Departamento de Eletrónica, Telecomunicações e Informática

Compements of Machine Learning

Lecture 3 : Practical Aspects of Deep Learning

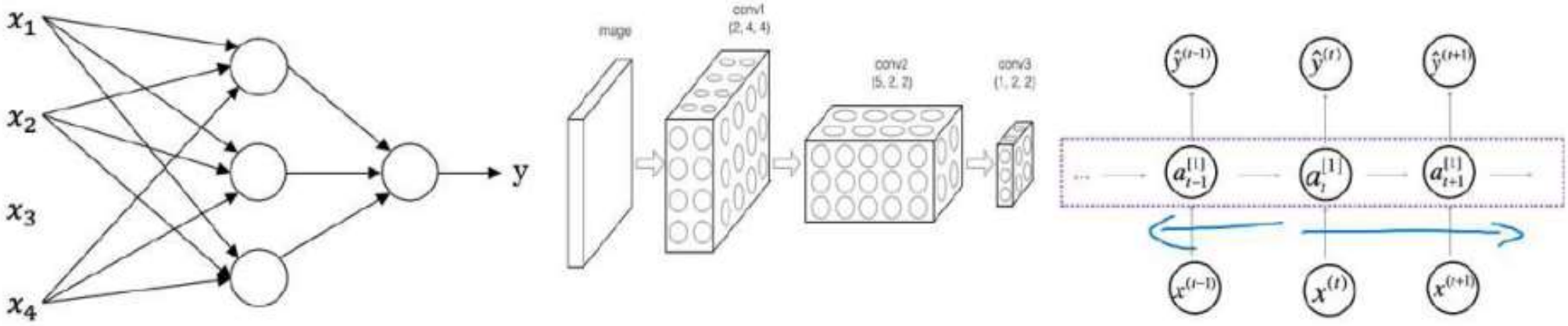
Petia Georgieva
(petia@ua.pt)

universidade de aveiro

Outline

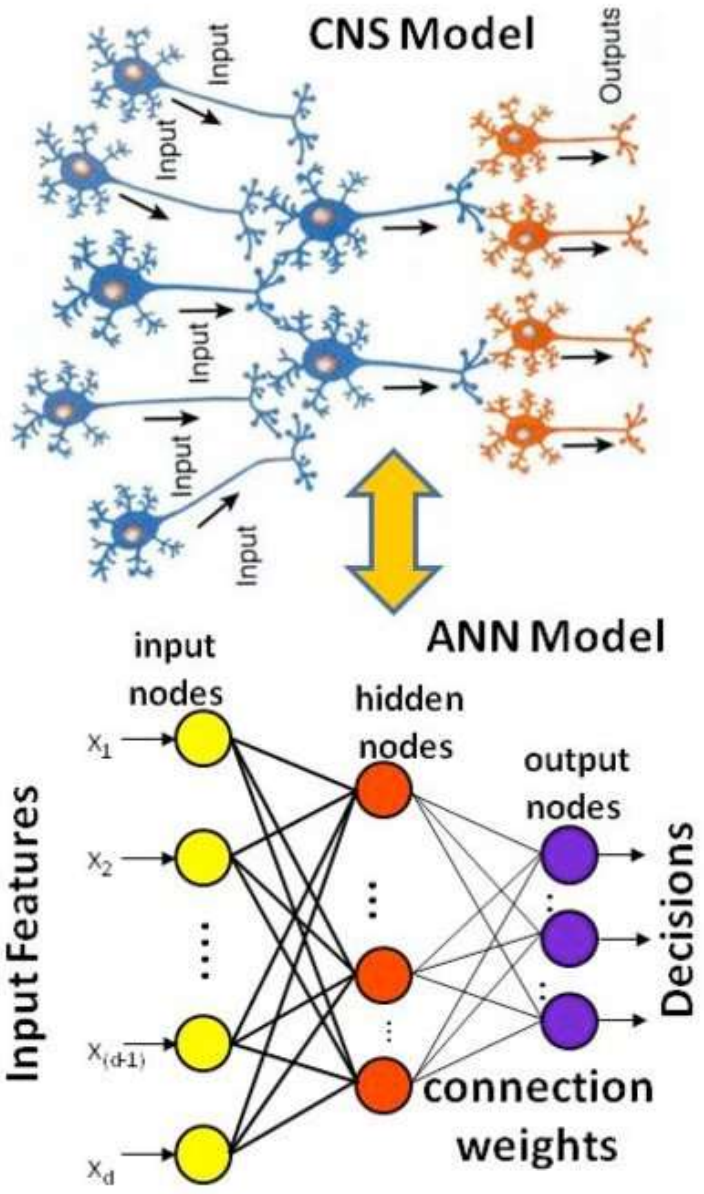
- **Learning with Neural Networks / Deep NN**
- **Regularization –dropout, early stopping**
- **Training and Optimization**

ANN Architectures



? ? ?

Analogy: natural learning



- Central Nervous System (CNS)
- The brain is a highly complex structure, **non linear** and highly **parallel;**
- It has an ability to organize its neurons, to perform complex tasks;
- A neuron is 5/6 times slower than a logical port;
- The brain overcomes this slowness through a parallel structure;
- The human cortex has 10 billion neurons and 60 trillion synapses!!

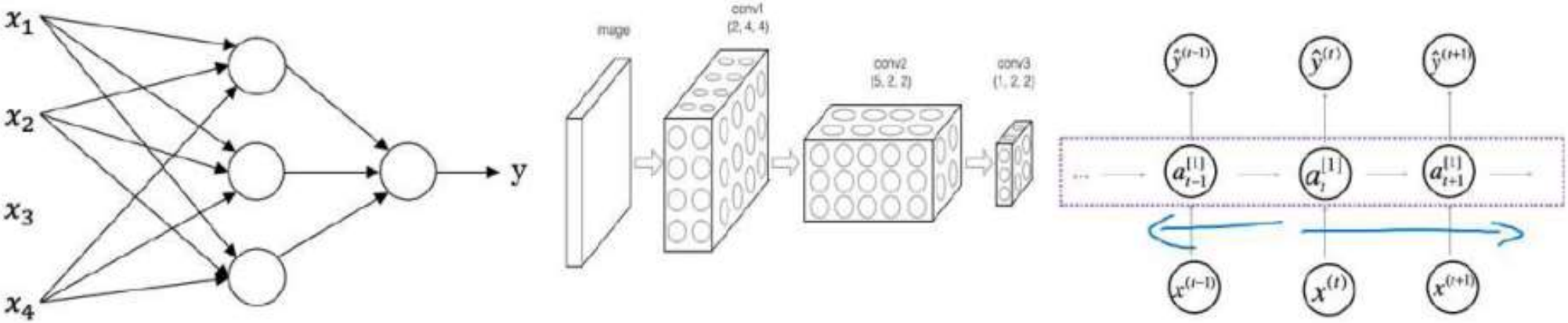
ANN

- Artificial Neural Networks (ANN) are models of machine learning that follow an analogy with the functioning of the human brain.
- **The relation between ANN (DNN) and the brain is a very basic/loose analogy !!!**
- A neural network is a parallel processor, consisting of simple processing (math) units (aka neurons).
- Knowledge is stored in the connections between the neurons.
- Knowledge is acquired from the environment (data) through a learning process (training algorithm) that adjusts the weights of the connections.

