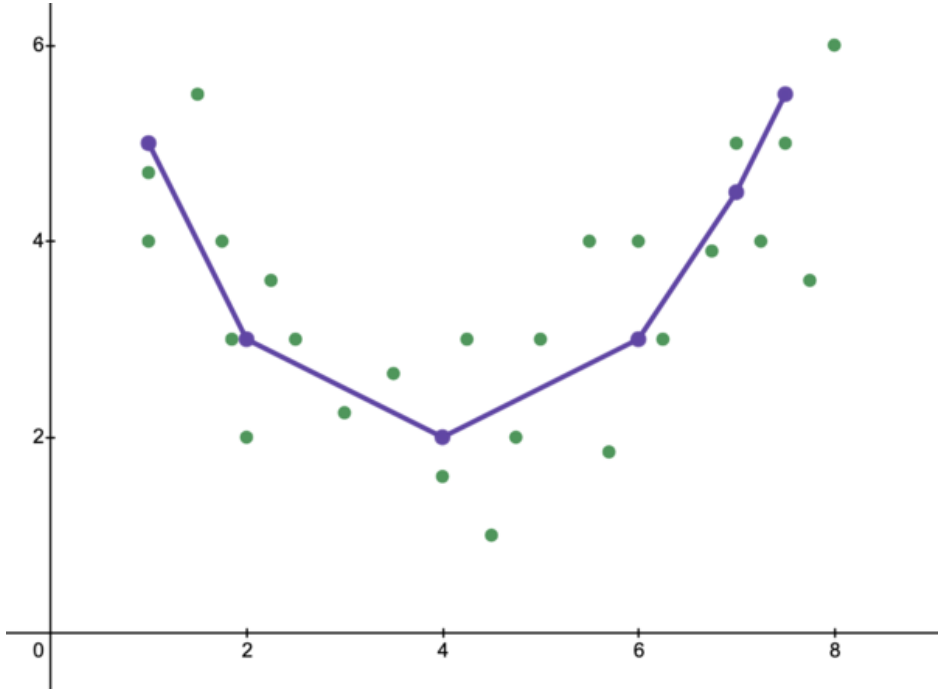
$x \in \mathbb{R}^2$



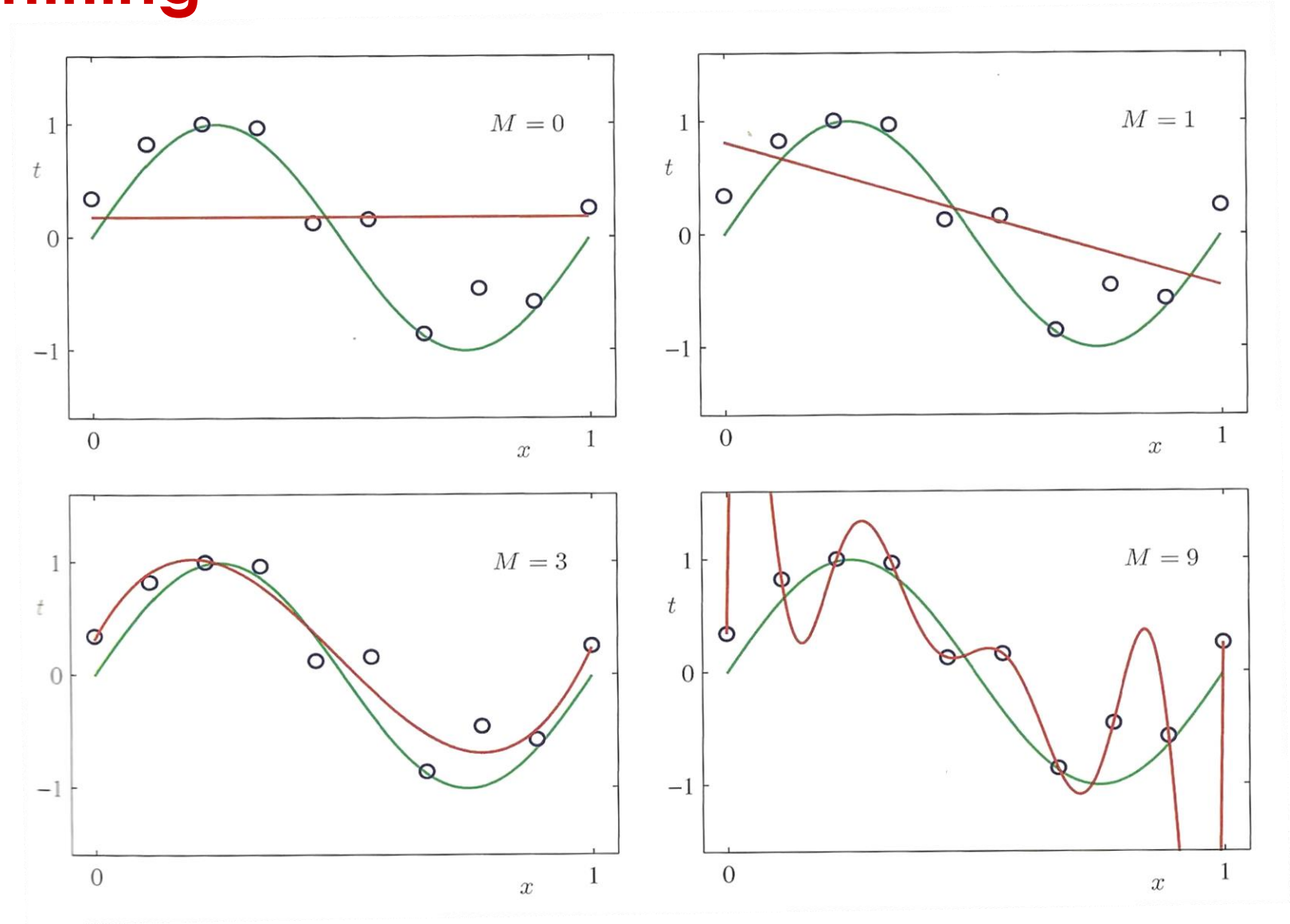


Model Selection, Underfitting, and Overfitting

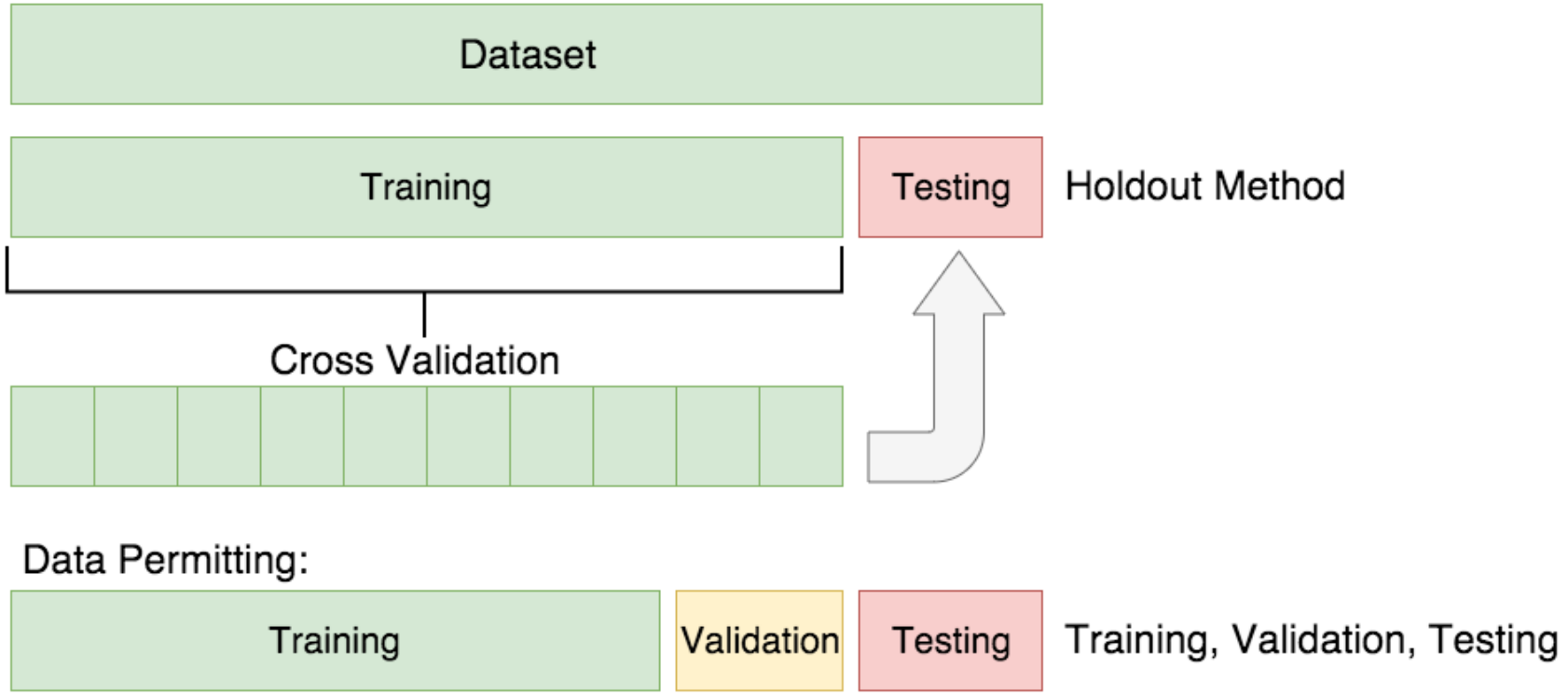
- As machine learning scientists, our goal is to discover *patterns*.
- But how can we be sure that we have truly discovered a *general* pattern and not simply memorized our data?



