EEML WORKSHOPS

Introduction to Deep Learning

Razvan Pascanu

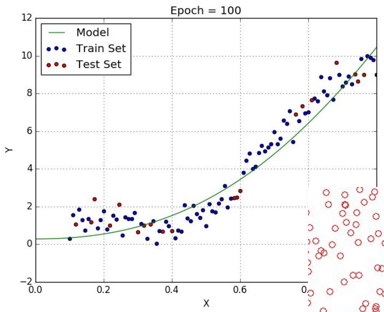
DeepMind

TMLW @ Timisoara, Romania Feb 2019

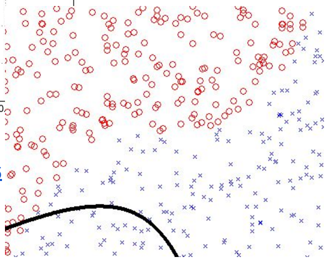
Primer on Machine Learning

Machine Learning

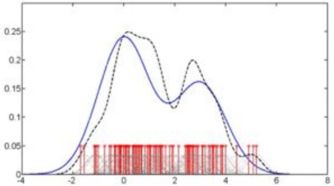
Supervised Learning | **Unsupervised Learning** | **Reinforcement Learning**



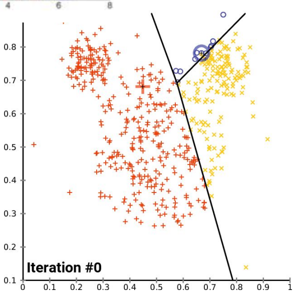
https://towardsdatascience.com/5-types-of-regression-and-their-properties-c5e1fa12d55e



https://www.mql5.com/en/articles/584



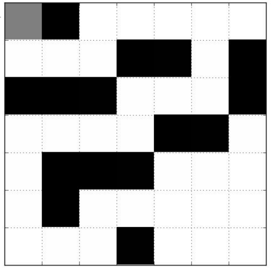
https://en.wikipedia.org/wiki/Density_estimation



https://en.wikipedia.org/wiki/K-means_clustering

https://medium.com/@awjuliani/super-simple-reinforcement-learning-tutorial-part-2-ded33892c724

https://www.samyzaf.com/ML/rl/qmaze.html

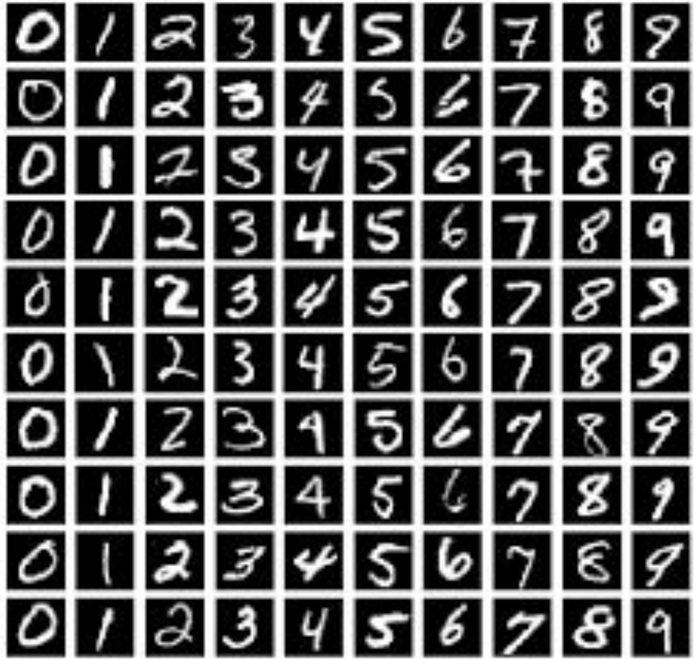


Supervised Learning (Classification)

Example of a dataset

Labels (y): **0 1 2 3 4 5 6 7 8 9**

Inputs (x):



Terminology:

- labels/targets
- Output/predictions
- input/input features
- Instance or example
- datasets

Function approximator

- Let $\mathcal{X}$ denote the space of input values
- Let $\mathcal{Y}$ denote the space of output values
- Given a data set $D \subset \mathcal{X} \times \mathcal{Y}$, find a function:

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

  such that $h(\mathbf{x})$ is a *"good predictor"* for the value of y.
- h is called a *hypothesis*
- Problems are categorized by the type of output domain
  - If $\mathcal{Y} = \mathbb{R}$, this problem is called *regression*
  - If $\mathcal{Y}$ is a categorical variable (i.e., part of a finite discrete set), the problem is called *classification*
  - In general, $\mathcal{Y}$ could be a lot more complex (graph, tree, etc), which is called *structured prediction*

Function approximator

A parameterized function is a function:

$$h : \theta \times \mathcal{X} \rightarrow \mathcal{Y}$$

for example a linear function of the form

$$h(w, x) = wx$$

Learning then boils down to finding *the best* θ to minimize the distance between prediction and targets

$$\arg\min_{\theta} L(\theta) = \arg\min_{\theta} \mathbb{E}\left[dist(h(\theta, x_i), y_i)\right]$$

Right distance for the problem?

$$\arg\min_{\theta} L(\theta) = \arg\min_{\theta} \mathbb{E}\left[dist(h(\theta, x_i), y_i)\right]$$

- Rely on a probabilistic interpretation of the model $p(y|x)$

$$\underbrace{p(\theta|D)}_{\text{posterior}} = \frac{\overbrace{p(\theta)}^{\text{prior}}\overbrace{p(D|\theta)}^{\text{likelihood}}}{\underbrace{p(D)}_{\text{normalizing constant}}}$$

Bayes Rule

Right distance for the problem?

$$\arg\min_{\theta} L(\theta) = \arg\min_{\theta} \mathbb{E}\left[dist(h(\theta, x_i), y_i)\right]$$

Equivalently:

$$p(\theta|D) \propto p(\theta)p(D|\theta)$$

Assuming uniform prior, we get:

$$p(\theta|D) \propto p(D|\theta)$$

$$\underbrace{p(\theta|D)}_{\text{posterior}} = \frac{\overbrace{p(\theta)}^{\text{prior}}\overbrace{p(D|\theta)}^{\text{likelihood}}}{\underbrace{p(D)}_{\text{normalizing constant}}}$$

Bayes Rule

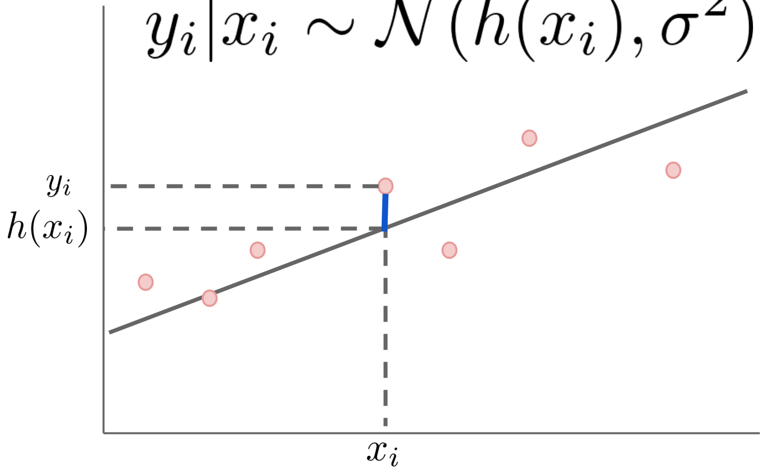
Loss functions (Mean Square Error)

Under uniform prior we have:

MAP = MLE

MAP: Maximum A Priori estimate
MLE: Maximum Likelihood Estimate

$$y_i | x_i \sim \mathcal{N}(h(x_i), \sigma^2)$$



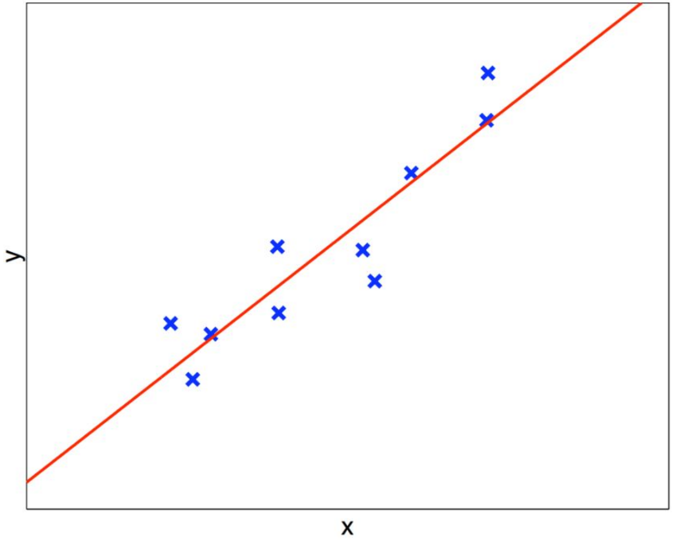
$$p(D|\theta) = \prod_i \exp - \frac{1}{2\sigma^2}(h(x_i) - y_i)^2 = \exp - \frac{1}{2\sigma^2} \sum_i (h(x_i) - y_i)^2$$