ANN architectures

Different types of Artificial Neural Networks (NN):

- Standard (shallow, 3 Layers) NN – useful for what kind of data ?
- CNN (convolutional neural networks) - useful for what kind of data ?
- RNN (Recurrent neural networks) - useful for what kind of data ?



Standard NN Convolutional NN Recurrent NN

Learning with NN

Different types of neural networks for supervised learning:

- Standard (shallow, 3 Layers) NN - for Structured data

- CNN (conv nets) - in Computer vision (image processing)

- Recurrent neural networks (RNN) - in Speech recognition, machine transl.
  LSTM, GRU Natural Language Processing (NLP)

- Hybrid/custom NN/Mixture of NNs types- structured & unstructured data

Structured vs Unstructured data

Structured data (databases, tables)

Size	#bedrooms	...	Price (1000$s)
2104	3		400
1600	3		330
2400	3		369
⋮	⋮		⋮
3000	4		540

User Age	Ad Id	...	Click
41	93242		1
80	93287		0
18	87312		1
⋮	⋮		⋮
27	71244		1

Unstructured data (audio, image, video, text)

Audio

Image

Four scores and seven years ago...

Text

Supervised learning examples

95%-99% of the economic value criated by AI nowadays is one type of AI: Supervised Learning (x to y mapping)
X (input) => y (output)

Input(X)	Output (y)	Application	NN type
Home features	Price	Real estate	Standard
Ad + user info	Click on Ad (0/1)	On-line advertising	Standard
Image	Object (1,...1000)	Photo taging	ConvNet (CNN)
Audio	Text transcript	Speech recognition	RNN
Portuguese	English	Machine translation	RNN
Image+Radar info	Position of other cars	Autonomous driving	Custom/Hybrid

AI in the industry – challenges

Software Industry (strongly affected by AI) : Web Search; On-line Advertysing; Language translation; Social Media

Non-Software Industry (still long way to go): Manifacture, Agriculture, Retail, Transportation, Logistics, etc.

Major bottleneck in (Non-Software) Industry :

- Valuable business use cases
- Lack of enough (labelled) data
- Lack of AI expertise in the company

Deep Learning (DL) Progress

Why DL is rising so fast over the last 10 years ? - many factors

- Lots of digital data - e.g. cell phone data / IoT / 5G
Small NN has bad performance on big data
Large NN have very good performance

- Better hardware - GPU's, NVIDIA

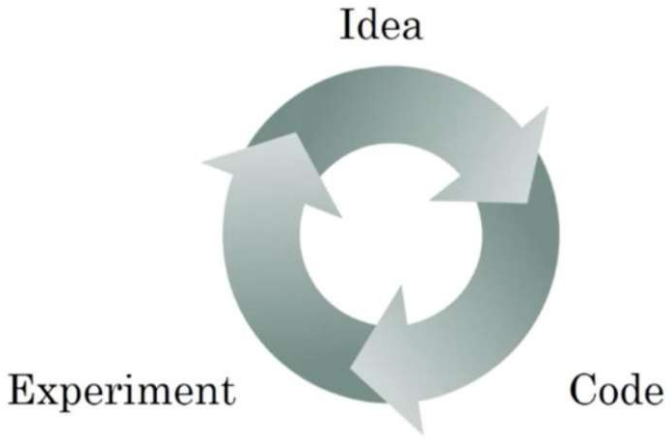
- New Algorithms & computation - e.g. change from Sigmoid to Relu act. function => Gradient descent performs faster => faster computation.

- Lots of research leaded by companies – OpenAI, Google, Microsoft, Facebook, Amazon, automotive industry, etc.

The academy goes behind the companies…

ML Strategy

- Lots of research –
typical cycle of making a ML strategy
Idea->Code->Experiment => REPEAT



- Collect more data
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Try dropout
- Add L_2 regularization
- Network architecture
  - Activation functions
  - # hidden units

Logistic Regression – binary classification

Given labelled data of m examples and n features
Labels $\{0,1\}$ => binary classification, e.g. cat/not cat image
x –vector of features (e.g. image pixels transformed into a long vector)
$\{w, b\}$ – model trainable parameters (w – weights; b- bias)

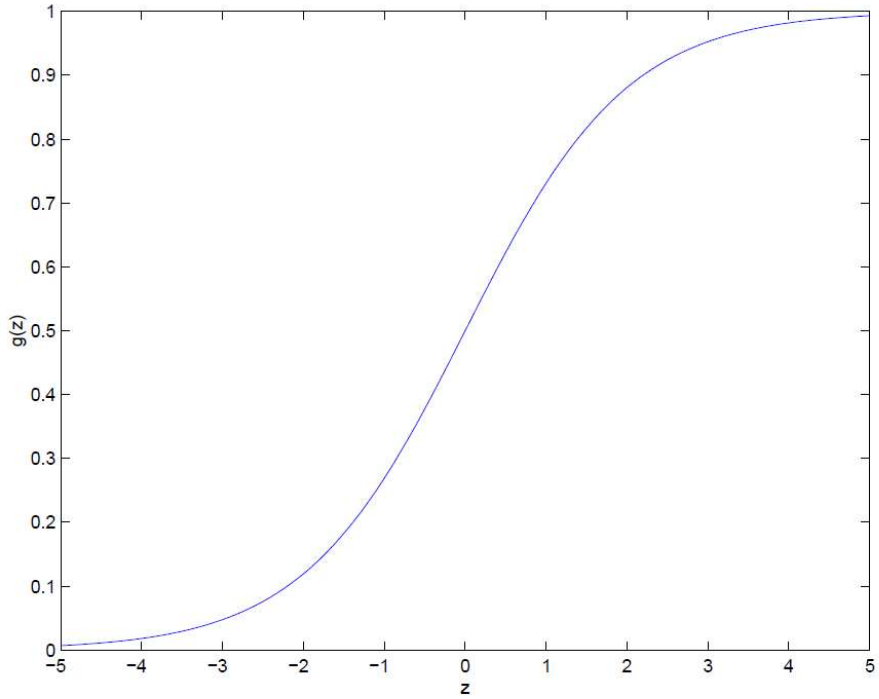
$$\hat{y} = \sigma(w^T x + b) = \sigma(z) = \frac{1}{1+e^{-z}}$$

$$z = w^T x + b = w_1 x_1 + w_2 x_2 + \ldots . w_n x_n + b$$

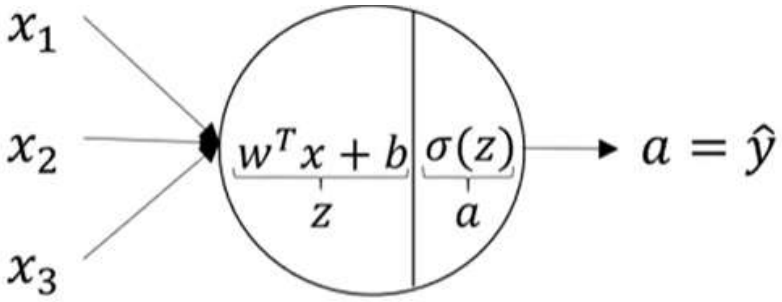
LogReg: 0<= σ(z) <=1

if σ(z) >=0.5, predict “y=1”
if σ(z) <0.5, predict “y=0”