Influence of model complexity on underfitting and overfitting



Model Selection, Underfitting, and Overfitting



Train multiple Models

(e.g. Logistic Regression, Decision Trees, KNN)

Validate Models

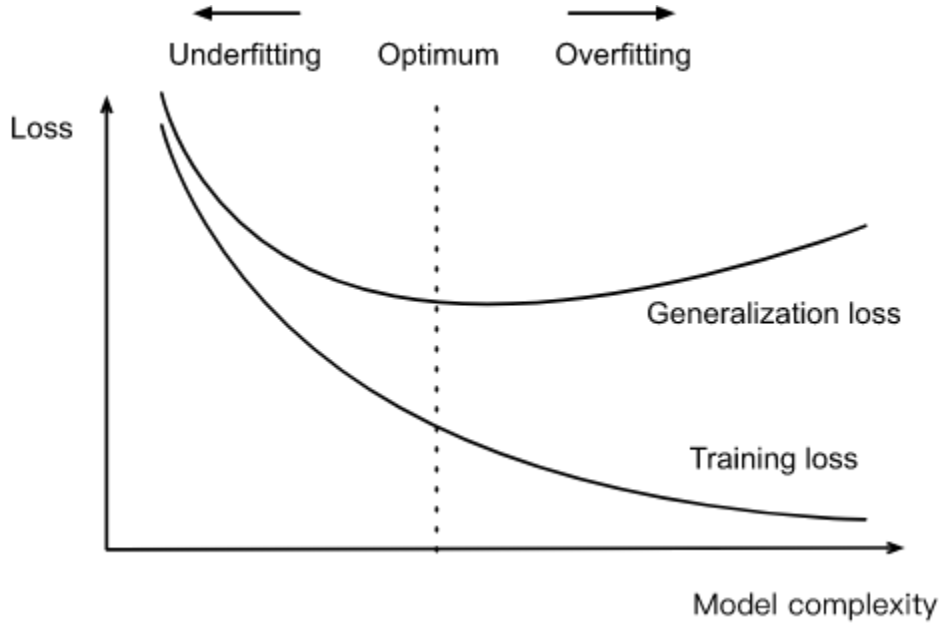
Tune Hyper parameters and Select the Best Model (e.g. Logistic Regression)

Evaluate Model

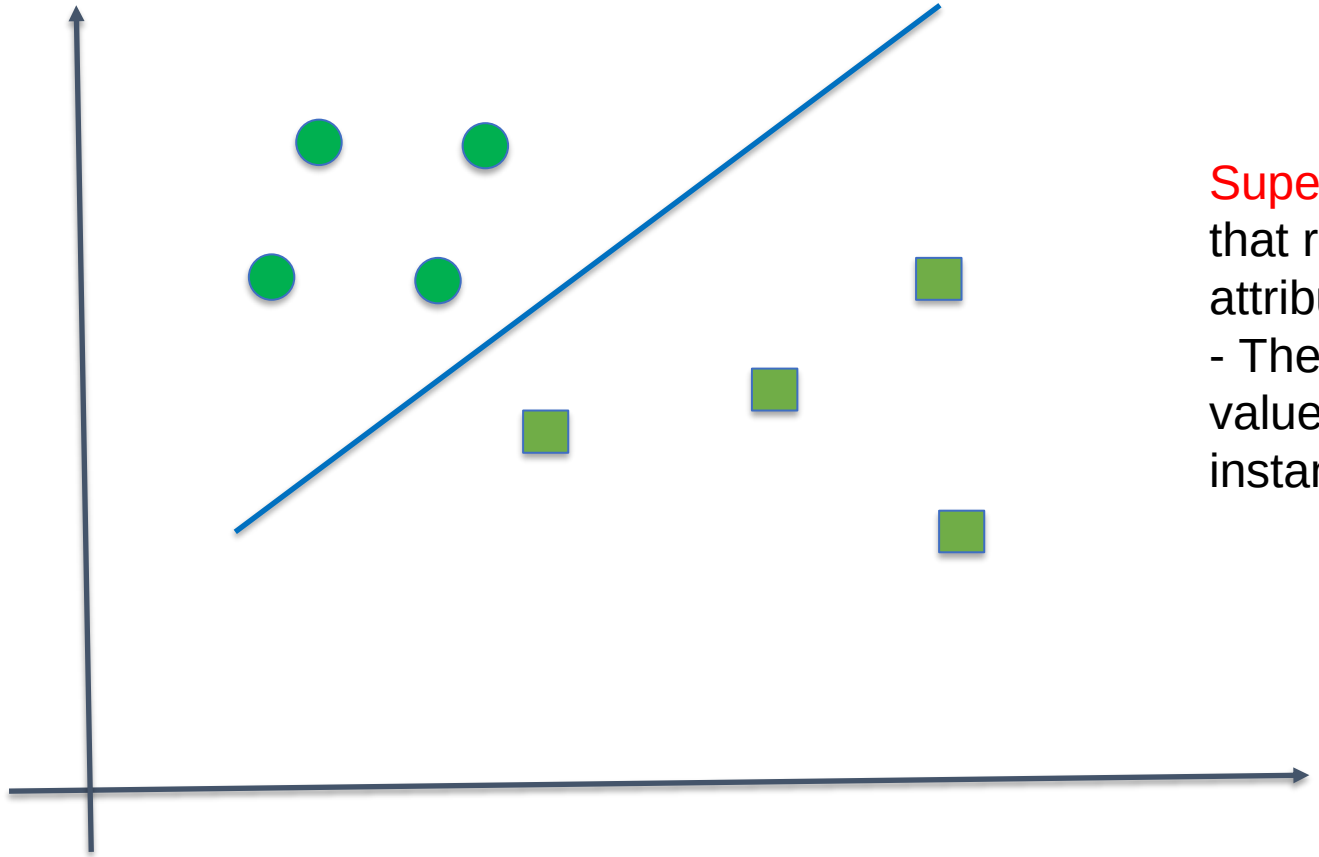
Evaluate the model based on various metrics (e.g. Confusion Matrix to evaluate the final performance of the selected Logistic Regression Model)

Influence of model complexity on underfitting and overfitting

- We want to watch out for the cases when our training error is significantly lower than our validation error, indicating **overfitting**.
- If the model is unable to reduce the training error, that could mean that our model is too simple, indicating **underfitting**
  - Training error and validation error are both substantial but there is a little gap between them.
  - If the generalization gap between our training and validation errors is small, we have reason to believe that we could get away with a more complex model.



Supervised Learning

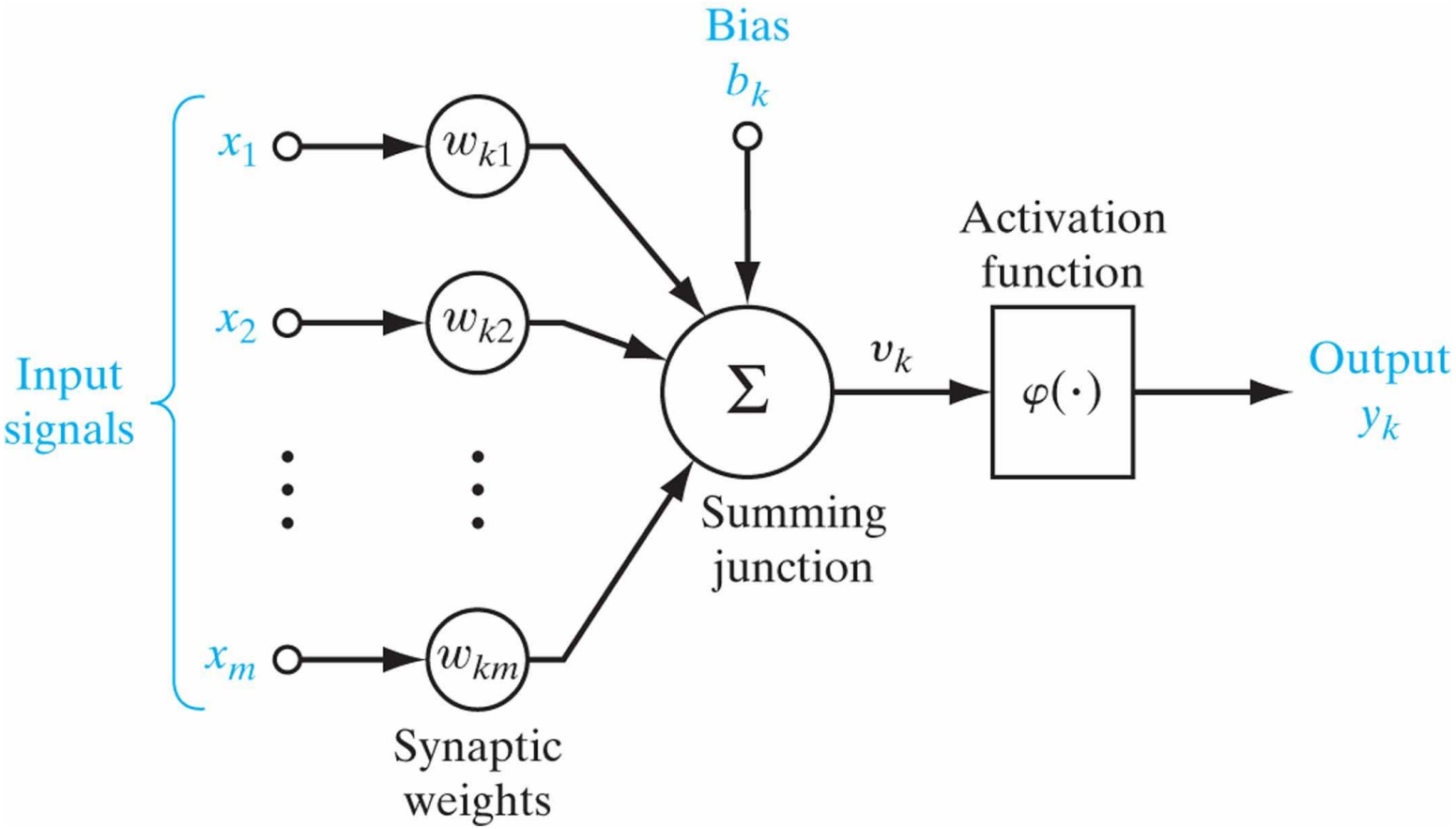


Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.

- These patterns are then utilized to predict the values of the target attribute in future data instances.

Training set: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\ldots (x_N, y_N)\}$

Artificial Neurons

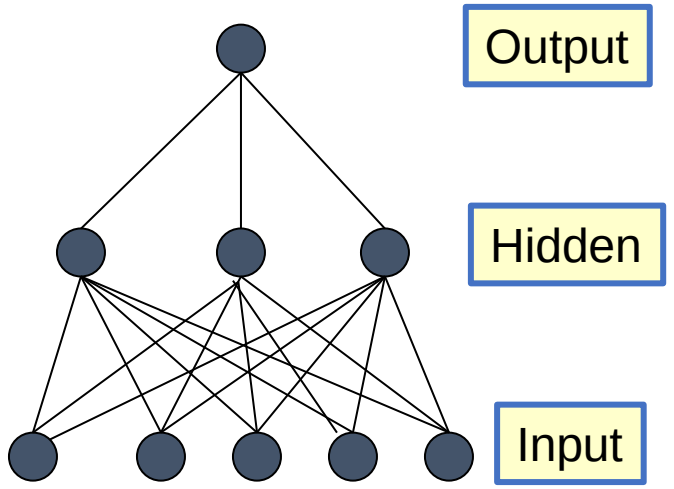


$$v_k = b_k + \sum_{i=1}^{m} w_{ki} x_i = \sum_{i=0}^{m} w_{ki} x_i, \qquad w_{k0} = b_k, \quad x_0 = 1$$

Neural Networks

- Multi-layer networks were designed to overcome the computational (expressivity) limitation of a single threshold element.

- The idea is to stack several layers of threshold elements, each layer using the output of the previous layer as input.