$$\arg\max_\theta p(D|\theta) = \arg\min_\theta L = \sum_i [h(x_i) - y_i]^2$$

Picking the right hypothesis class

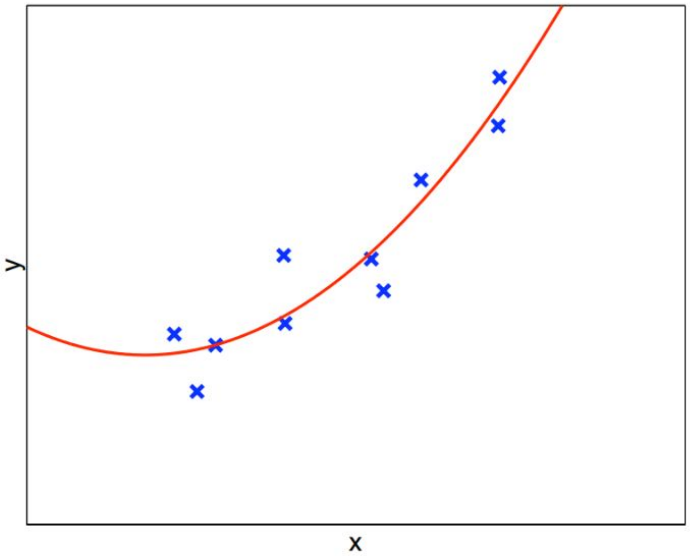
Example: Data and best linear hypothesis

$$y = 1.60x + 1.05$$



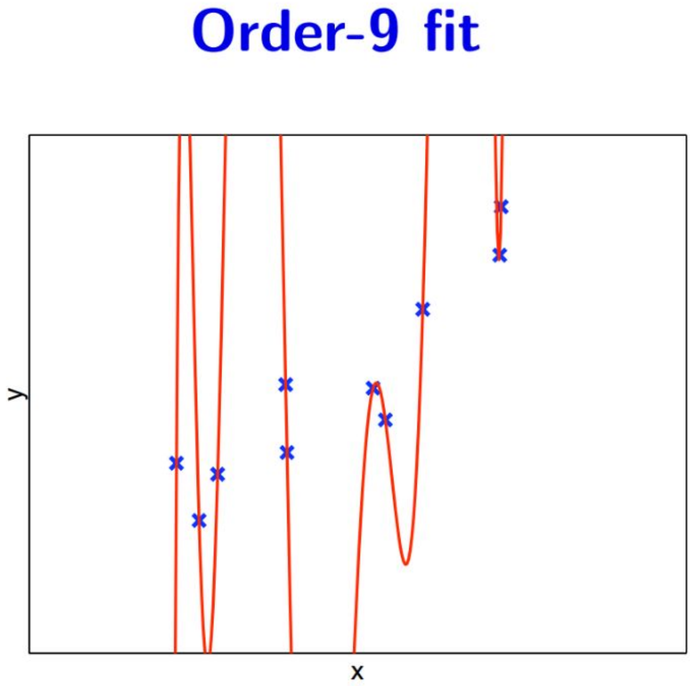
Picking the right hypothesis class

Order-2 fit



Is this a better fit to the data?

Picking the right hypothesis class



Is this a better fit to the data?

Overfitting/Underfitting

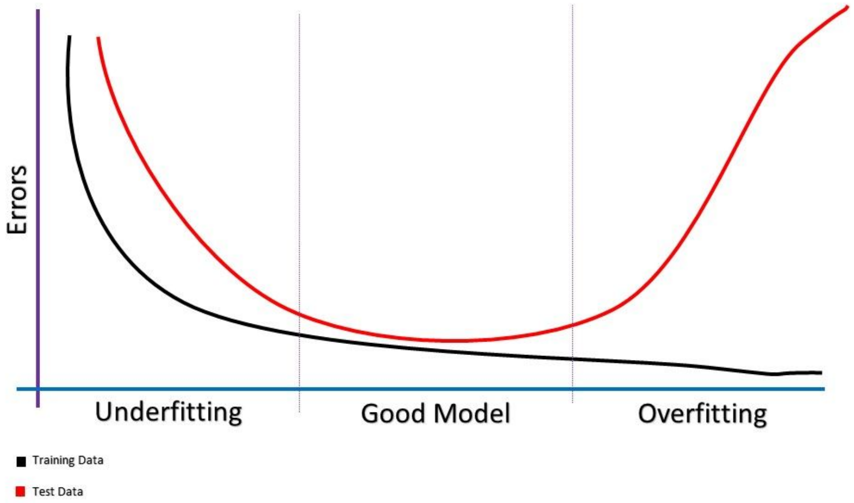
- *We want to be able* ***to generalize***

Use

Training set: data used for finding the right parameters

Validation set: data used to estimate true loss on unseen data

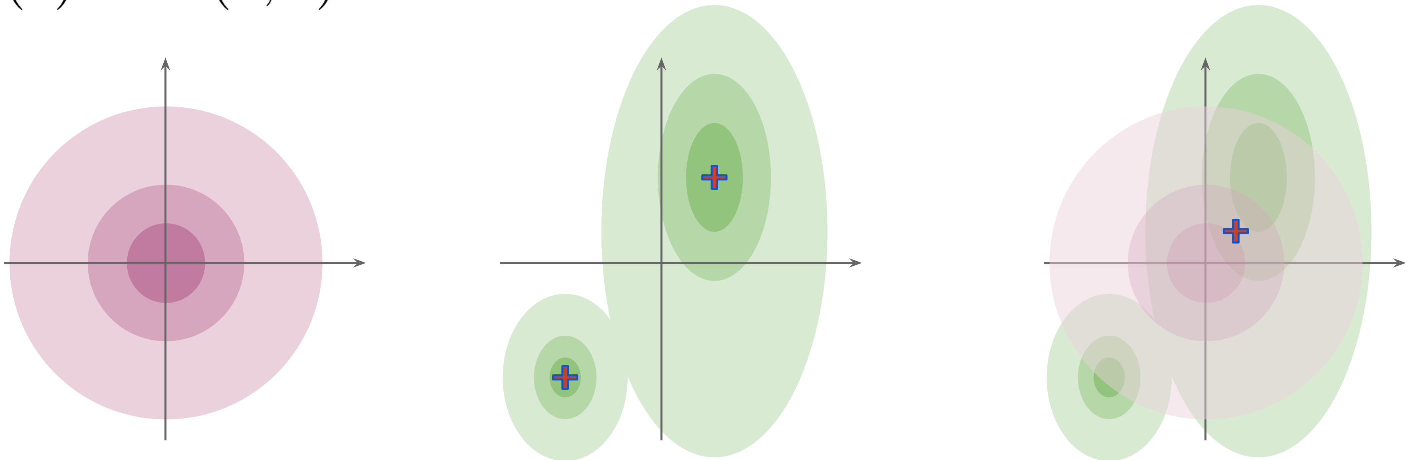
Learning is about minimizing an intractable function via optimizing a tractable approximation of it



Role of the prior - Regularization

- Prior provides a mechanism to introduce knowledge in the learning problem
- It restricts the search space for the parameters of the model
- Ends up being an additive gradient field to the one generated by MLE

$$p(\theta) = \mathcal{N}(0, 1)$$



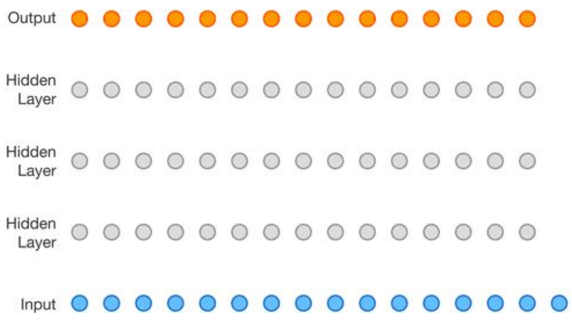
$$\|\theta\|^2 \qquad \sum_i [h(x_i) - y_i]^2 \qquad \gamma\|\theta\|^2 + \sum_i [h(x_i) - y_i]^2$$

Unsupervised learning

Many topics under the umbrella:

Clustering
Dimensionality reduction
Density Estimation
Metric learning
Generative models

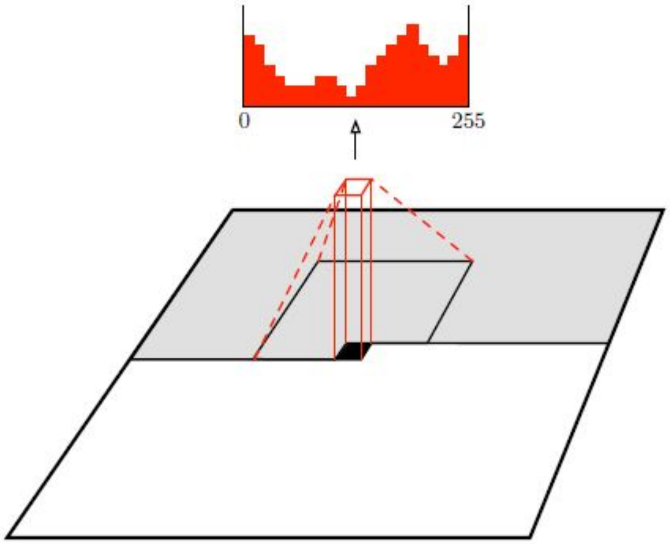
Generative models: i) autoregressive



Wavenet [van den Oort et al.]

Figure 1. Image completions sampled from a PixelRNN.