Logistic (sigmoid) function



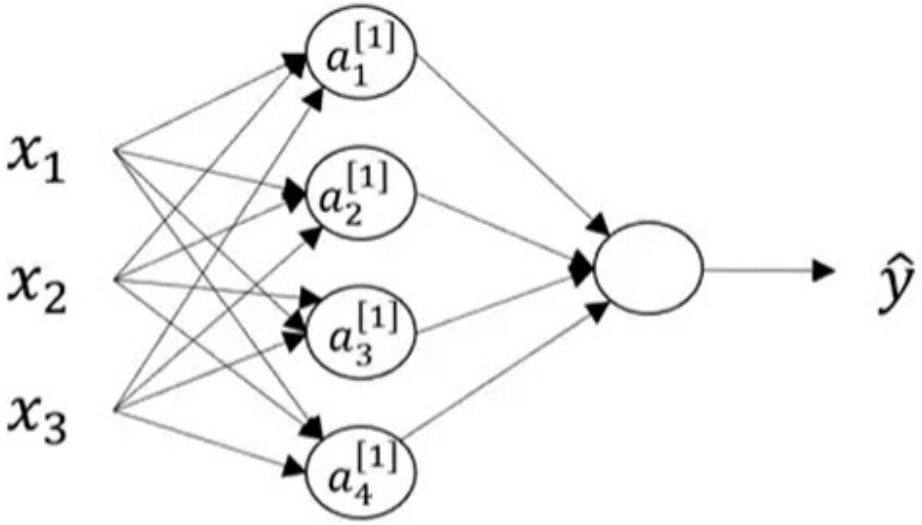
Logistic Regression vs Neural Network



<= Log Regression is a simple NN

$$z = w^T x + b$$

$$a = \sigma(z)$$



<= NN is a generalization of Log Reg

$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]},\ a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]},\ a_3^{[1]} = \sigma(z_3^{[1]})$$

$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]},\ a_4^{[1]} = \sigma(z_4^{[1]})$$

Forward & Backward Computations in DNN

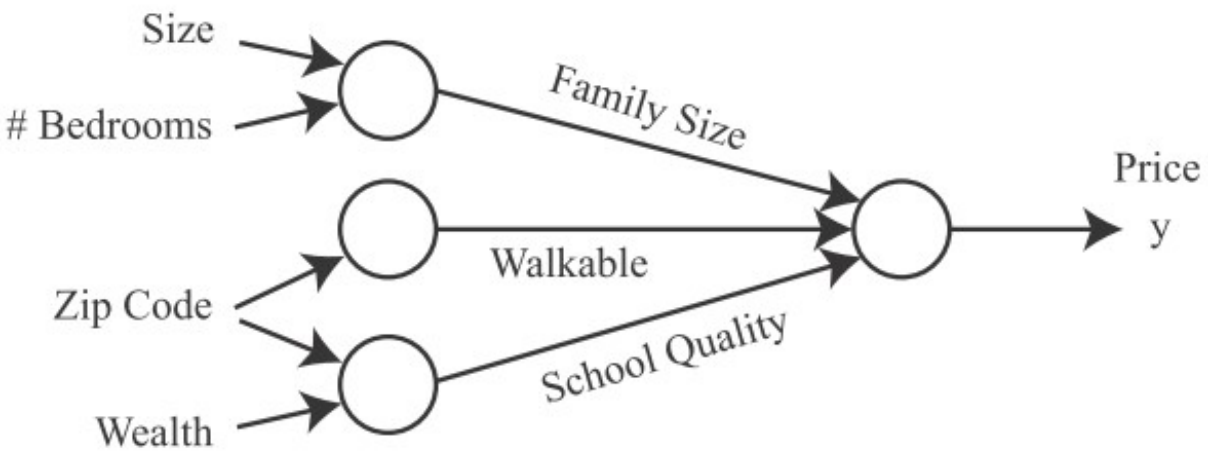
$$
\begin{aligned}
Z^{[1]} &= W^{[1]}X + b^{[1]} \\
A^{[1]} &= g^{[1]}(Z^{[1]}) \\
Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\
A^{[2]} &= g^{[2]}(Z^{[2]}) \\
&\vdots \\
A^{[L]} &= g^{[L]}(Z^{[L]}) = \hat{Y}
\end{aligned}
$$

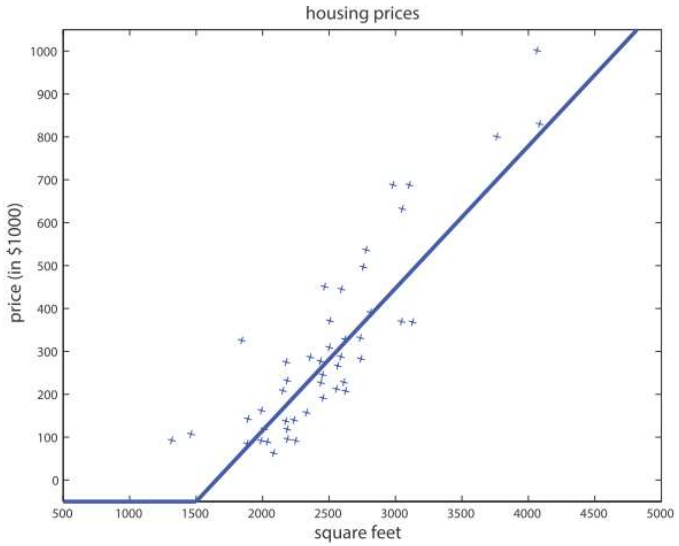
$$
\begin{aligned}
dZ^{[L]} &= A^{[L]} - Y \\
dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]^T} \\
db^{[L]} &= \frac{1}{m} np.\text{sum}(\text{dZ}^{[L]}, axis = 1, keepdims = True) \\
dZ^{[L-1]} &= dW^{[L]^T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\
&\vdots \\
dZ^{[1]} &= dW^{[L]^T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\
dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]^T} \\
db^{[1]} &= \frac{1}{m} np.\text{sum}(\text{dZ}^{[1]}, axis = 1, keepdims = True)
\end{aligned}
$$

Take $W^{[1]}, b^{[1]}, \ldots, W^{[L]}, b^{[L]}$ and reshape into a big vector θ.

It is questionable if human brains update their neural networks in a way similar to how ANNs learn (using backpropagation)…?