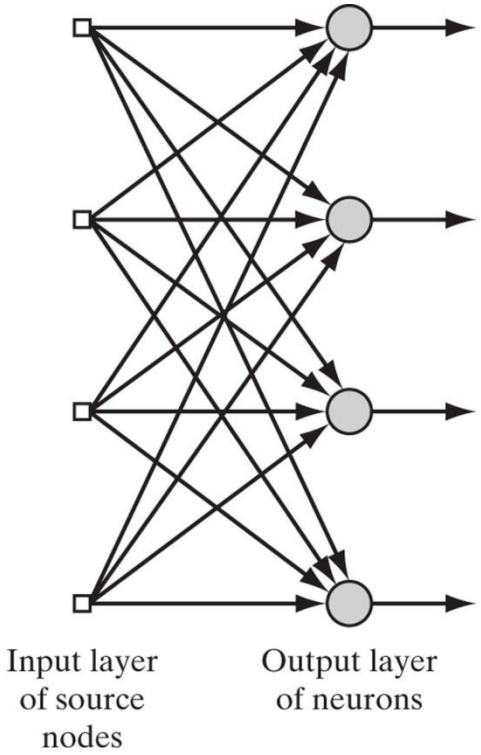


- Multi-layer networks can represent arbitrary functions, but building effective learning methods for such network was [thought to be] difficult.
  - Universal Approximation Theorem

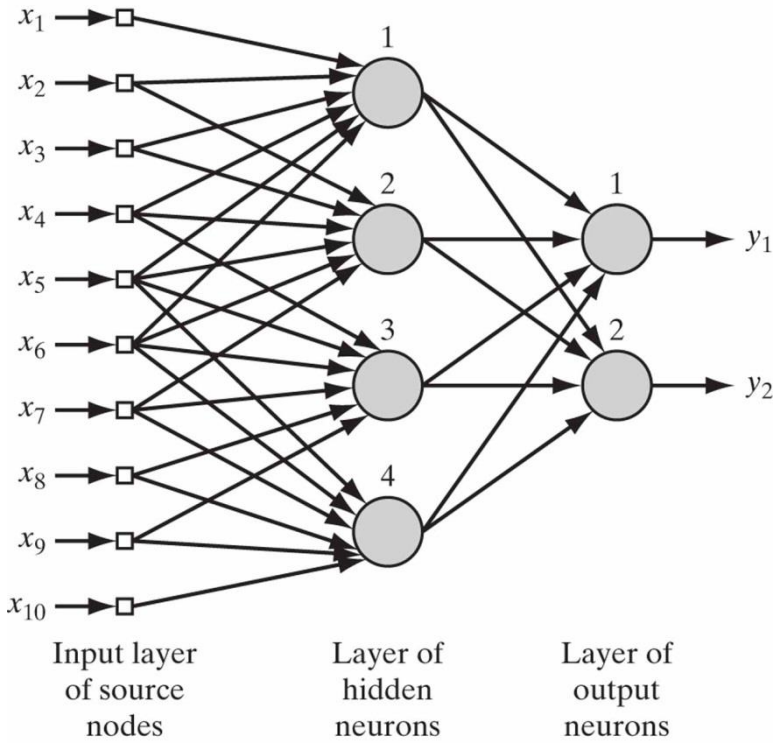
Building Artificial Neural Networks (ANNs)

Feedforward neural networks

Mostly applied to the broad areas of function approximation, classification and data processing (e.g. compression)

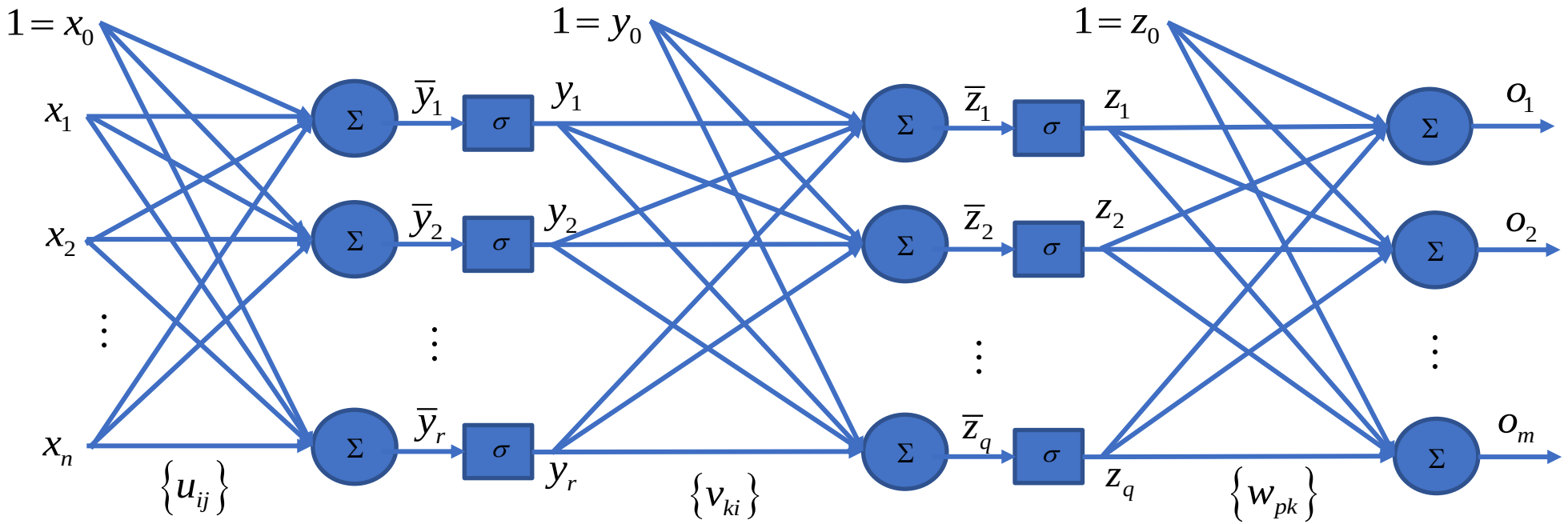


Single Layer



Multilayer

Multilayer Neural Networks



$$O = WZ$$
$$[m \times 1] = [m \times (q+1)] \times [(q+1) \times 1]$$

$$z_i = \sigma(\bar{z}_i),\ i = 1, \ldots, q \quad , \quad \bar{Z} = VY$$
$$[(q+1) \times 1] = [q \times (r+1)] \times [(r+1) \times 1]$$

$$y_j = \sigma(\bar{y}_j),\ j = 1, \ldots, r \quad , \quad \bar{Y} = UX$$
$$[(r+1) \times 1] = [r \times (n+1)] \times [(n+1) \times 1]$$

Weights: W, V, U

Input: X

Output: O

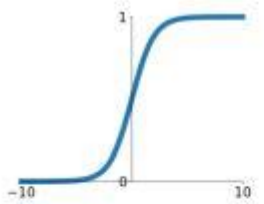
Activation or squashing function: $\sigma(.) : \mathbb{R} \mapsto \mathbb{R}$

Non-linear activation functions

- Choosing the right activation function is another new hyper-parameter!
  - ***Derivative or Differential:*** *Change in y-axis w.r.t. change in x-axis.It is also known as slope.*
  - ***Monotonic function:*** *A function which is either entirely non-increasing or non-decreasing.*

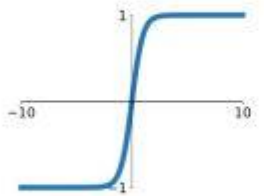
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



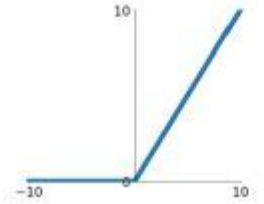
tanh

$$\tanh(x)$$



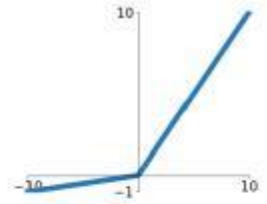
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

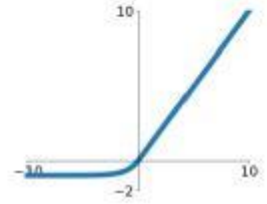


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

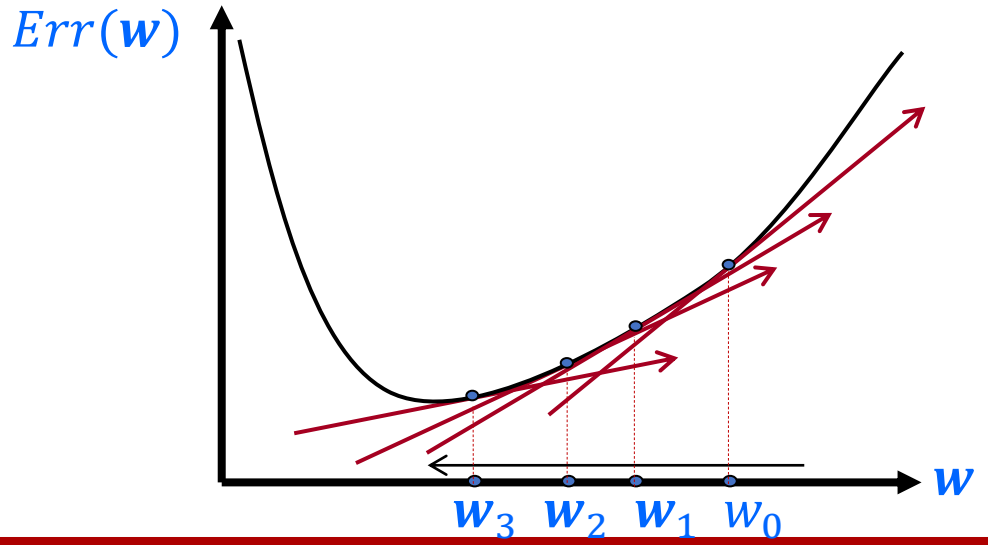
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Gradient Descent

- We use gradient descent to determine the weight vector that minimizes some scalar valued loss function $Err(\boldsymbol{w}^{(j)})$;
- Fixing the set D of examples, $E\mathrm{rr}$ is a function of $\boldsymbol{w}^{(j)}$
- At each step, the weight vector is modified in the direction that produces the steepest descent along the error surface.