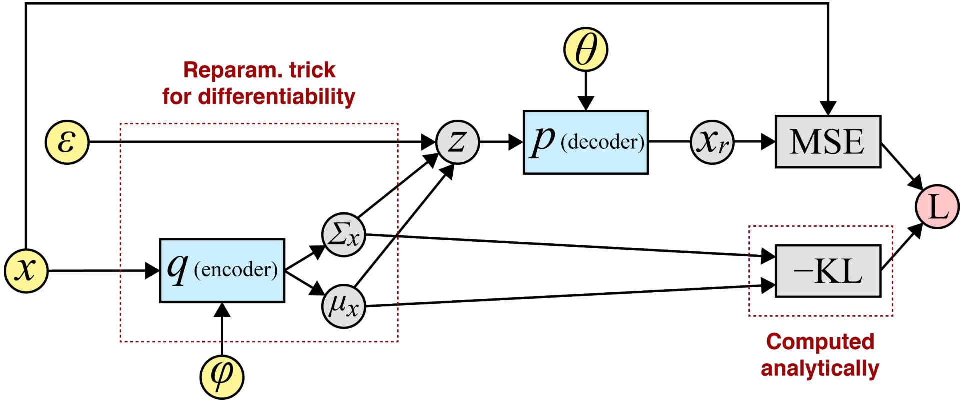
PixelCNN [van den Oort et al.]



https://wiki.math.uwaterloo.ca/statwiki/index.php?title=STAT946F17/Conditional_Image_Generation_with_PixelCNN_Decoders

Generative models: ii) VAE

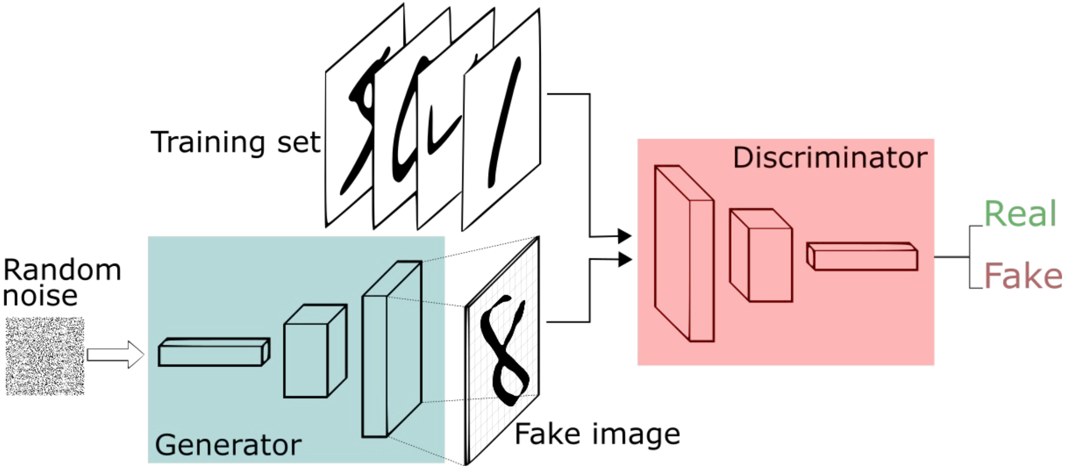
https://blog.openai.com/generative-models/



http://gregorygundersen.com/blog/2018/04/29/reparameterization/

Generative models: iii) GANs

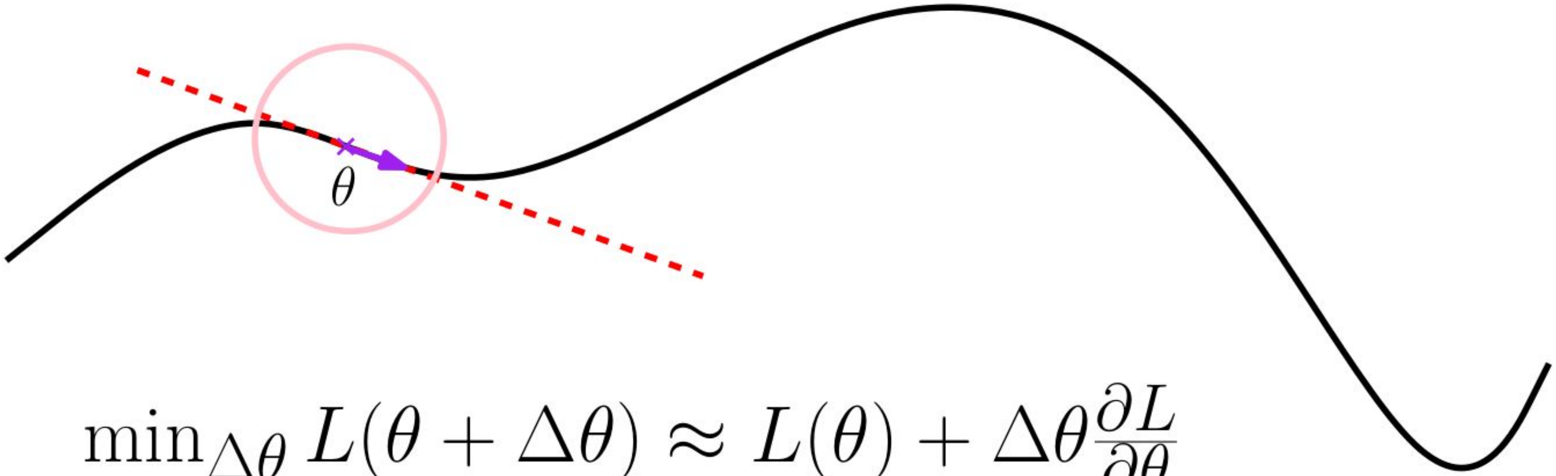
https://blog.openai.com/generative-models/



https://skymind.ai/wiki/generative-adversarial-network-gan

How do we search for theta?

- One approach is following the gradient (gradient descent)

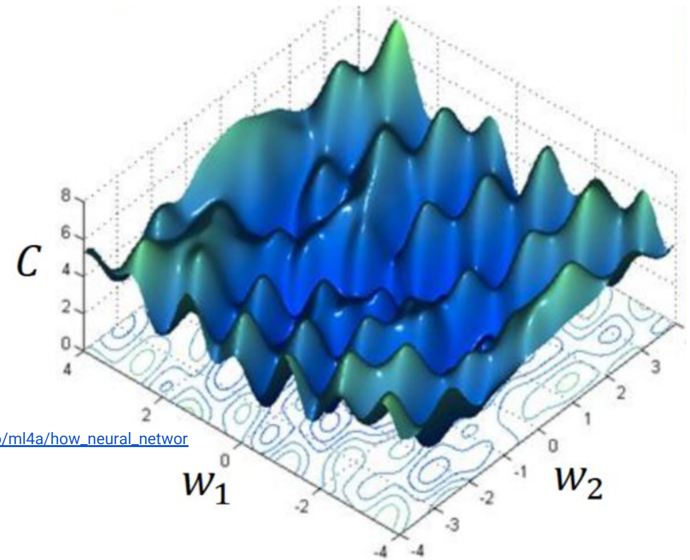


$$\min_{\Delta\theta} L(\theta + \Delta\theta) \approx L(\theta) + \Delta\theta \frac{\partial L}{\partial \theta}$$

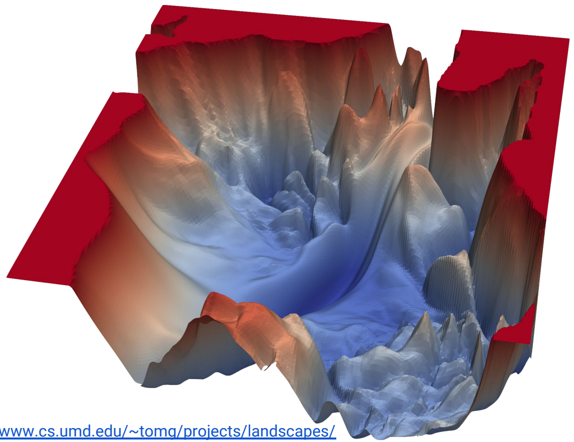
$$\text{s.t. } |\Delta\theta| < \epsilon$$

How do we search for theta?

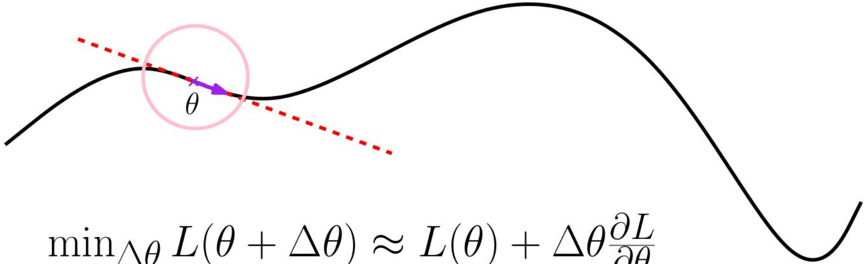
- Are unconstrained models impossible to optimize?



https://ml4a.github.io/ml4a/how_neural_networks_are_trained/



https://www.cs.umd.edu/~tomg/projects/landscapes/



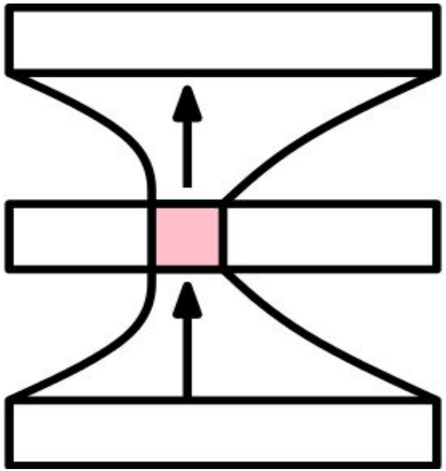
$$\min_{\Delta\theta} L(\theta + \Delta\theta) \approx L(\theta) + \Delta\theta \frac{\partial L}{\partial \theta}$$
$$\text{s.t. } |\Delta\theta| < \epsilon$$

Deep Neural Networks?