More Technical Inspiration for DNN





- **Given features: size, # of bedrooms, zip code, wealth**

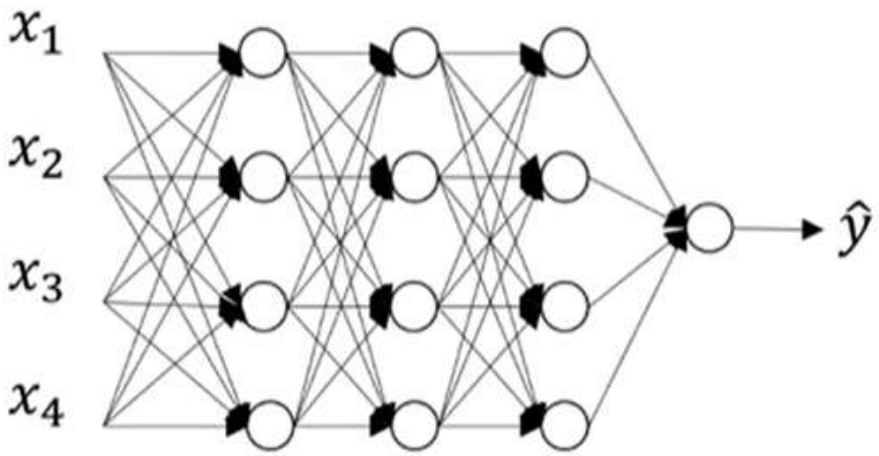
- We may decide that the house price depends on the max family size it can accommodate.
- Suppose the family size is a function of the size of the house and # of bedrooms.
- The zip code may provide additional information such as how walkable the neighborhood is (i.e., can you walk to the grocery store or do you need to drive everywhere).
- Combining the zip code with the wealth of the neighborhood may predict the quality of the local elementary school.

- **Derived features: family size, walkable, school quality**

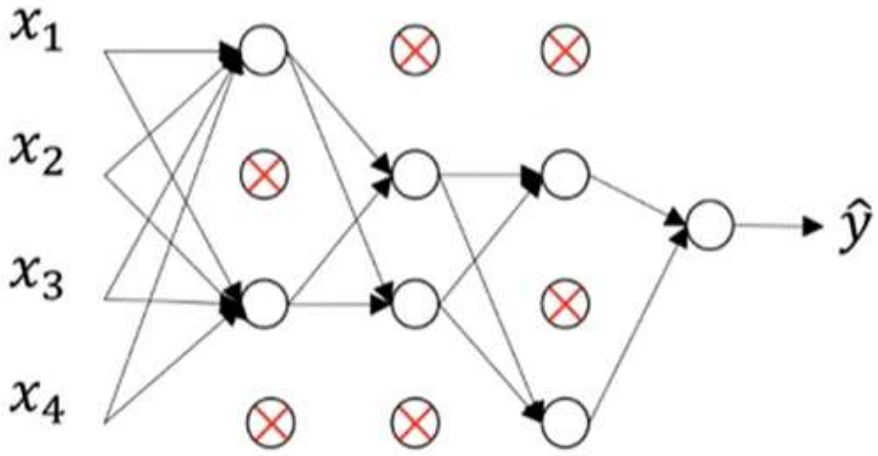
- We may conclude that **the house price ultimately depends on these derived features.**

- Instead of fitting a straight line, we wish to prevent negative housing prices by setting the absolute minimum price as zero => RELU activation function

Dropout regularization



Initial NN



NN after dropout step

Set some probability of eliminating a NN node (e.g. 0.5 chance of keeping each node and 0.5 chance of removing each node) => get a much smaller network and do back propagation training.
For each training example (or each mini batch), different nodes will be dropped out and different diminished network will be trained.

It seems like a slightly crazy technique, but this actually works.

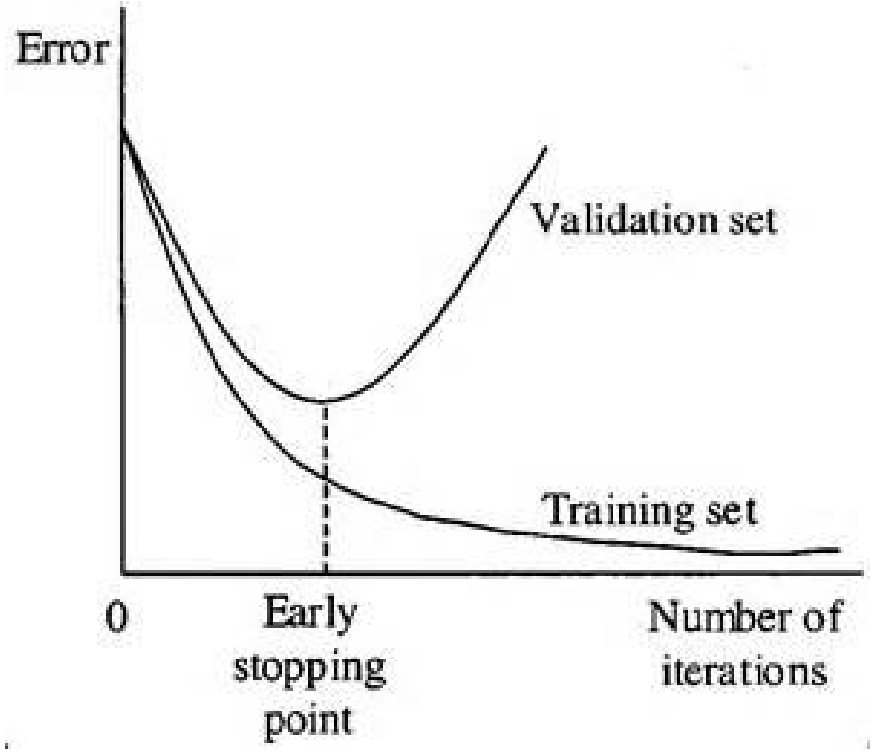
Dropout Intuition: cannot rely on any feature (giving a high weight),
have to spread out weights, distribute the importance between all features .

NO drop out at test stage, only during the training stage !!!

Data Augmentation

Generate more data has a positive effect on the over-fitting problem => data augmentation can be seen as another regularization method