Gradient descent with back propagation

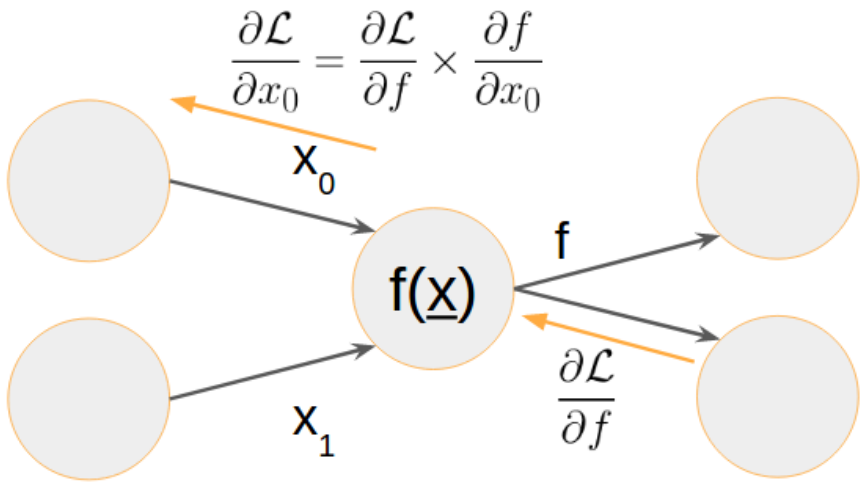
- To compute all the derivatives, we use a **backward** sweep called the **back-propagation algorithm**
  - Gradients are computed in the direction from output to input layers and combined using chain rule
  - Local derivatives are easy to compute because they care only about their own input and output.
  - Backward phase

- **Chain Rule**: If $z = f(g(\underline{x}))$ and $f(.)$, $g(.)$ are continuous differentiable functions then:

$$\frac{dz}{dx_i} = \frac{\partial f}{\partial g}\frac{\partial g(\underline{x})}{\partial x_i}$$

- More chain rule, remember:

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$



The Backpropagation Algorithm

- Create a fully connected network. **Initialize weights.**
- Until all examples produce the correct output within ϵ (or other criteria)

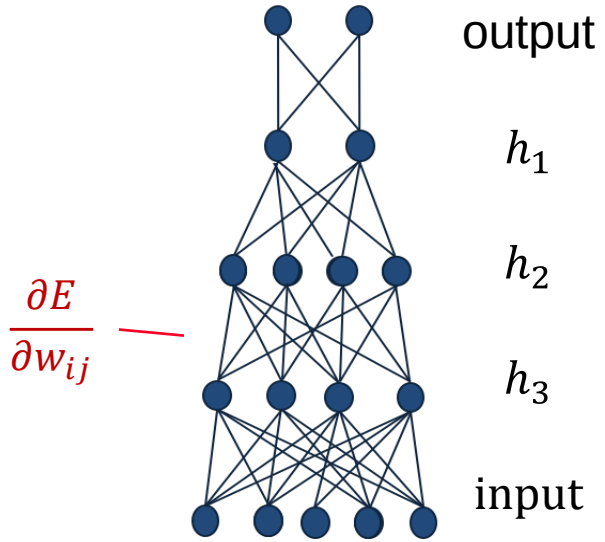
For each example (x_i, t_i) in the training set do:

1. Compute the network output y_i for this example
2. Compute the error between the output and target value

$$E = \sum \left(y_i^k - \hat{y}_i^k\right)^2$$

3. Compute the gradient for all weight values, Δw_{ij}
4. Update network weights with $w_{ij} = w_{ij} - \lambda * \Delta w_{ij}$

End epoch



Loop over instances:

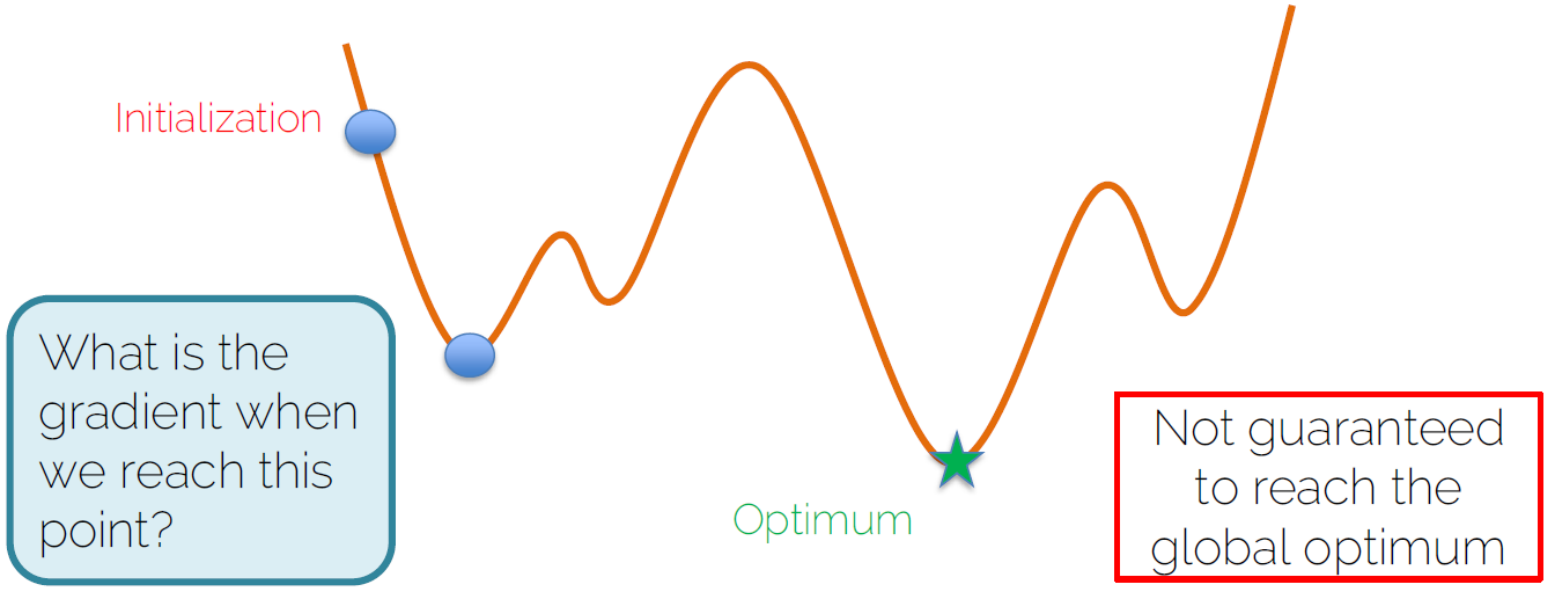
1. The forward step:
   Given the input, make predictions layer-by-layer, starting from the first layer)
2. The backward step:
   Calculate the error in the output
   Update the weights layer-by-layer, starting from the final layer

Quiz time!

- What is the purpose of forward step?
  - To make predictions, given an input.
- What is the purpose of backward step?
  - To update the weights, given an output error.
- Why do we use the chain rule?
  - To calculate gradient in the intermediate layers.
- Why backpropagation could be efficient?
  - Because it can be parallelized.

Convergence of Gradient Descent

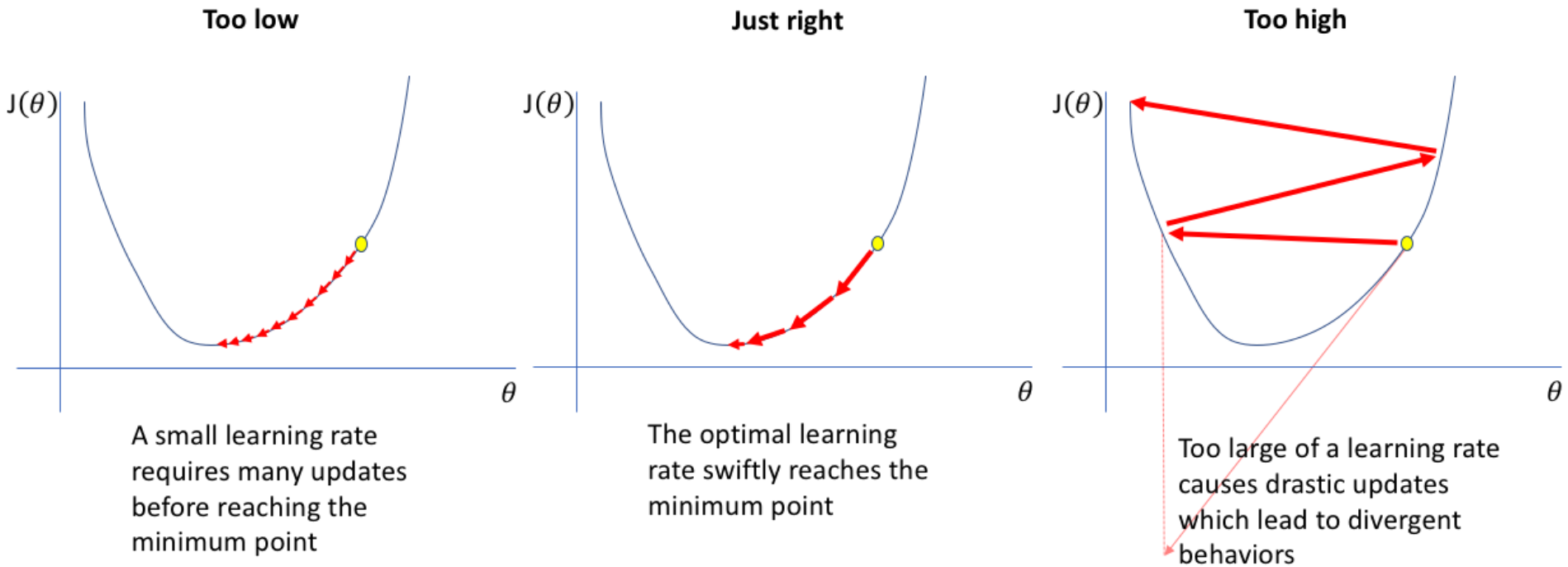
- Neural networks are non-convex
  - many (different) local minima
  - no (practical) way to say which is globally optimal
  - Learning rate schedulers change the learning rate dynamically



Effect of learning rate

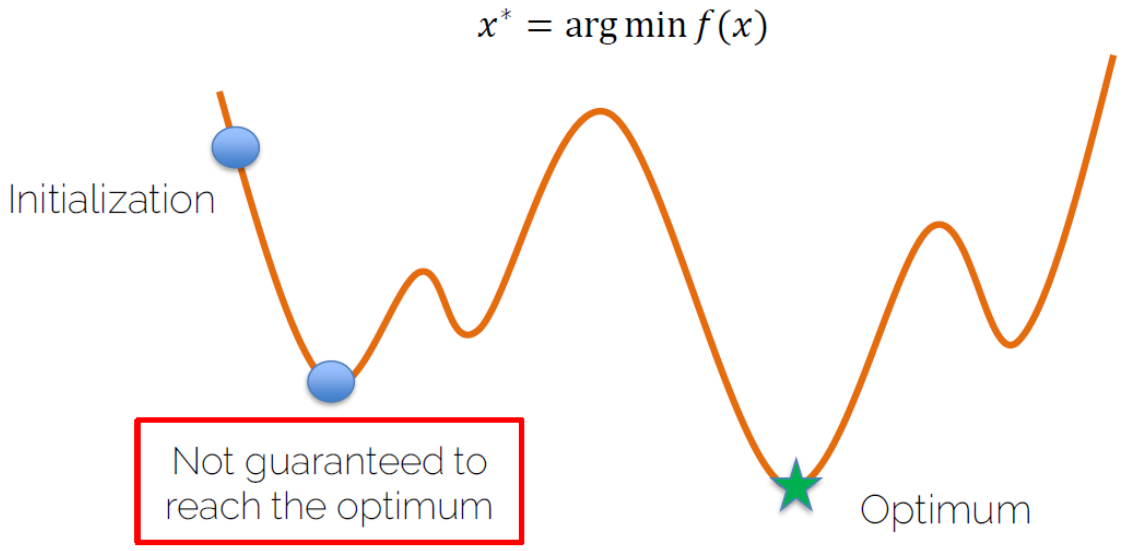
- Constant learning rate $0 < \alpha \leq 1$
  - Small: slow convergence.
  - Large: Oscillatory behavior near a local minimum

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}}$$



Initialization is Extremely Important

- The initialization step can be critical to the model's ultimate performance, and it requires the right method.
- How to initialize the weights?