$$ReLU(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

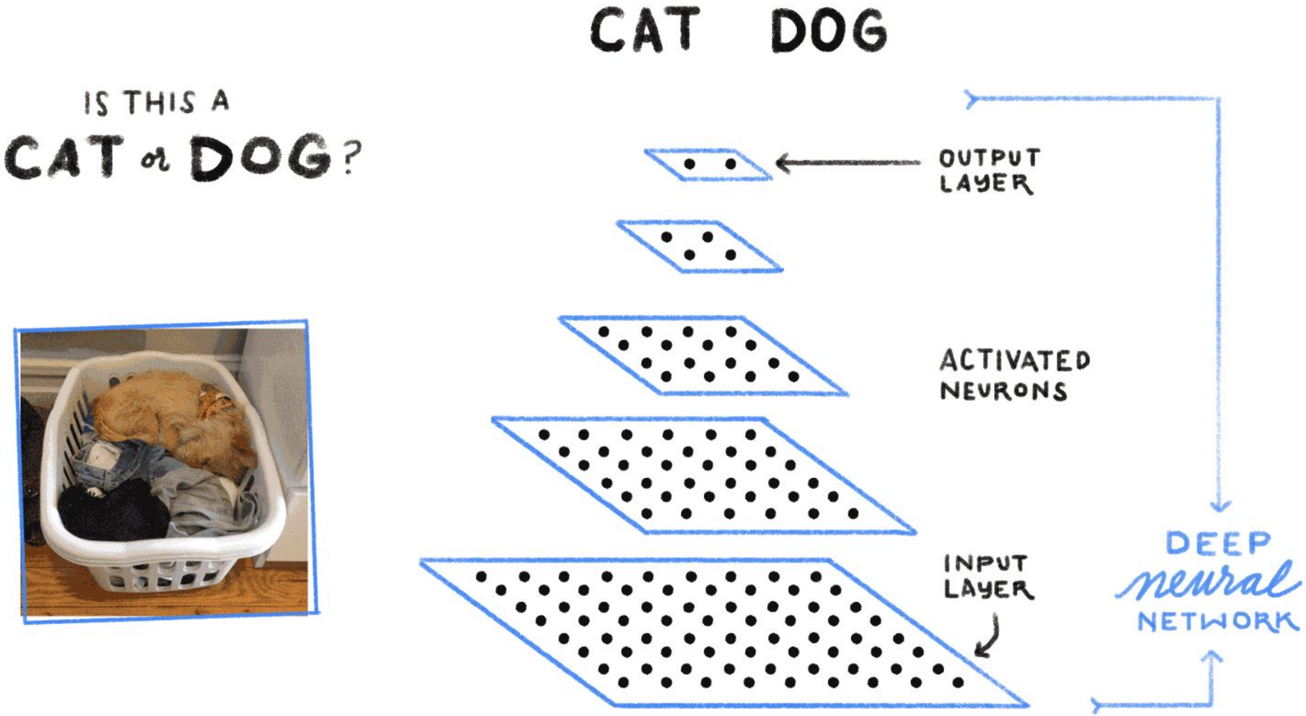
$$h(x) = l3(l2(l1(x)))$$

$$l_k = ReLU(W_k l_{k-1} + b_k)$$



- Other non-linearities are possible
- Why have a non-linearity?

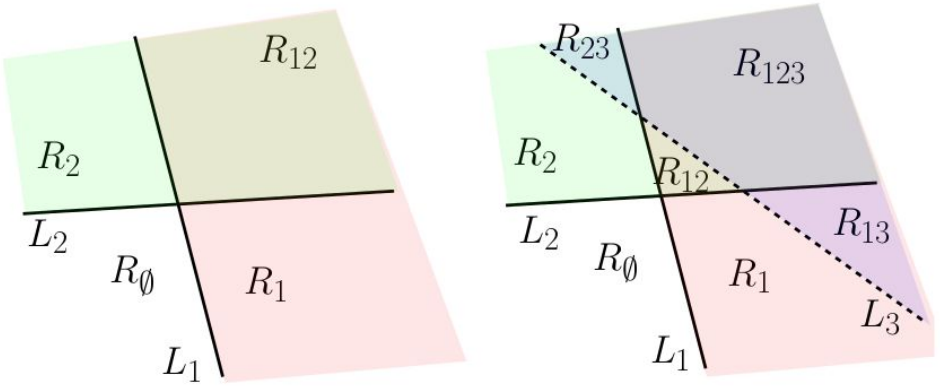
Deep Neural Networks?



https://medium.com/datadriveninvestor/how-a-computer-looks-at-pictures-image-classification-a4992a83f46b

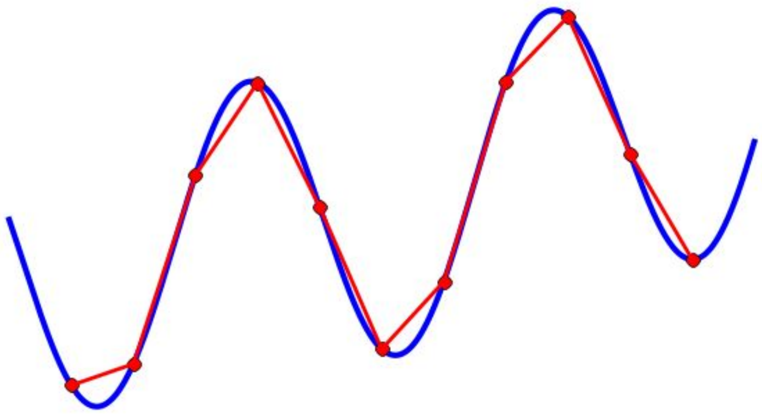
ReLU networks

Single hidden layer ReLU neural network



[Guido Montufar, Razvan Pascanu, Kyunghyun Cho & Yoshua Bengio, *On the number of linear regions of Deep Neural Networks*, NIPS 2014](#)

ReLU networks



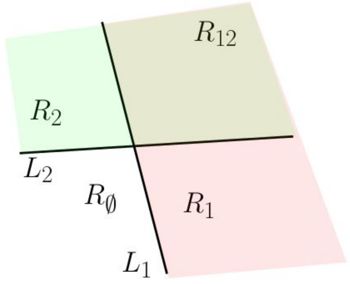
We know Neural Nets are universal approximators of any functions!

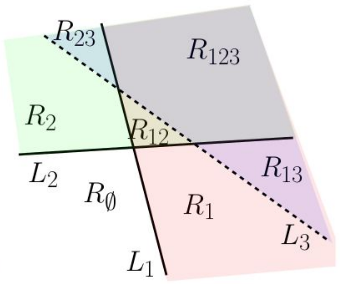
But is it enough?

Why “deep” in “deep networks”

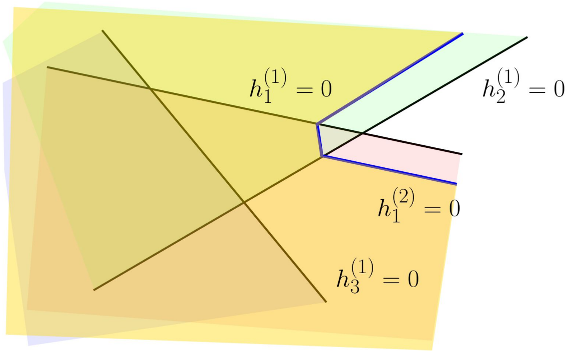
ReLU networks: representation

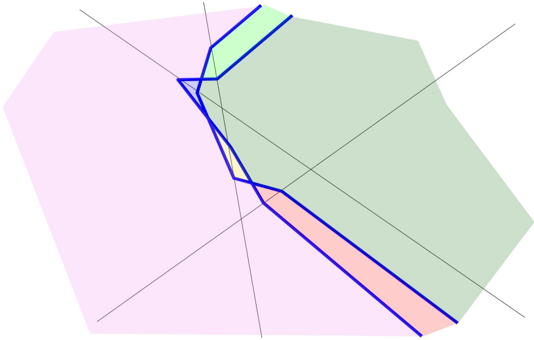
Single hidden layer ReLU neural network





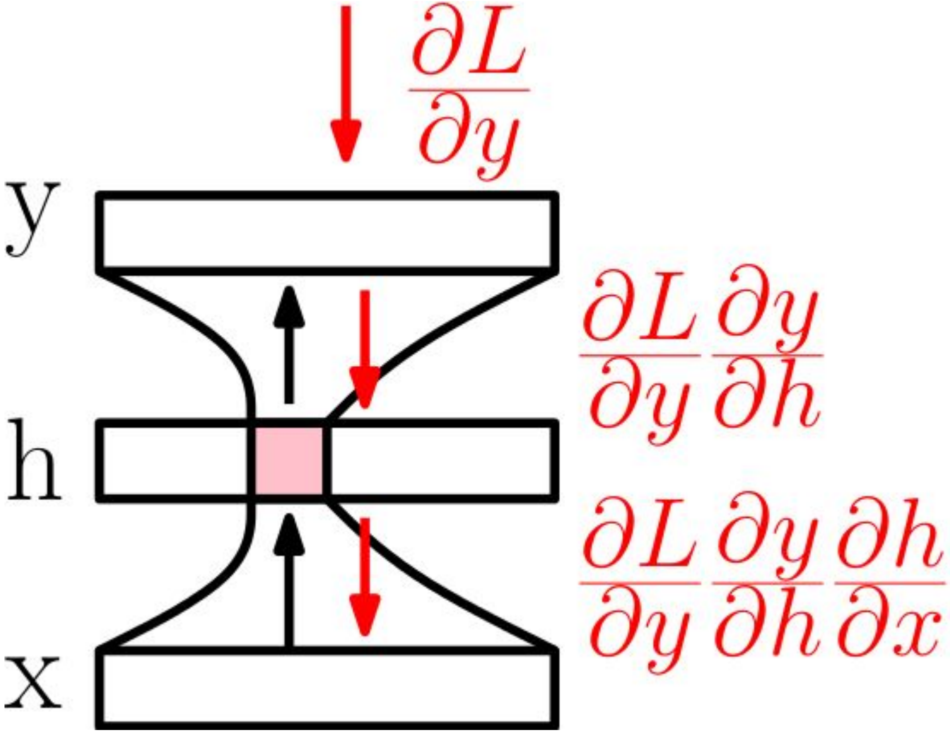
Two hidden layer ReLU neural network

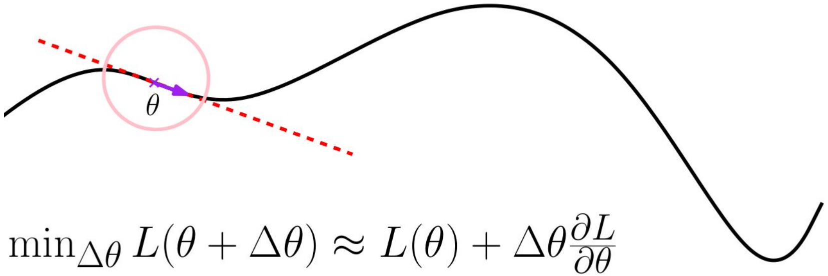




[Guido Montufar, Razvan Pascanu, Kyunghyun Cho & Yoshua Bengio, *On the number of linear regions of Deep Neural Networks*, NIPS 2014]