Early Stopping

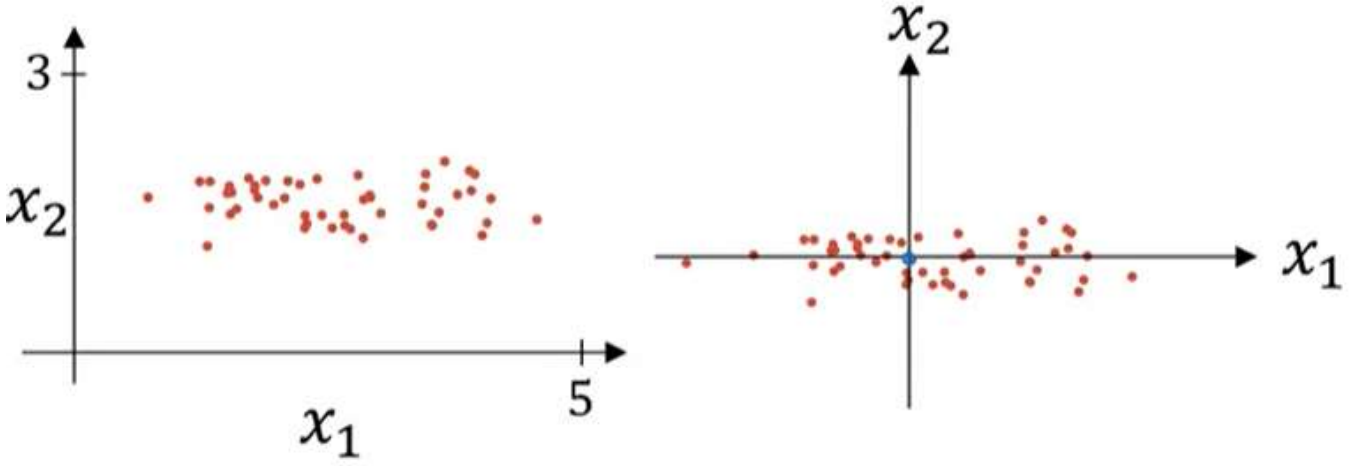


Early Stopping fights the overfitting problem =>
can be seen as another regularization method

Stopping the training is not the best solution, because the cost function was not minimized.

It is better to use L2/L1 regularization instead of early stopping !!!

Normalizing training data



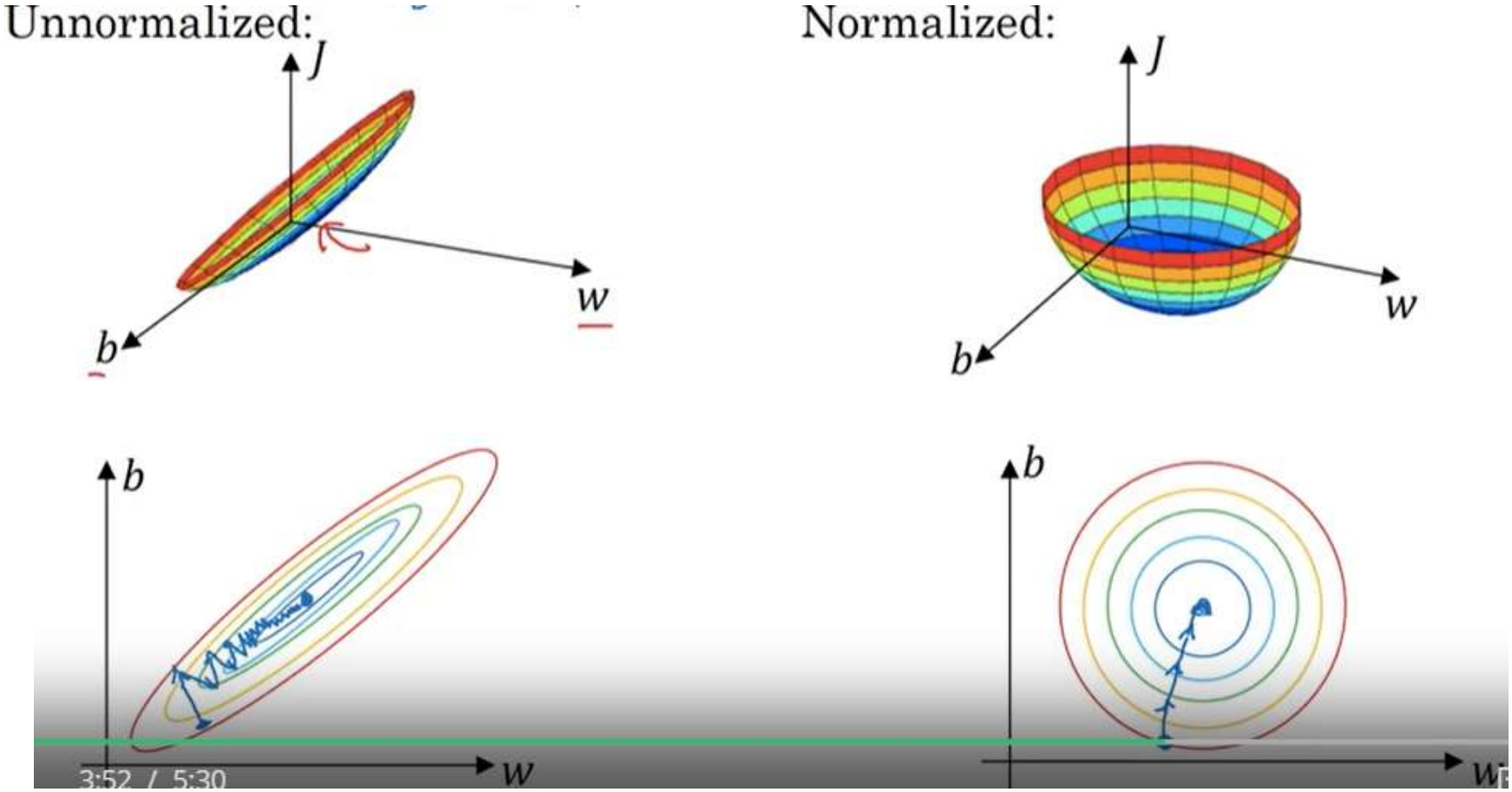
Mean normalization:

- Subtract the mean value of each feature from the values of this feature
- Divide by the standard deviation

Weight initialization helps to overcome the vanishing gradients.

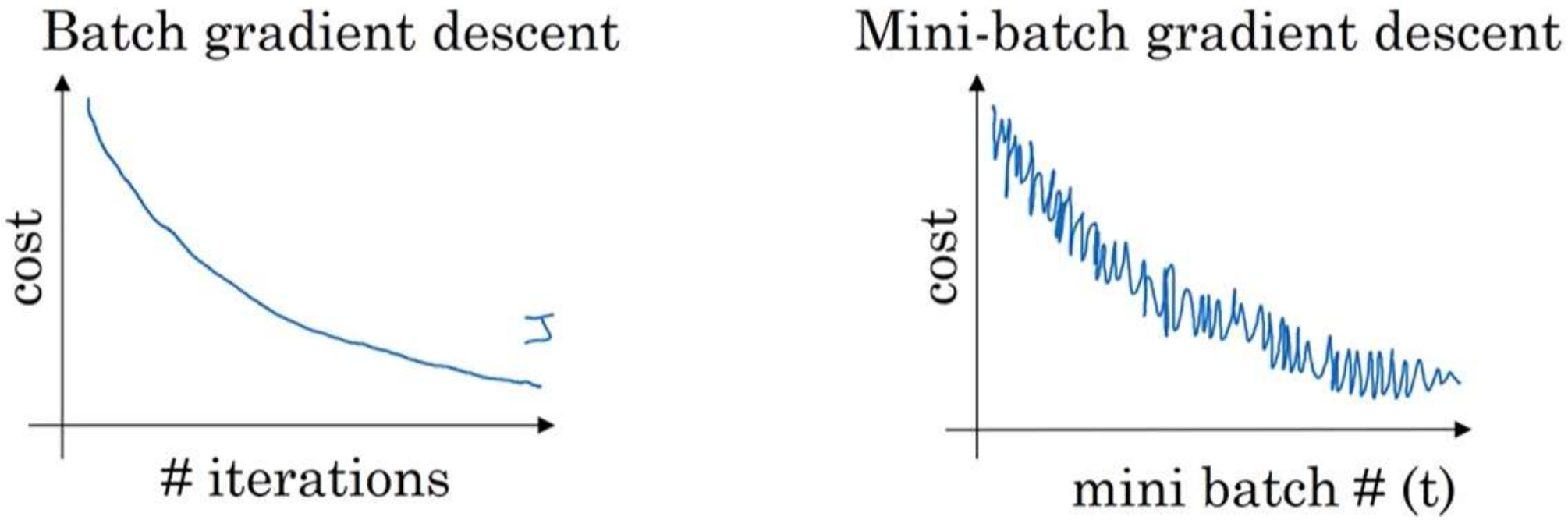
Why normalize inputs ?