Regularization

- **Regularization**: Prevent the model from doing *too* well on training data

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too* well on training data

- L2-regularization (weight decay): (regularization parameter)

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2}W^2$$

Enforces similar values for weights

λ = regularization strength (hyperparameter)

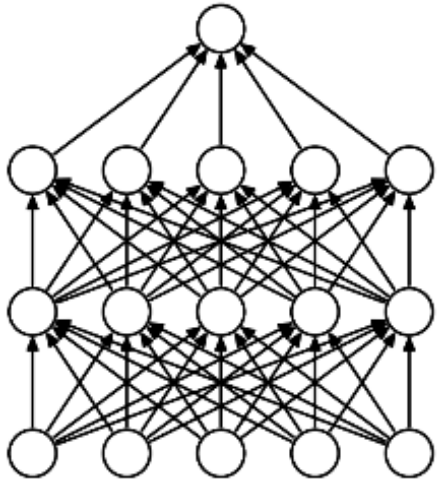
- L1-regularization:

$$\mathcal{L}_{new} = \mathcal{L} + \frac{\lambda}{2}|W|$$

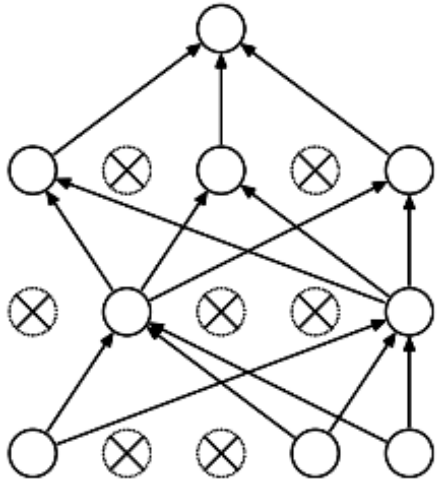
Enforces Sparsity

Dropout training

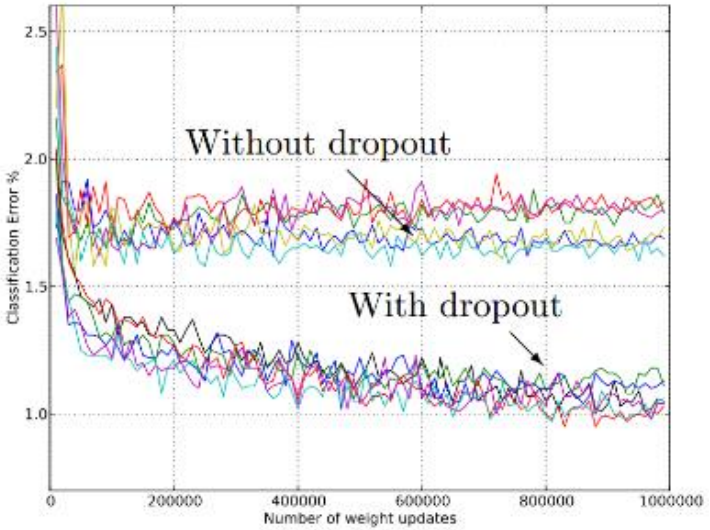
- We can specify the drop rate (or keep rate) per layer, e.g., 0.5
- Each time decide whether to delete one hidden unit with some probability p



(a) Standard Neural Net



(b) After applying dropout.

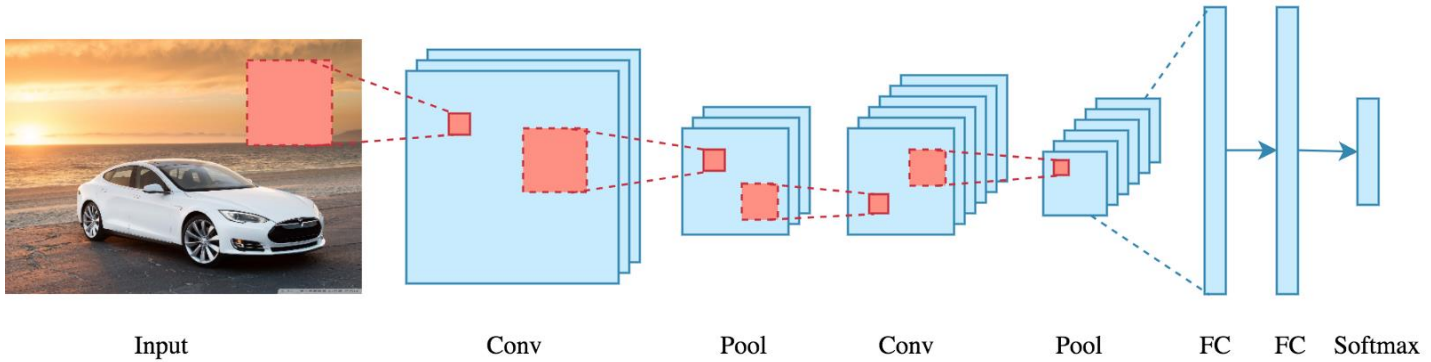


Dropout: A simple way to prevent neural networks from overfitting [Srivastava JMLR 2014]

Adapted from Jia-bin Huang

Convolutional Neural Networks (CNN)

- CNN is a class of deep neural networks
- Typically, used for image processing, signal processing, sound recognition.
- Avoids the fully connectness between layers
  - based on the shared-weight architecture
- Avoids the overfitting problem and requires significantly fewer adjustable parameters compared to the full-connected multilayer neural network
- A convolution kernel slides along the input matrix generating a feature map, followed by other layers such as pooling layers, fully connected layers, and normalization layers.



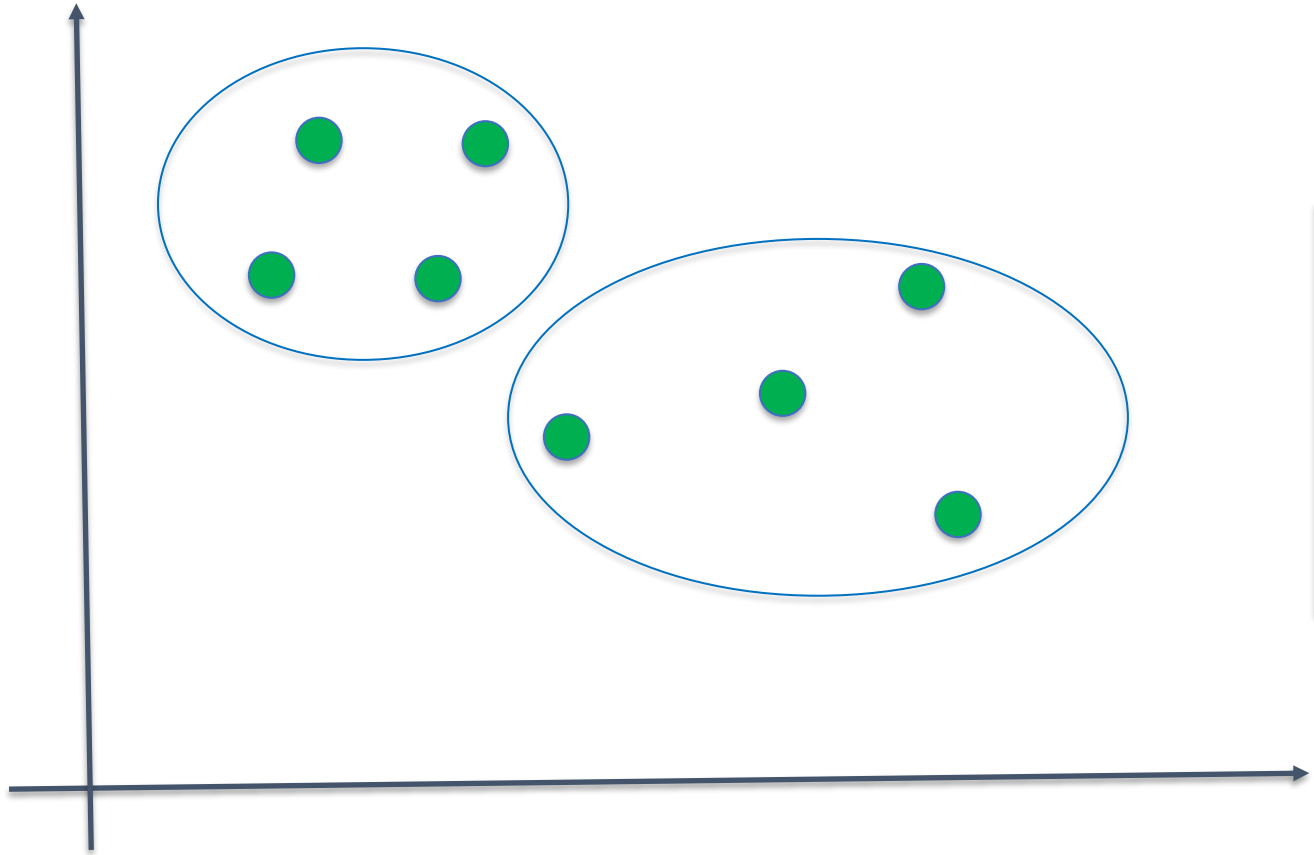
Comments on Training

- No guarantee of convergence; neural networks form non-convex functions with multiple local minima
- In practice, many large networks can be trained on large amounts of data for realistic problems.
- Many epochs (tens of thousands) may be needed for adequate training. Large data sets may require many hours even on GPU
- Termination criteria: Number of epochs; Threshold on training set error; No decrease in error; Increased error on a validation set.
- To avoid local minima: several trials with different random initial weights with majority or voting techniques

Comments on Training (2)

- Running too many epochs and/or a NN with many hidden layers may lead to an overfit network
- Using too many hidden units leads to over-fitting.
- Too few hidden units prevent the system from adequately fitting the data and learning the concept.
- Keep a held-out validation set and test accuracy after every epoch
- Early stopping: maintain weights for best performing network on the validation set and return it when performance decreases significantly beyond that.