ReLU networks: learning



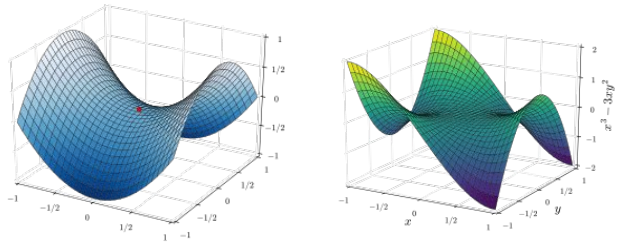


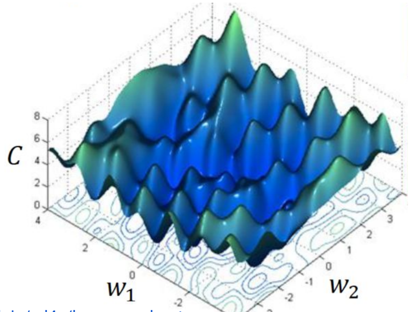
$$\min_{\Delta\theta} L(\theta + \Delta\theta) \approx L(\theta) + \Delta\theta \frac{\partial L}{\partial \theta}$$
$$\text{s.t. } |\Delta\theta| < \epsilon$$

$$\theta_t = \theta_{t-1} - \eta \frac{\partial L}{\partial \theta}$$

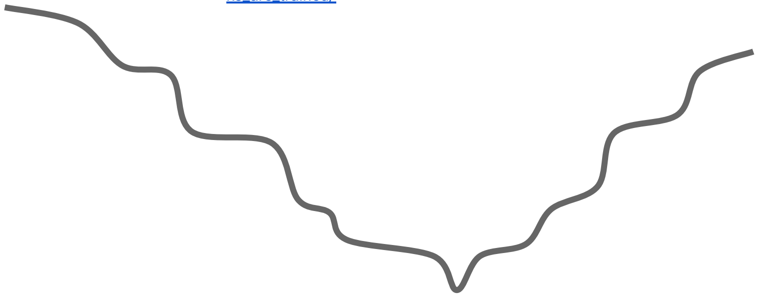
ReLU networks: learning?

How do we learn?





https://ml4a.github.io/ml4a/how_neural_networks_are_trained/



- Somehow over-parameterization makes the loss surface well behaved!

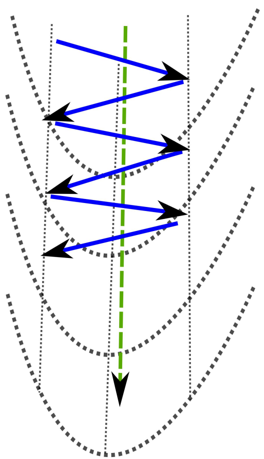
Yann Dauphin, et, al , *Identifying and attacking the saddle point problem in high-dimensional nonconvex optimization*

New view on the surface error of deep learning?

ReLU networks: learning?

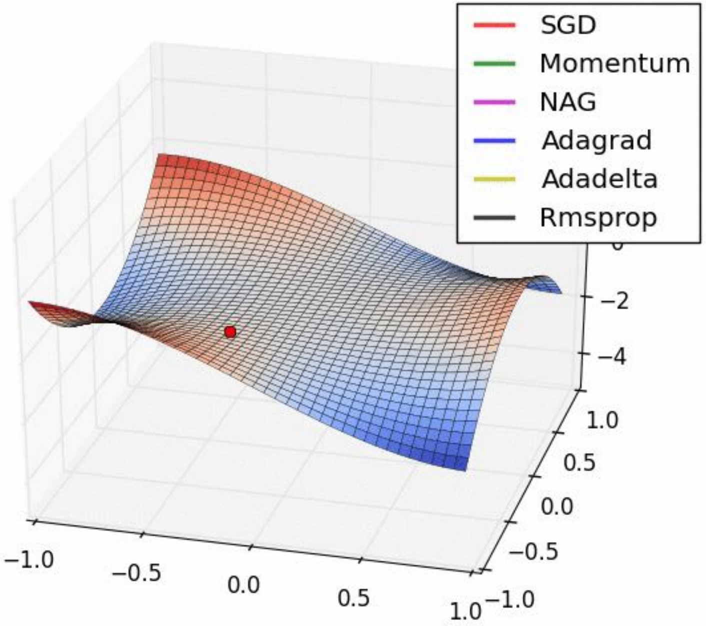
- Somehow over-parameterization makes the loss surface well behaved!

Still many issues remain!



- Address issues around flat regions
    - RMSPROP/ADAM account for speed of change
    - Momentum for consistency in movement
    - Fixed step in function change

https://devblogs.nvidia.com/introduction-neural-machine-translation-gpus-part-2/sgd_viz/



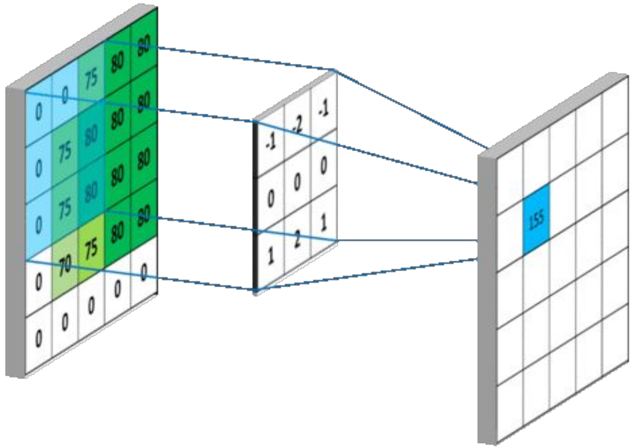
Structure in models

Convolutional Neural Networks

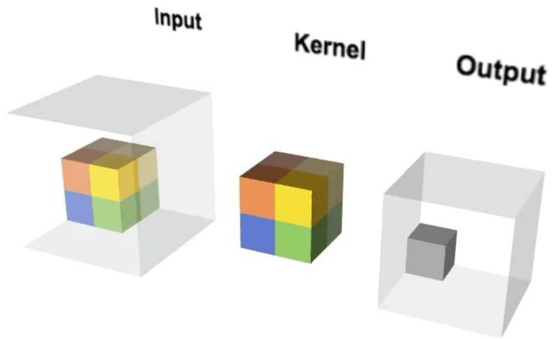
Convolutional Networks

- **Structural prior:** spatial neighbourhood defines the role of a pixel
- Apply **same function** at all position
- Induces translation invariance as features are computed independent of position

Can an MLP reproduce a ConvNet?

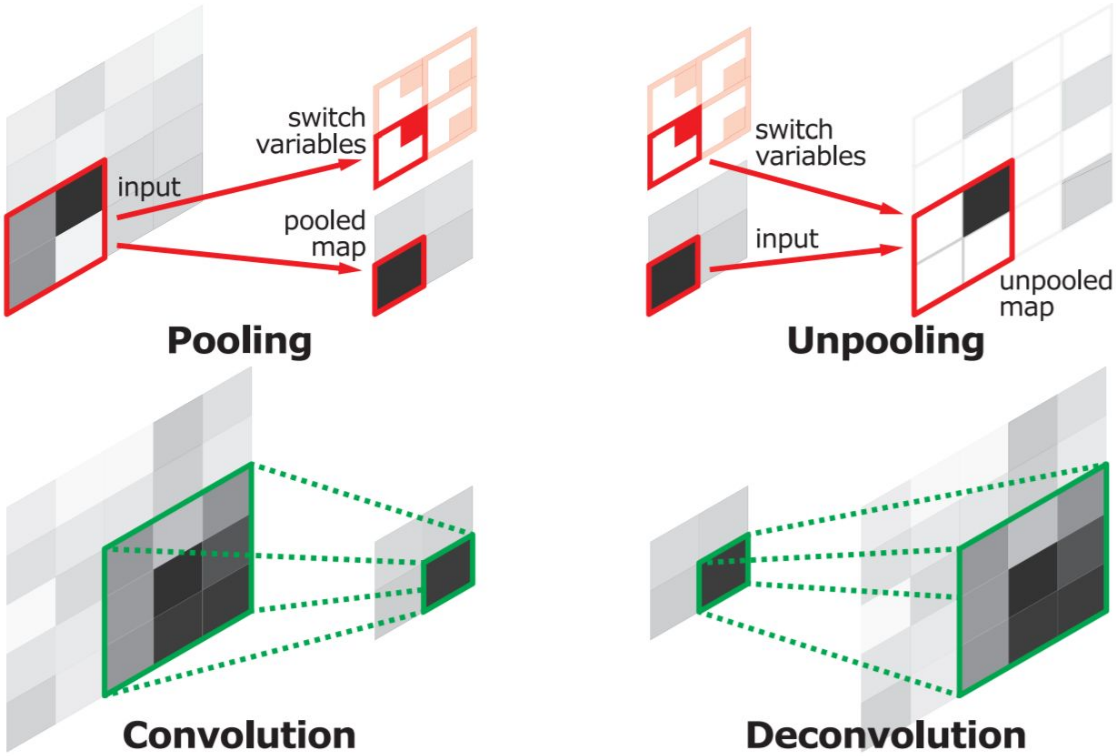


https://www.analyticsindiamag.com/convolutional-neural-network-image-classification-overview/



https://medium.com/apache-mxnet/1d-3d-convolutions-explained-with-ms-excel-5f88c0f35941