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$$



If the features come from different scales, it is important to normalize them, the learning algorithm is expected to learn faster
(e.g. the cost function *J(w,b)* will converge faster)

Training with mini-batch gradient descent



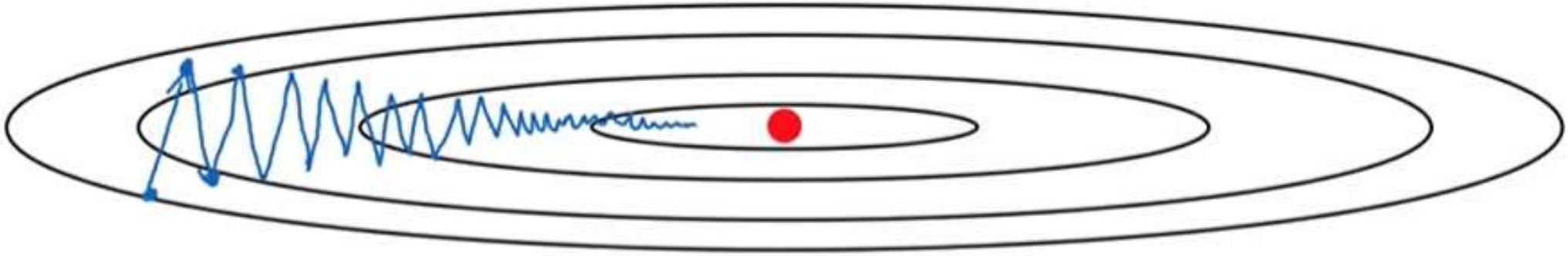
m - set of training examples

- if m<= 2000 (small data set) => Batch Gradient Descent
- if m =1 => Stochastic Gradient Descent

- if m >> 2000 (big data) => long duration of each iteration => use mini-batch
- In practice mini-batch size is typically m= 64, 128, 256, 512, 1024

Make sure mini batch fit the CPU/GPU memory

Gradient Descent with momentum



$$W^{(r)} = W^{(r-1)} - \alpha \frac{\partial J}{\partial W} + \beta\left(W^{(r-1)} - W^{(r-2)}\right)$$

alfa – learning rate
beta – momentum term
Hyper-parameters: beta , alfa, e.g. beta=0.9 is a good default value

Vertical oscillations => slower learning
With momentum term smaller vertical oscillations => faster learning

Cost Function Optimization Methods

Cost Functions:
Cross entropy (categorical, binary) – for classification problems
Mean Squared Error (MSE) - for regression problems

Optimization methods:
- Gradient Descent
- Gradient Descent with momentum
- Backpropagation – the most used historically

- **RMSprop** (Root Mean Square propagation) => speed up Gradient Descent.

Extra terms to slow down the oscillations into the vertical direction.

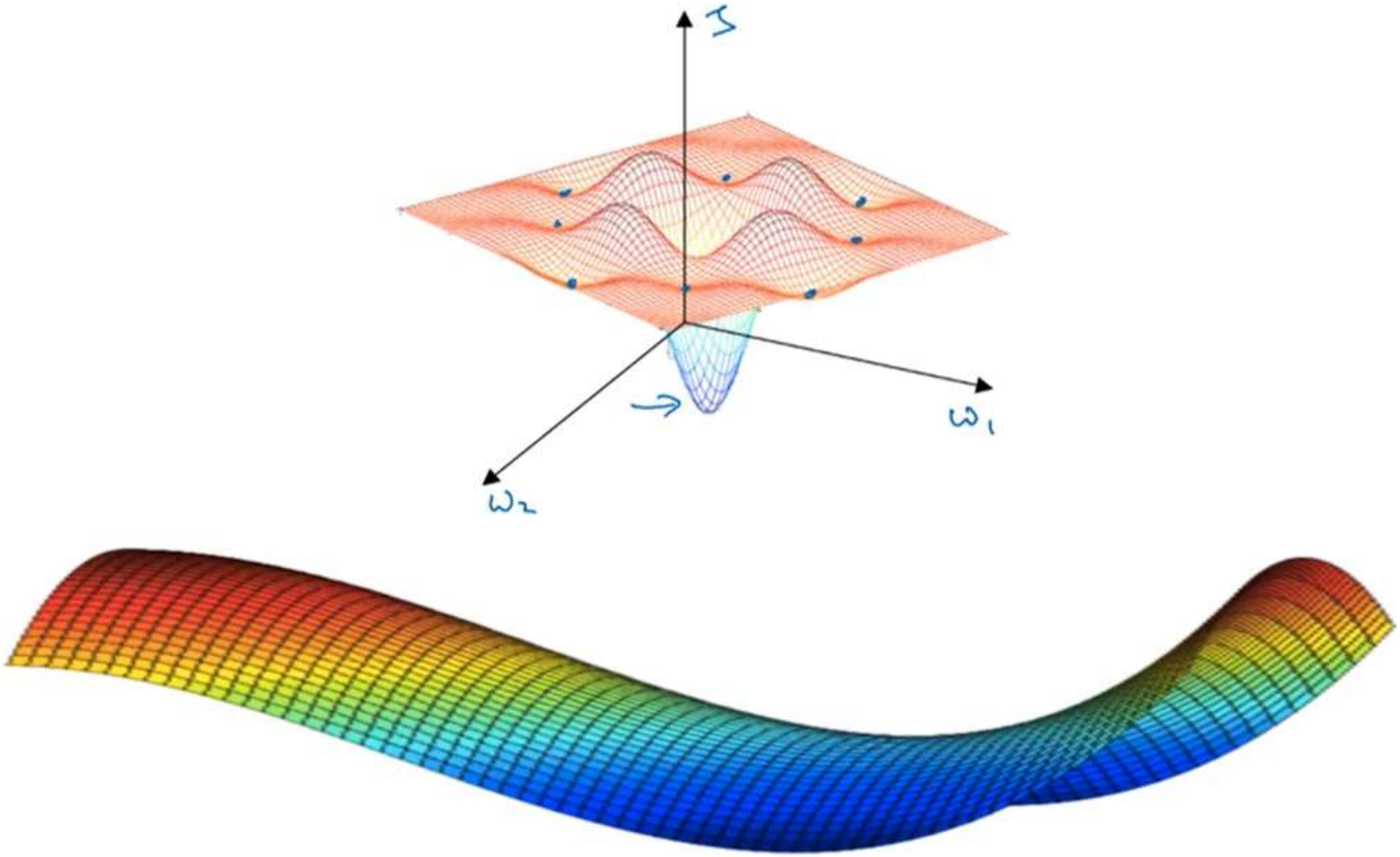
- **Adam** (Adaptive Moment Estimation) combine Gradient Descent with momentum and RMSprop.