Unsupervised Learning



- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Training set: $\{x_1, x_2, x_3, \ldots x_N\}$

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.
- Due to historical reasons, clustering is often considered synonymous with unsupervised learning.
  - In fact, many other tasks are also unsupervised

Aspects of clustering

- A clustering algorithm
  - Partitional clustering
  - Hierarchical clustering
  - …
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance $\Rightarrow$ maximized
  - Intra-clusters distance $\Rightarrow$ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

Clustering: K-means algorithm

INPUT:

- K – number of clusters ← hyperparameter
- Initial locations of cluster centroids $\{\mu_1^0, \mu_2^0, \mu_3^0, \dots \mu_K^0\} \mu_i^0 \in \mathbb{R}^n$
- Training set: $\{x_1, x_2, x_3, \dots x_N\} x_i \in \mathbb{R}^n$

OUTPUT:

- Final locations of cluster centroids $\{\mu_1^*, \mu_2^*, \mu_3^*, \dots \mu_K^*\} \mu_i^* \in \mathbb{R}^n$

Clustering: K-means algorithm

Randomly initialize K cluster centroids $\mu_1^0, \mu_2^0, \mu_3^0, \ldots \mu_K^0 \in \mathbb{R}^n$

Repeat{

for i = 1 to N

$c^{(i)} \in \{1, 2, \ldots K\} :=$ index of cluster centroid closest to x_i

for k = 1 to K

$\mu_k :=$ average (mean) of points assigned to cluster k

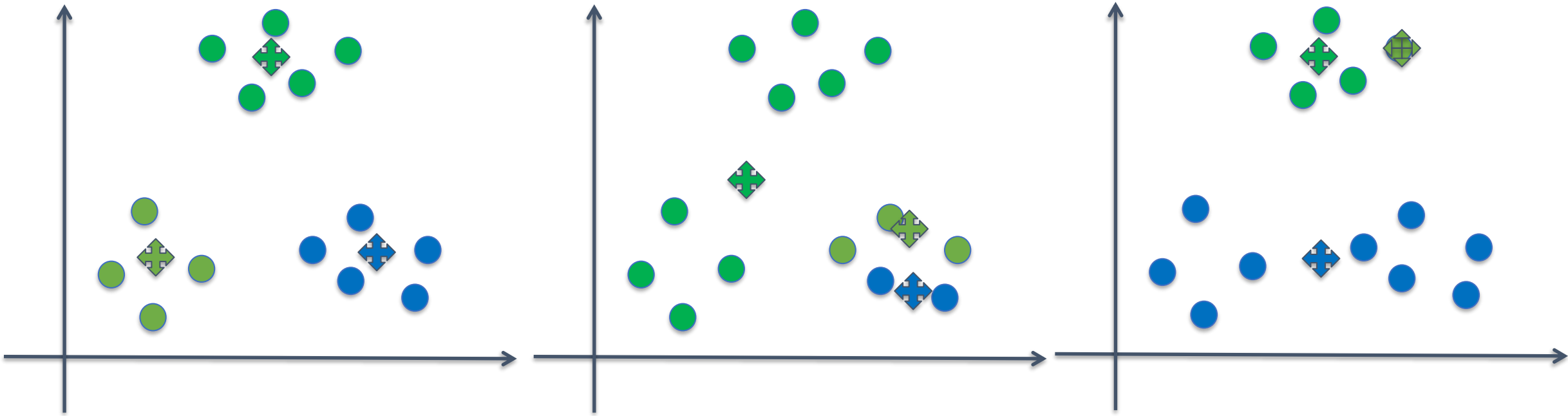
}

$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$

Clustering: K-means algorithm initialization

- Randomly pick K training points and set $\mu_1^0, \mu_2^0, \mu_3^0, \ldots \mu_K^0$ equal to these points
- The initialization of the cluster centroids sometimes affects the final result (two runs of the K-means Algorithm may result in two different models)
- Some variants of the K-means Algorithm compute the initial positions of the centroids based on some properties of the dataset

Clustering: K-means algorithm – local minima



$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K) = \frac{1}{N} \sum_{i=1}^{N} \left\| x_i - \mu_{c^{(i)}} \right\|^2$$

K-means Algorithm – Random Initialization

For j = 1 to 100 {

- Randomly initialize K-means
- Run K-means. Obtain: $c^{(1)}, \ldots, c^{(N)}, \mu_1, \ldots, \mu_K$
- Computer cost function (distortion)

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$$

}

Pick clustering solution that gave the lowest distortion

$$J(c^{(1)}, c^{(2)}, \ldots, c^{(N)}, \mu_1, \mu_2, \ldots, \mu_K)$$

K-means Algorithm – choosing the value of K

- Choosing the right number of clusters K is a difficult task. There are several methods, but none is optimal.
- One approach is to break up the dataset into training and test data and to use the concept of prediction strength to determine a suitable value of K.
- The cost function (distortion) can be used to find the relationship between the number of clusters and the distortion. However, as the number of clusters increases, the distortion may decrease, but there are trade-offs.

Dimensionality Reduction – Motivation

- Data compression
- Data visualization
- Interpretability of learning procedure

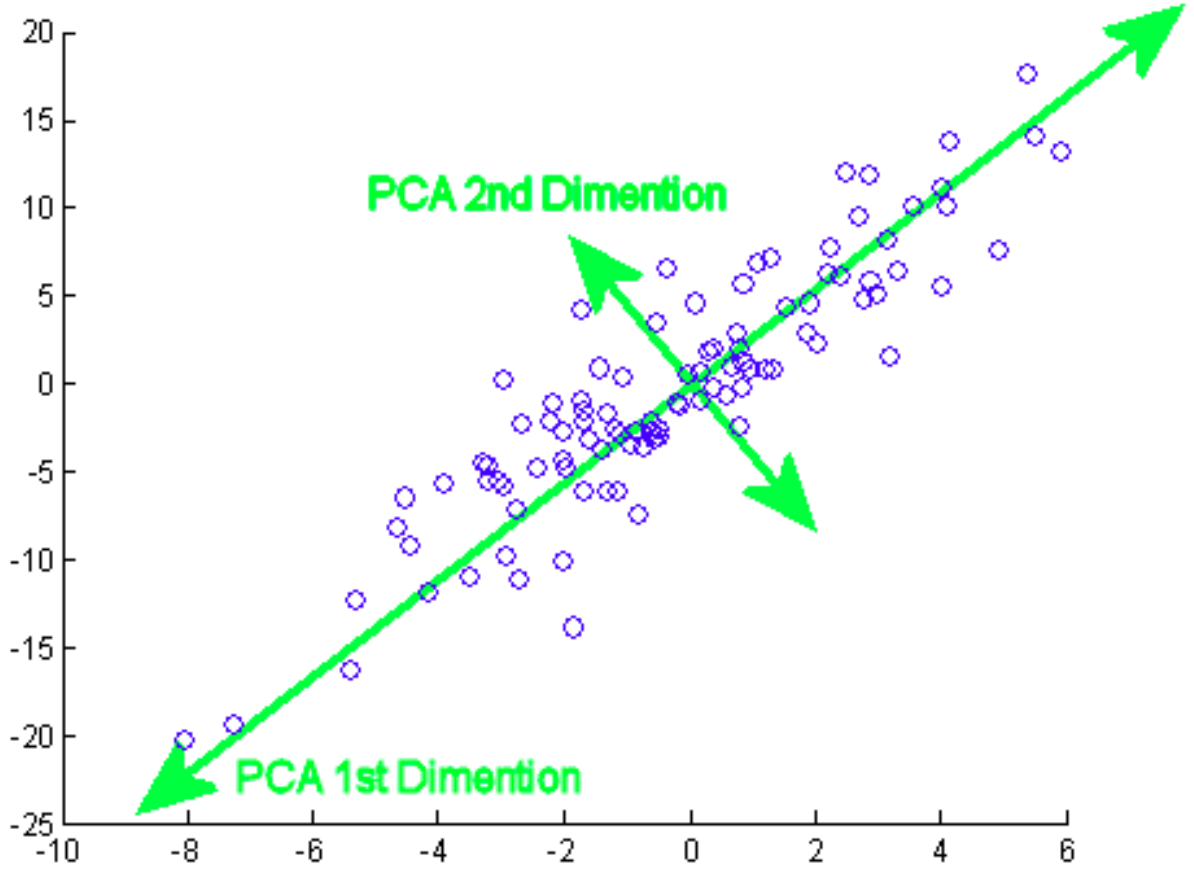
- Working in high-dimensional spaces can be undesirable:
  - raw data are often sparse as a consequence of the curse of dimensionality,
  - analyzing the data is usually computationally intractable (hard to control or deal with).
    - Dimensionality reduction is common in fields that deal with large numbers of observations and/or large numbers of variables

Principal Component Analysis (PCA)

- Nonparametric method for obtaining relevant information from complex datasets.
  - It is one of the oldest dimensionality reduction methods.
- It is a technique used to emphasize variation and bring out strong patterns in a dataset by highlighting similarities and differences.
- PCA is mainly an exploratory technique that can be used to gain better understanding of the correlation between variables.

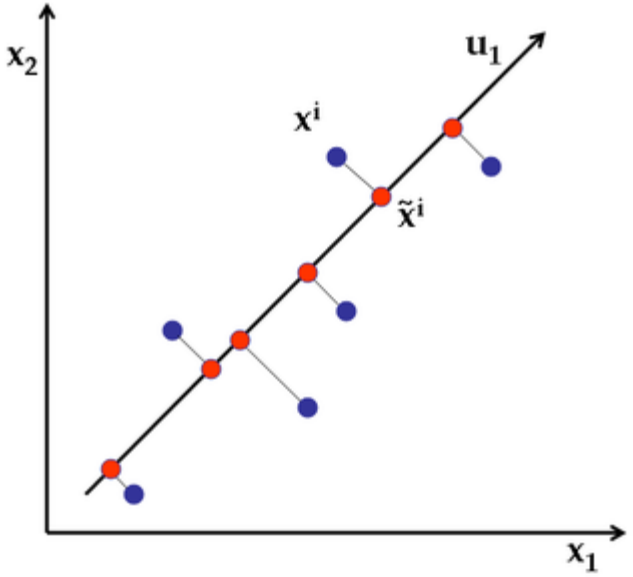
Principal Component Analysis (PCA)