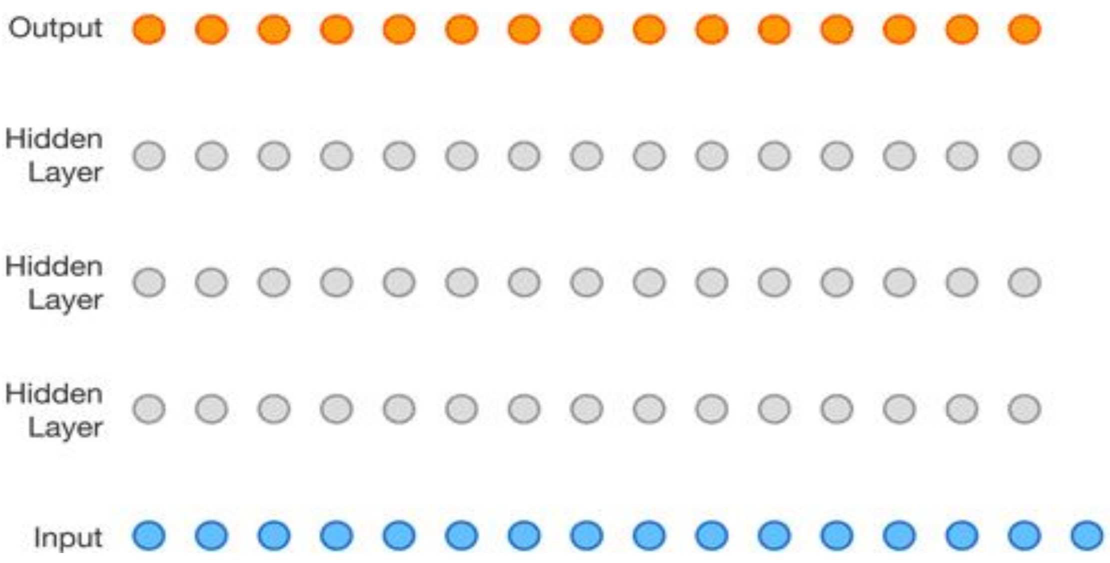
Convolutional Networks



Noh et al.

Convolutional Networks

Dilated Convolutions: (Wavenet)



https://arxiv.org/abs/1711.10433

Conv Nets: BatchNorm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

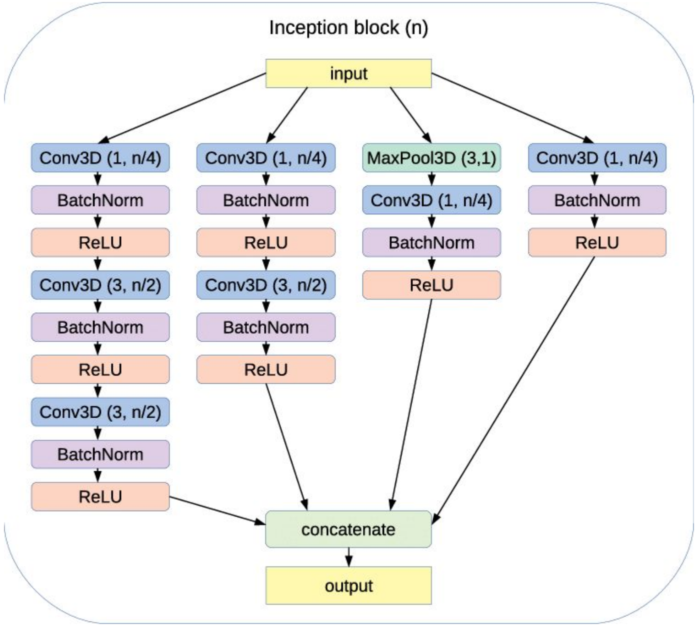
$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\boxed{\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}} \qquad \text{// normalize}$$

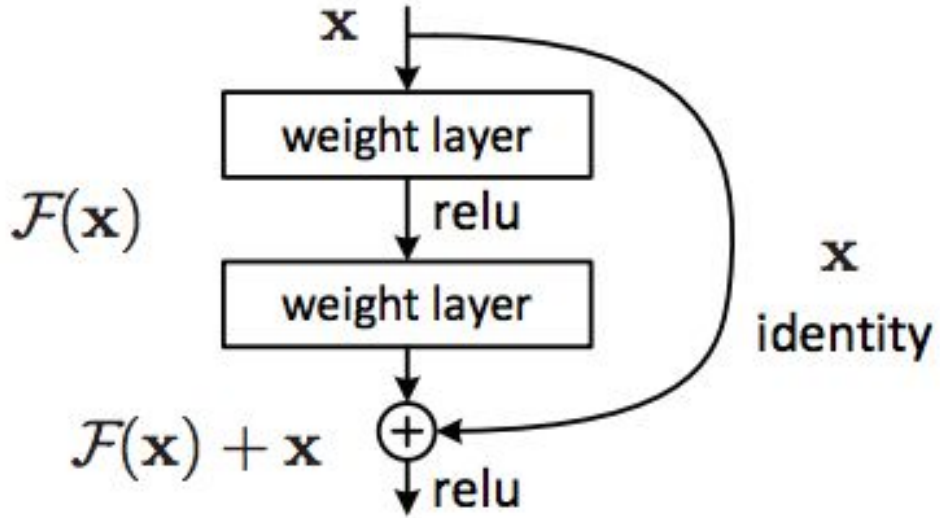
$$y_i \leftarrow \underline{\gamma \widehat{x}_i + \beta} \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

https://towardsdatascience.com/understanding-batch-normalization-with-examples-in-numpy-and-tensorflow-with-interactive-code-7f59bb126642

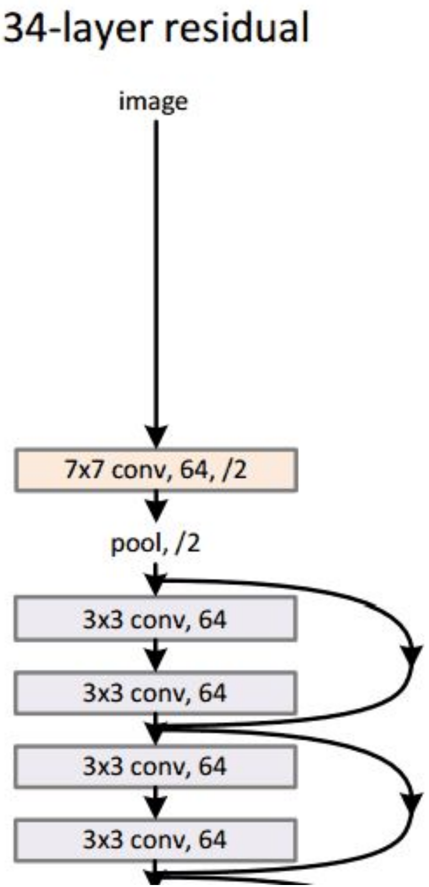


Khvostikov et al

Convolutional Networks



https://stats.stackexchange.com/questions/268820/gradient-backpropagation-through-resnet-skip-connections

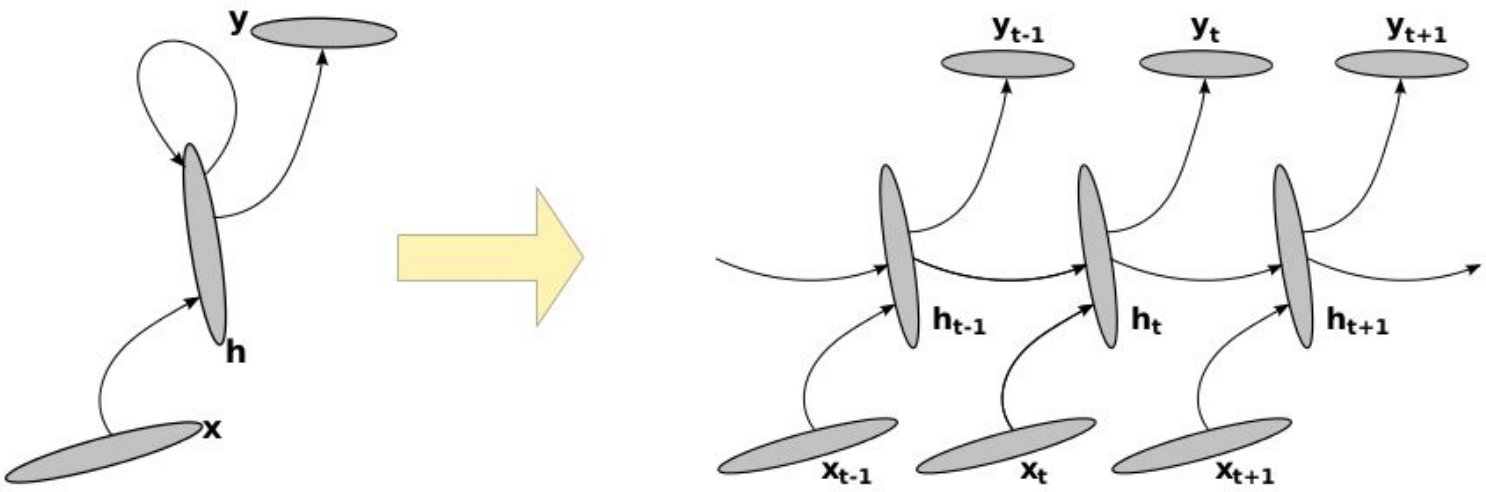


https://towardsdatascience.com/implementing-a-resnet-model-from-scratch-971be7193718