- Marquardt- Levenberg

- Rprop

- Quickprop

The problem with local optima or plateaus



The local optima (top image) is a bad problem in ML but it is much more likely to see saddle points (bottom image) than local optimum.
Plateau is a region where gradient is 0 or near zero for a long time, the surface is quite flat and it really slows down learning.

ML Hyper-Parameters

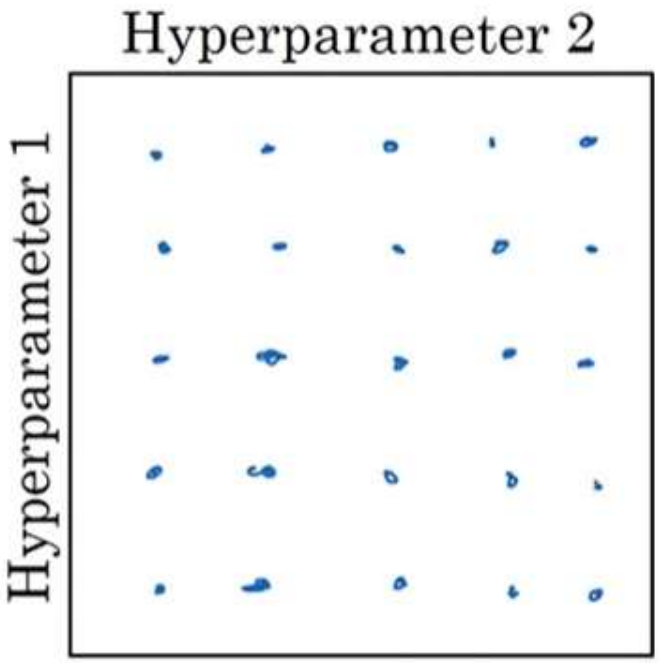
Applied ML is a very empirical process

Need to choose many hyper-parameters:

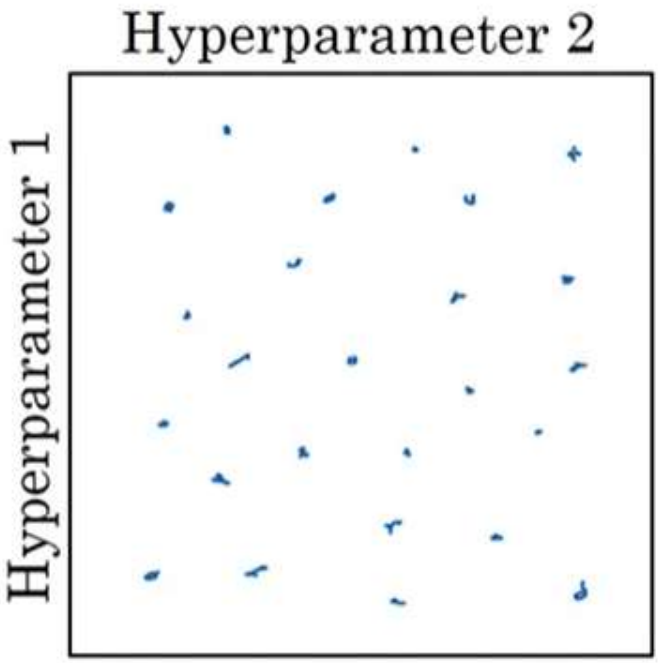
- Learning rate (**the most important**)
- Number of hidden units (very important)
- Mini-batch size (very important)
- Momentum (very important)
- Number of iterations
- Number of hidden layers
- Choice of activation functions
- Regularization parameters (alfa, etc.)
- Learning rate decay

Hyper-Parameter Tuning

Grid of values



Random values



Recommended - try random values, not a grid
For learning rate log scale is better than a linear scale.
alfa =[0.0001,…1] choose [10^a ,…10^b]

from sklearn.model_selection import RandomizedSearchCV, GridSearchCV

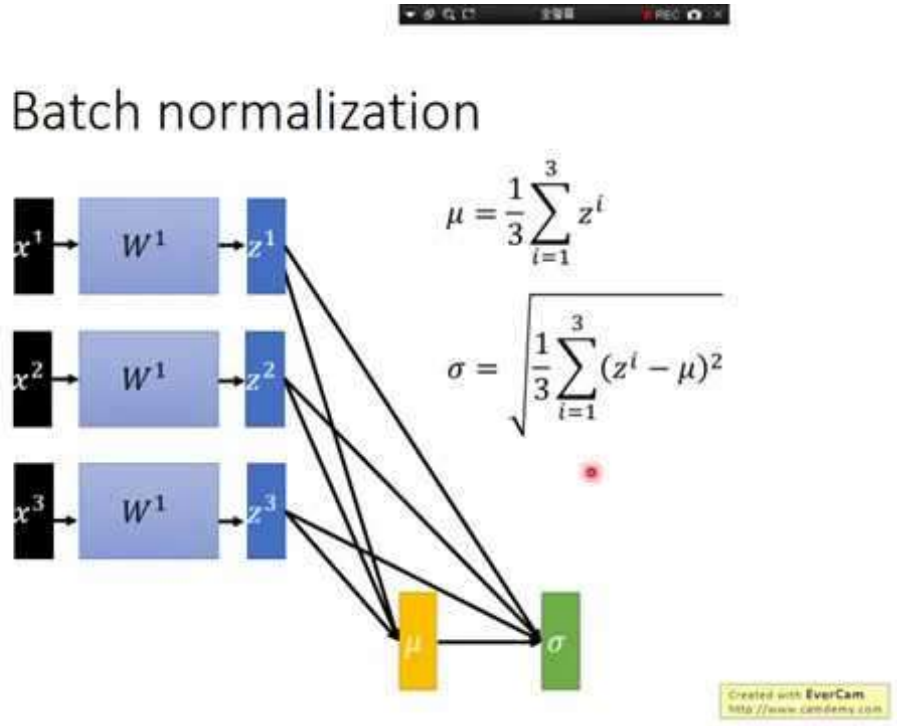
Batch Normalization (BN)

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z^{(i)} - \mu)^2$$

$$z^{(i)}_{\text{norm}} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma z^{(i)}_{\text{norm}} + \beta$$