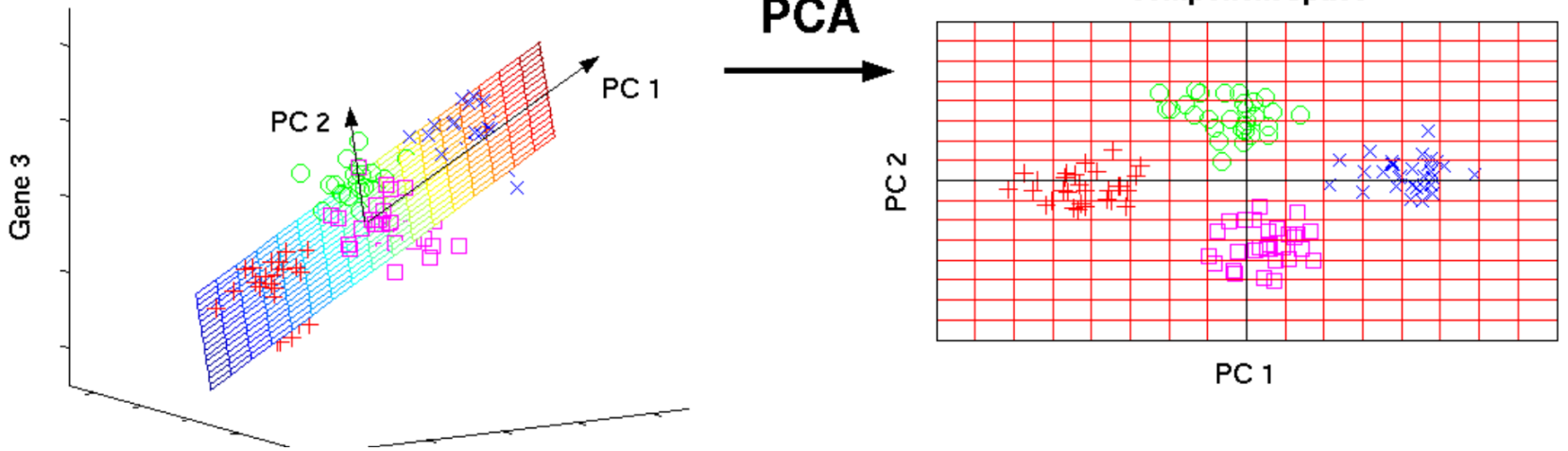
- The first axis (first principal component) is in the direction of largest variation.
- The second axis is orthogonal to the first axis and goes in the direction of second highest variance in the data.
- The third axis is orthogonal to both the first and second axis, and so on.



PCA problem formulation

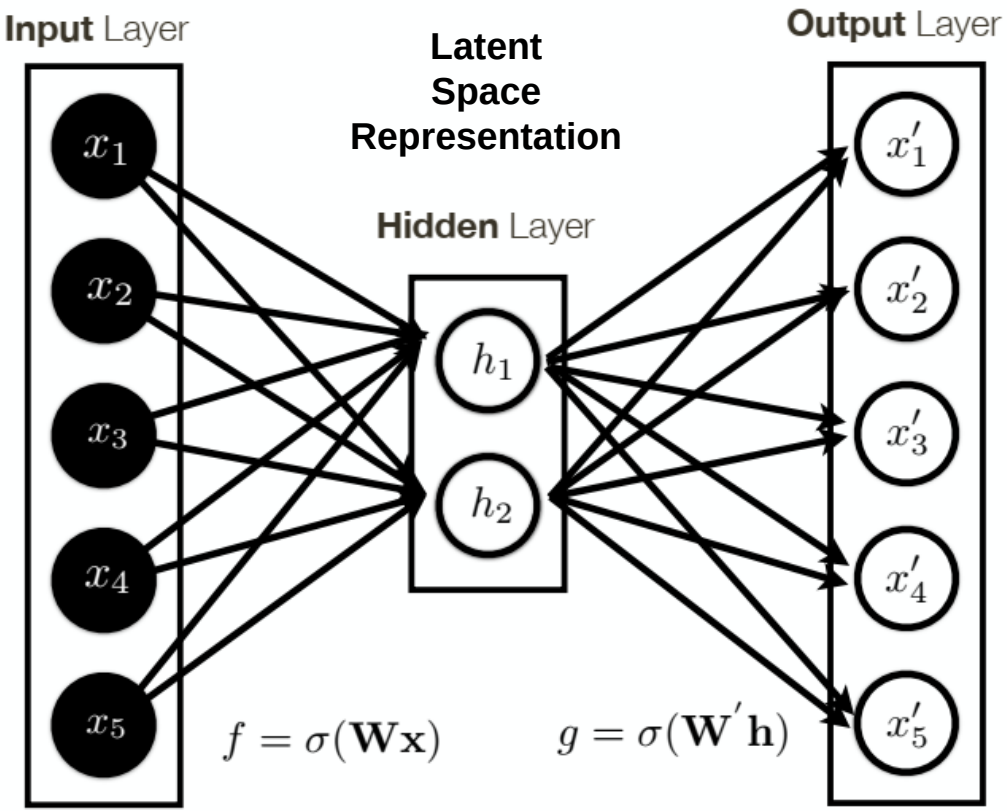
- Reduce from 2-dimension to 1-dimension: Find a direction (a vector $u_1 \in \mathbb{R}^n$) onto which to project the data so as to minimize the projection error.
- Reduce from N-dimension to K-dimension: Find K vectors onto which to project the data, so as to minimize the projection error





Autoencoders

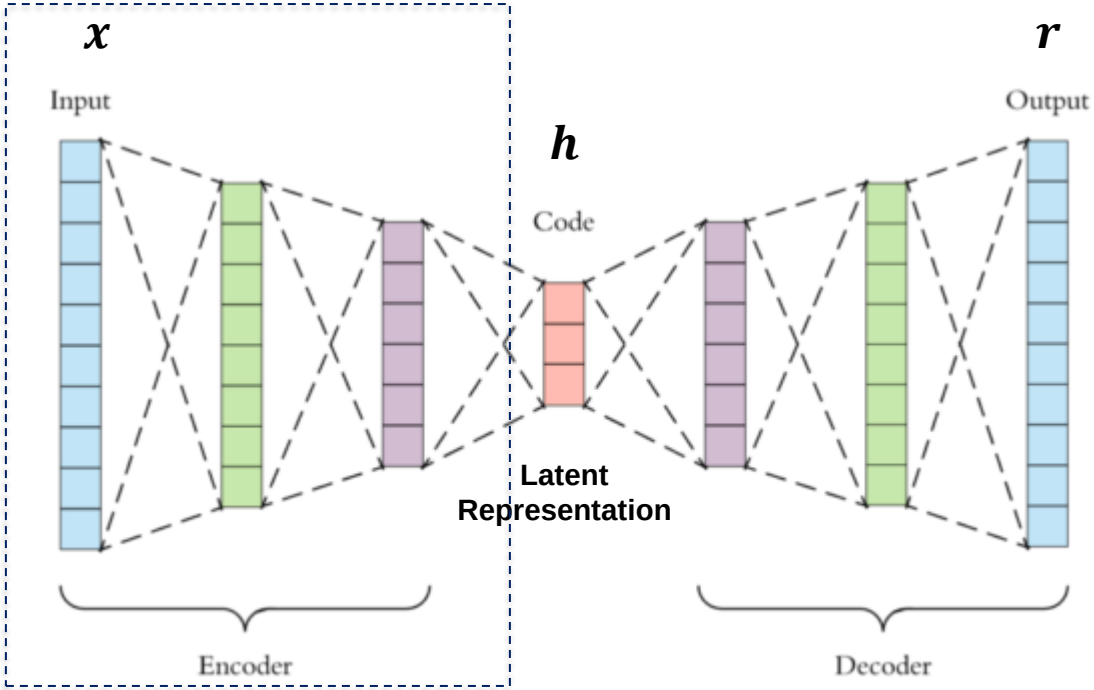
- Autoencoders are designed to reproduce their input
- Key point is to reproduce the input from a learned encoding.
- Feed forward network intended to reproduce the input



https://www.edureka.co/blog/autoencoders-tutorial/

Deep Autoencoders: structure

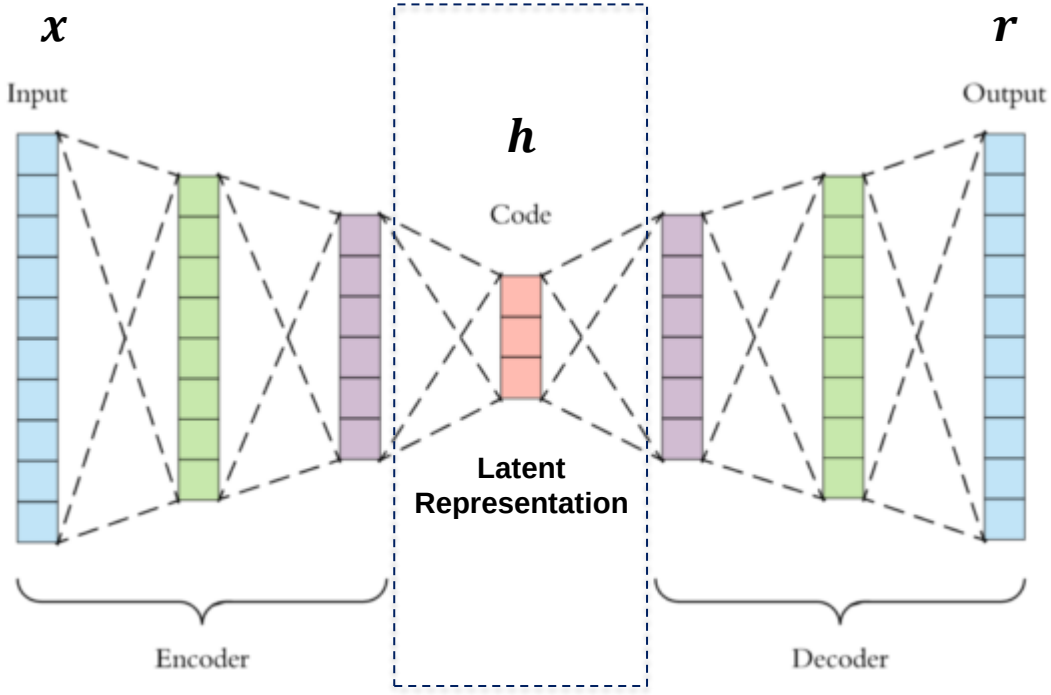
- Encoder:
  - Compress input into a latent-space representation of usually smaller dimension. $h = f(x)$