Structure in models

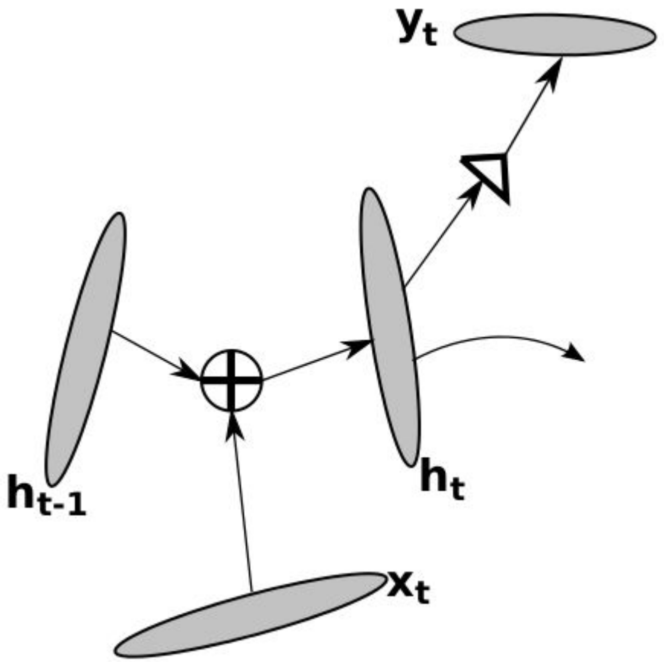
Recurrent Neural Networks

Recurrent Neural Networks



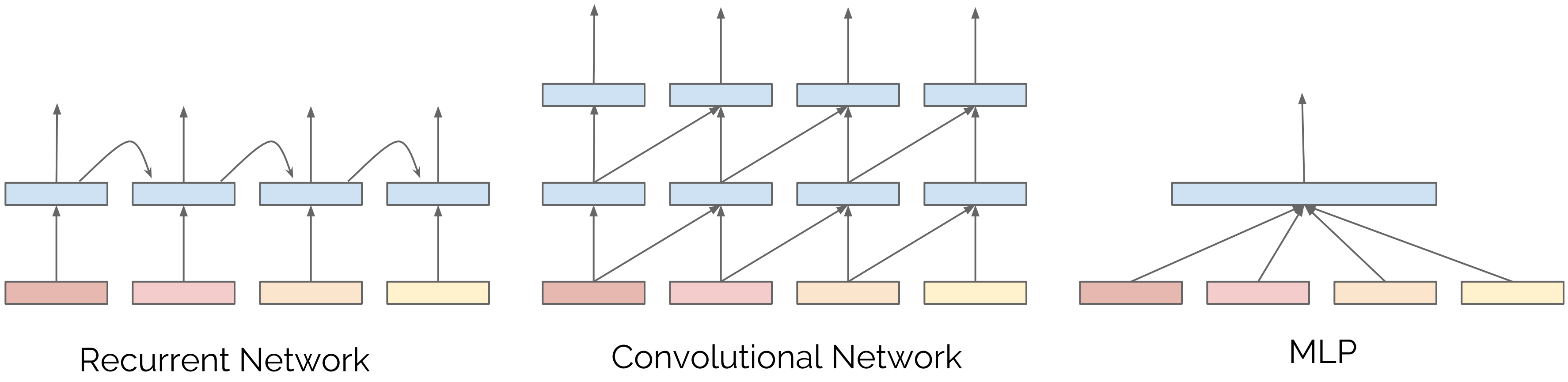
Pascanu et al. 2014

Recurrent Neural Networks

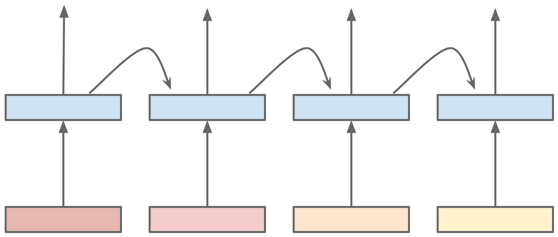


Pascanu et al. 2014

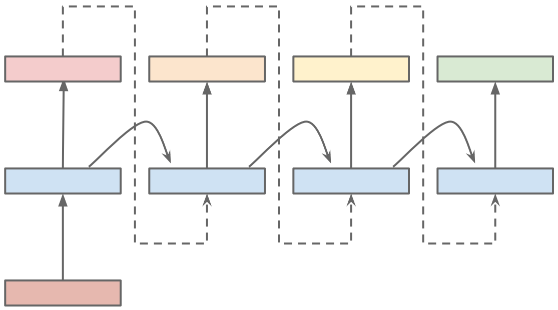
Recurrent Neural Networks



Recurrent Neural Networks

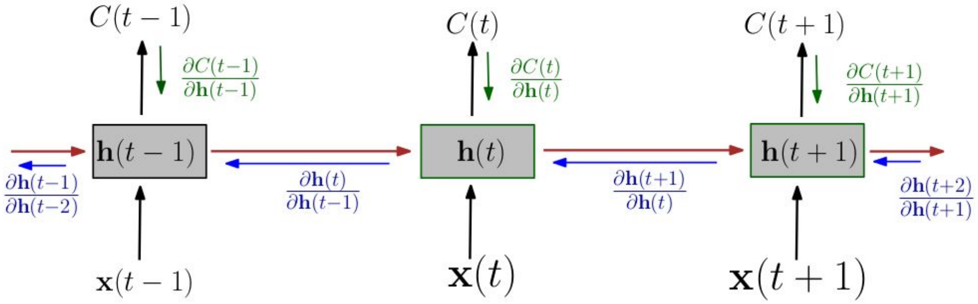


Teacher forcing



Unconstrained

Recurrent Neural Networks

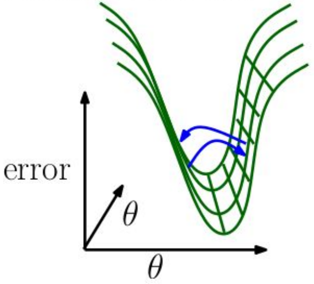


$$\frac{\partial C}{\partial \mathbf{W}} = \sum_t \frac{\partial C(t)}{\partial \mathbf{W}} = \sum_t \sum_{k=0}^{t} \frac{\partial C(t)}{\partial \mathbf{h}(t)} \frac{\partial \mathbf{h}(t)}{\partial \mathbf{h}(t-k)} \frac{\partial \mathbf{h}(t-k)}{\partial \mathbf{W}}$$

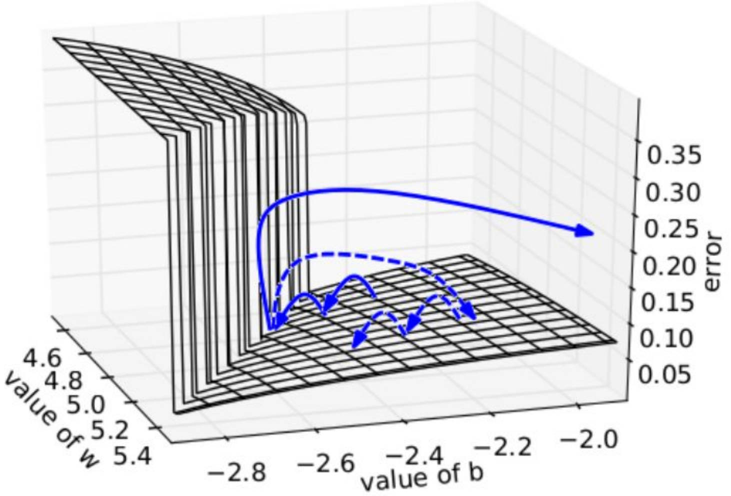
$$\frac{\partial \mathbf{h}(t)}{\partial \mathbf{h}(t-k)} = \prod_{j=k+1}^{t} \frac{\partial \mathbf{h}(j)}{\partial \mathbf{h}(j-1)}$$

Recurrent Neural Networks

Classical view for:

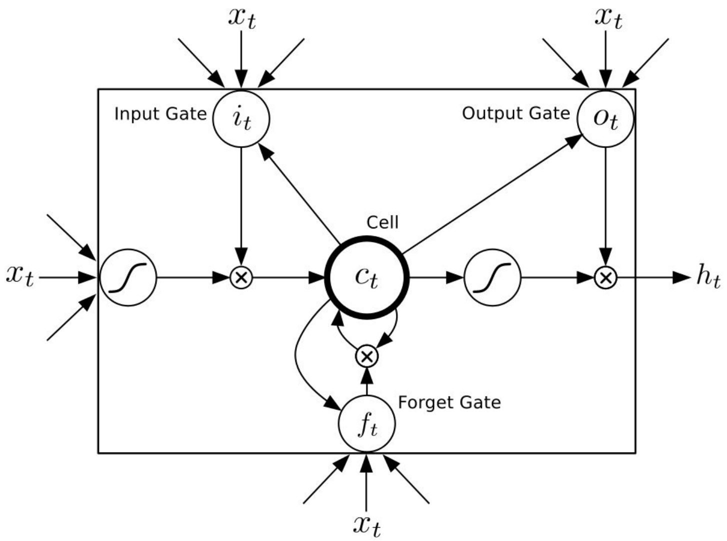


The error is $(h(50) - 0.7)^2$ for $h(t) = w\sigma(h(t-1)) + b$
with $h(0) = 0.5$