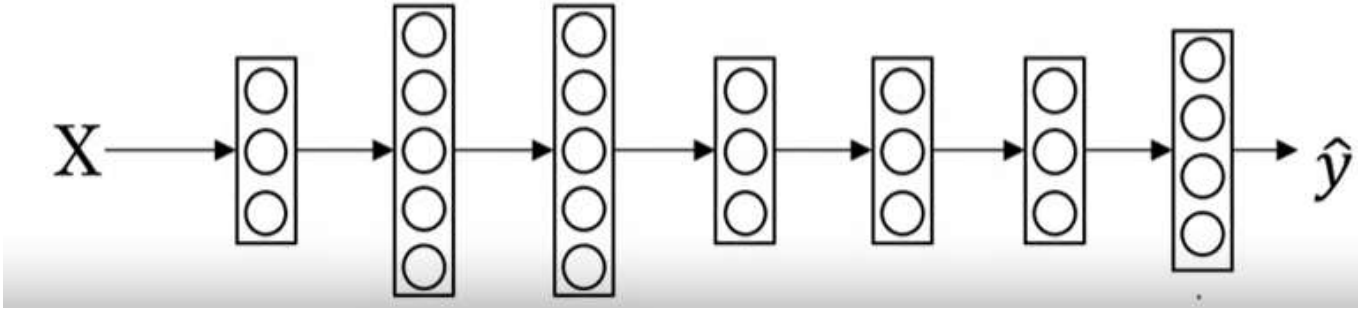


- If standard deviation =0, it is better to have some small value epsilon.
- z_norm values have mean=0, standard deviation=1, but it may be very restrictive, therefore with gama and beta (trainable parameters) the model can learn different distributions.
- z_tilda are the batch normalized values used for the next computations.
- Mean and standard deviation are computed separately for each mini-batch.

In TensorFlow: tf.nn.batch_normalization

Multiclass classification - Softmax layer

3 1 2 0 3 2 0 1



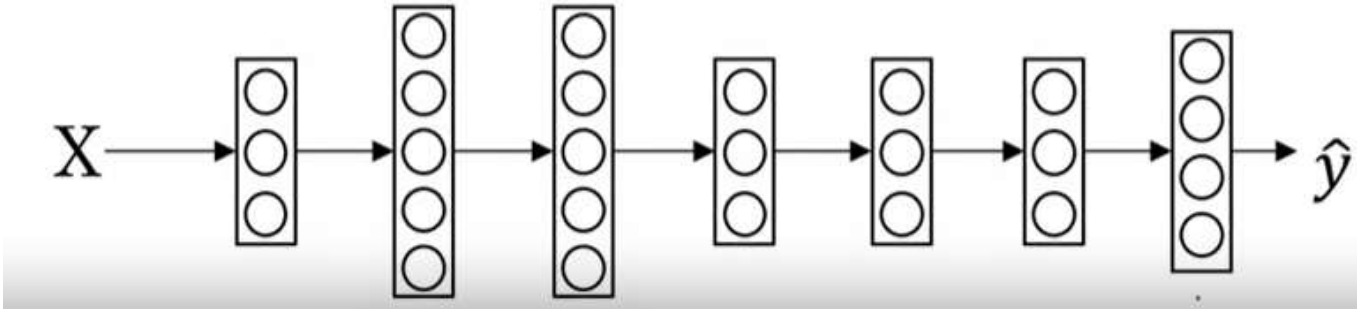
Example:

$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

Softmax activation => function

$$g^{[L]}(z^{[L]}) = \begin{bmatrix} e^5/(e^5+e^2+e^{-1}+e^3) \\ e^2/(e^5+e^2+e^{-1}+e^3) \\ e^{-1}/(e^5+e^2+e^{-1}+e^3) \\ e^3/(e^5+e^2+e^{-1}+e^3) \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

Multiclass classification - Softmax layer



Softmax regression is a generalization of logistic regression to more than two classes.

The name softmax contrasts the so called hard max function that will look at the elements of Z and just put 1 in the position of the biggest element of Z and 0s everywhere else.
softmax is a more gentle mapping from Z to these probabilities.

Single Number Evaluation Metric

Classifier	Precision (p)	Recall (r)	F1-Score
A	95%	90%	92.4 %
B	98%	85%	91.0%

F1 is the "average" of P and R, aka "Harmonic mean".
- F1=2*(Recall * Precision) / (Recall + Precision)

Dev set (development/validation set) + single real number evaluation metric will take less time to develop the model

- True Positive Rate (TPR), Sensitivity, Recall

- True Negative Rate (TNR), Specificity

$$TPR = \frac{TP}{TP + FN} \qquad TNR = \frac{TN}{TN + FP} \qquad \text{Precision} = \frac{TP}{TP + FP}$$

Optimizing Metric + some Constraints

Classifier	Accuracy	Running time
A	90%	80 ms
B	92%	95 ms
C	95%	1 500 ms

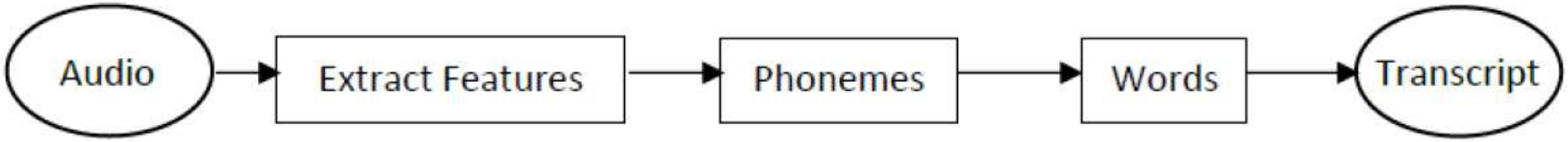
Maximize Accuracy + **Running time <= 1000 ms**.
(satisficing metric + constraint)

Example: Wake-words/trigger words for voice control devices:
Amazon Echo => wake up by saying Alexa
Google => OK
Apple => Hey Siri
When someone says one of these trigger words, how likely is to wakeup the device, but also care about the number of false positives.

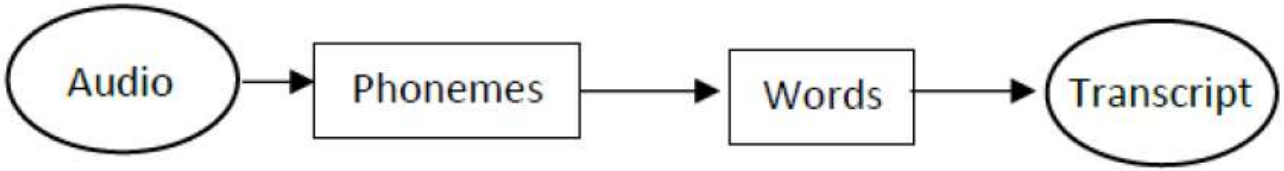
- Maximize Accuracy (optimizing metric)
- such that <= **1 false positive every 24 hours** (satisficing metric+constraint)

End-to-End Deep Learning

The traditional way - small data set



The hybrid way - medium data set



The End-to-End deep learning way – large data set

In 5G+ (wireless, optical) communications

End-to-End Deep Learning

In 5G+ communications