https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

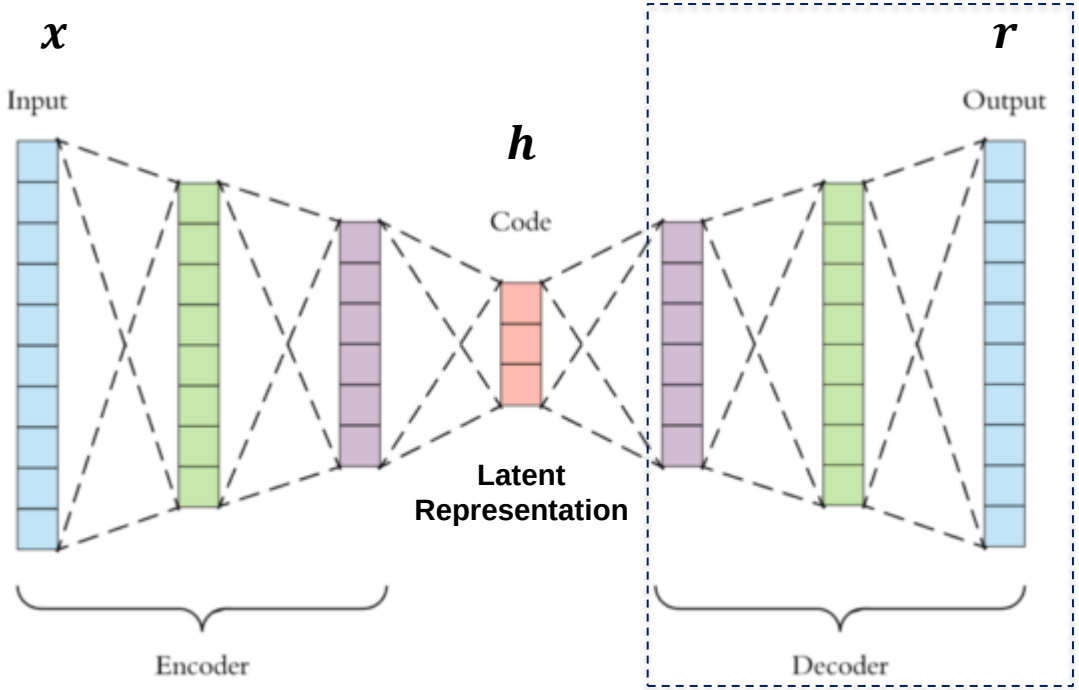
Deep Autoencoders: structure

- Code (i.e., also known as Bottleneck):
  - The layer between the encoder and decoder
  - This part of the network represents the compressed input h which is fed to the decoder.
  - Need a well-designed approach to decide which aspects of observed data are relevant information and what aspects can be discarded by balancing two criteria:
    - Compactness of representation, measured as the compressibility.
    - It retains some behaviourally relevant variables from the input.



Autoencoders: structure

- Decoder:
  - Reconstruct input from the latent space. $r = g(f(x))$ with r as close to x as possible
- The decoded data is a lossy reconstruction of the original data and it is reconstructed from the latent space representation.



https://towardsdatascience.com/deep-inside-autoencoders-7e41f319999f

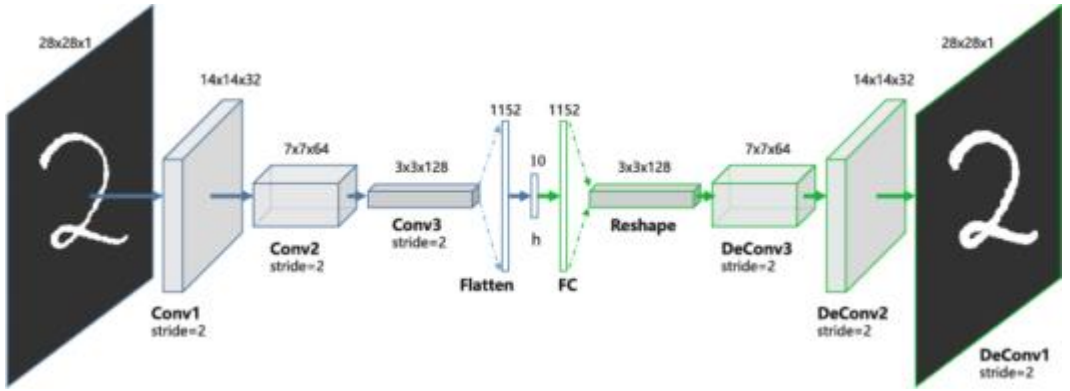
Properties of Autoencoders

- **Data-specific**: Autoencoders are only able to compress data similar to what they have been trained on.
- **Lossy:** The decompressed outputs will be degraded compared to the original inputs.
- **Learned automatically from examples:** It is easy to train specialized instances of the algorithm that will perform well on a specific type of input.
- It doesn't have to use dense layers.
  - It can use convolutional layers to learn which is better for video, image and series data.

https://www.edureka.co/blog/autoencoders-tutorial/

Convolution Autoencoders

- Autoencoders in their traditional formulation does not take into account the fact that a signal can be seen as a sum of other signals.
- Convolutional Autoencoders use the convolution operator to exploit this observation.
- They learn to encode the input in a set of simple signals and then try to reconstruct the input from them.
- Appropriate operations can be formed to upscale a tensor to higher resolution.

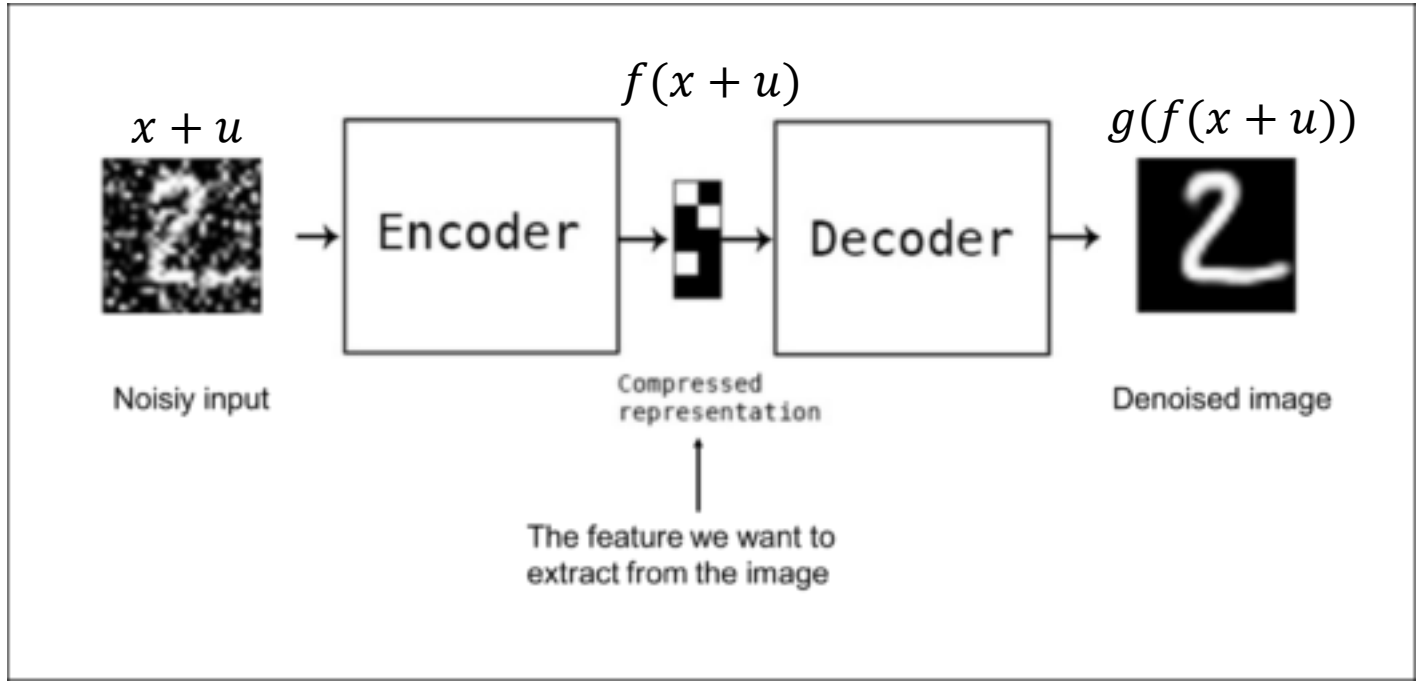


Hyperparameters of Autoencoders

- Code size:
  - It represents the number of nodes in the middle layer. Smaller size results in more compression.
- Number of layers:
  - The autoencoder can consist of as many layers as we want.
- Number of nodes/filters per layer:
  - The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder. The decoder is usually symmetric to the encoder in terms of the layer structure.
- Loss function:
  - We either use mean squared error or binary cross-entropy. If the input values are in the range [0, 1] then we typically use binary cross-entropy, otherwise, we use the mean squared error.

Denoising autoencoders

- Basic autoencoder trains to minimize the loss between x and the reconstruction $g(f(x))$.
- Denoising autoencoders train to minimize the loss between x and $g(f(x+u))$, where u is **random noise**.
- Denoising autoencoders are a stochastic version of standard autoencoders that **reduces the risk of learning the identity function.**



- **Note:** different noise is added during each epoch

https://www.edureka.co/blog/autoencoders-tutorial/

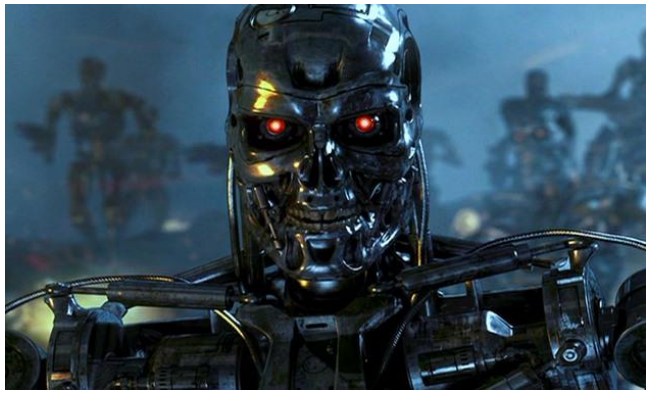
Autoencoder Tasks

- Dimensionality Reduction
  - For example, a 2006 study resulted in better results than PCA, with the representation easier to interpret and the categories manifested as well-separated clusters
- Anomaly Detection
  - If the input and reconstruction do not match this might signal an anomaly (out of distribution/corrupted input).
- Feature Learning
  - Good features can be obtained in the hidden layer

Issues

- According to two recent Gartner reports, 85% of AI and machine learning projects fail to deliver, and only 53% of projects make it from prototypes to production.

- Ethics, e.g., Amazon has/had sub-par employees fired by an AI automatically



Resources: Journals

- Journal of Machine Learning Research www.jmlr.org
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- IEEE Transactions on Neural Networks and Learning Systems
- IEEE Transactions on Artificial Intelligence
- Pattern Recognition

Resources: Conferences

- International Conference on Machine Learning (ICML)
- Neural Information Processing Systems (NeurIPS)
- International conference on Learning Representations (ICLR)
- International Joint Conference on Artificial Intelligence (IJCAI)
- International Conference on Neural Networks (Europe)
- European Conference on Machine Learning (ECML)

Some Final Thoughts

- We have learned a lot
- A lot that we have not learned!
- The importance of gaining intuition
- Never lose the big picture and the applications