Recurrent Neural Networks

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \quad (7)$$
$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \quad (8)$$
$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \quad (9)$$
$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \quad (10)$$
$$h_t = o_t \tanh(c_t) \quad (11)$$

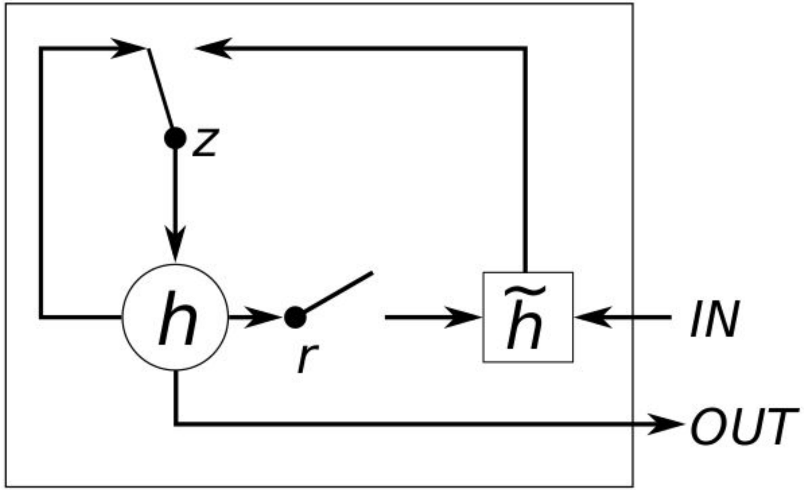


[Hochreiter et al. 1997](#)
[Graves 2013](#)

Recurrent Neural Networks

Chung et al. 2015

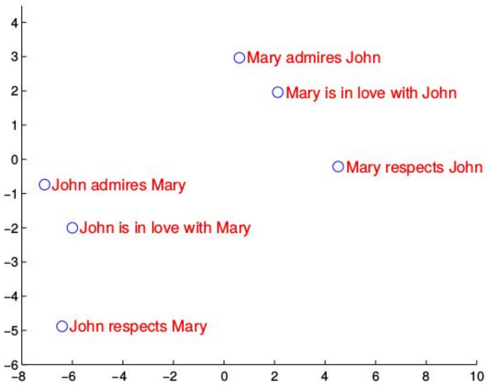
$$z = \sigma(W_z x_t + U_z h_{t-1})$$
$$r = \sigma(W_r x_t + U_r h_{t-1})$$
$$\tilde{h} = tanh(W_h x_t + U_h(r \circ h_{t-1}))$$
$$h_t = (1 - z) \circ h_{t-1} + z \circ \tilde{h}$$

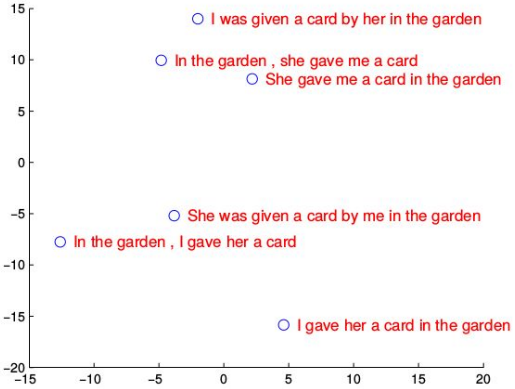


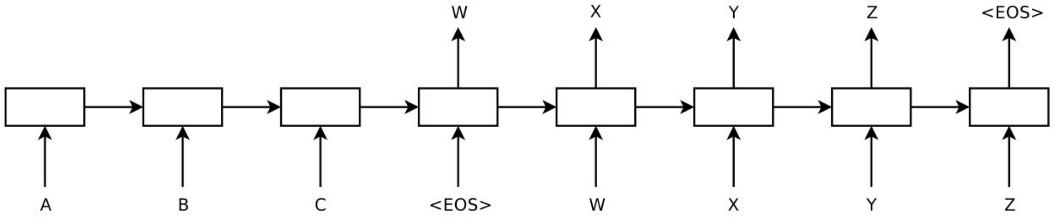
(b) Gated Recurrent Unit

Sequence to Sequence

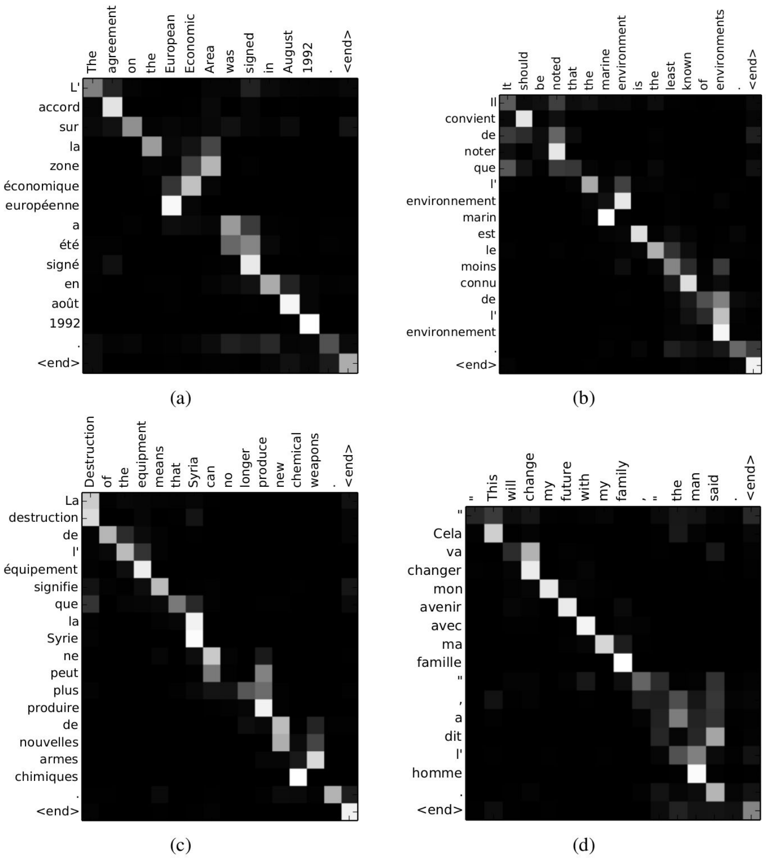
[Sutskever et al. 2014](#)



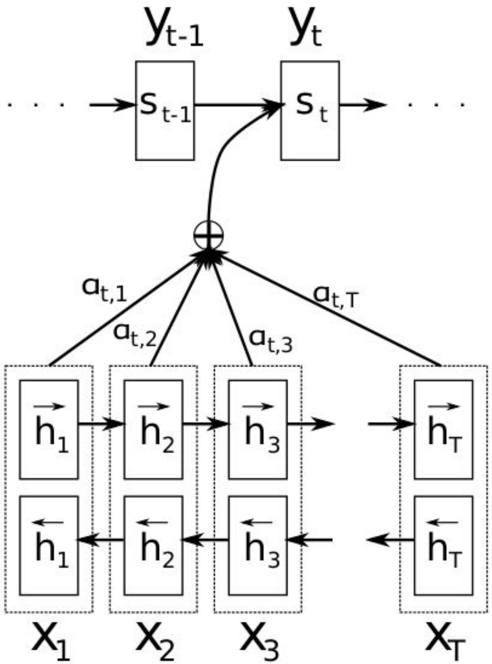




Sequence to Sequence

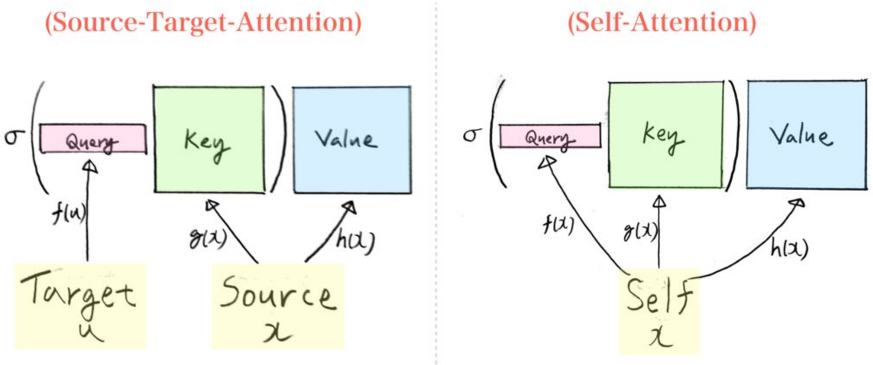


[Bahdanau et al. 2015]



Transformer

https://mchromiak.github.io/articles/2017/Sep/12/Transformer-Attention-is-all-you-need/#.XG9ar-GTJkY



http://deeplearning.hatenablog.com/entry/transformer

FINAL THOUGHTS & DISCUSSION

Conclusions

- **Learning** is about discovering the solution from data
- **Deep Learning** is about a particular family of function approximators
- **ConvNets / RNNs / Transformer** is about particular structure on the architecture (inductive bias)
- A lot of open questions, a lot of interesting questions, fast growing field

Thank you!

Questions?