Language Technologies Institute

Carnegie Mellon University

Advanced Multimodal Machine Learning

Lecture 9.1: Multimodal Optimization

Louis-Philippe Morency

Lecturer: Amir Zadeh

Lecture Objectives

- Practical Deep Model Optimization
  - Background
  - Optimization and Iterative approaches
  - Learning rate and Momentum
  - Regularization
  - Co-adaptation
  - Multimodal Optimization
- Variational Methods
  - Variational AE

Lecture Objectives

- Practical Deep Model Optimization
  - Background
  - Optimization and Iterative approaches
  - Learning rate and Momentum
  - Regularization
  - Co-adaptation
  - Multimodal Optimization
- Variational Methods
  - Variational AE

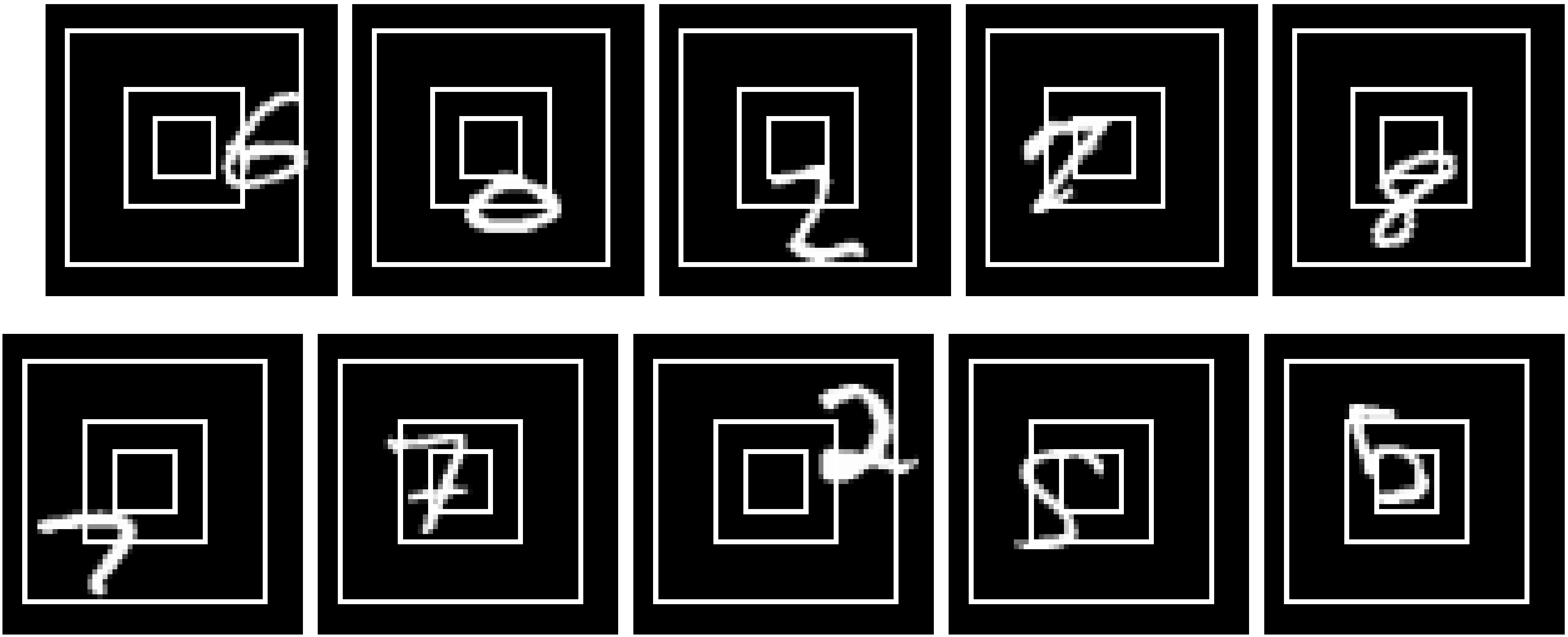
Hard attention

Hard attention

- Soft attention requires computing a representation for the whole image or sentence
- Hard attention on the other hand forces looking only at one part
- Main motivation was reduced computational cost rather than improved accuracy (although that happens a bit as well)
- **Saccade followed by a glimpse – how human visual system works**

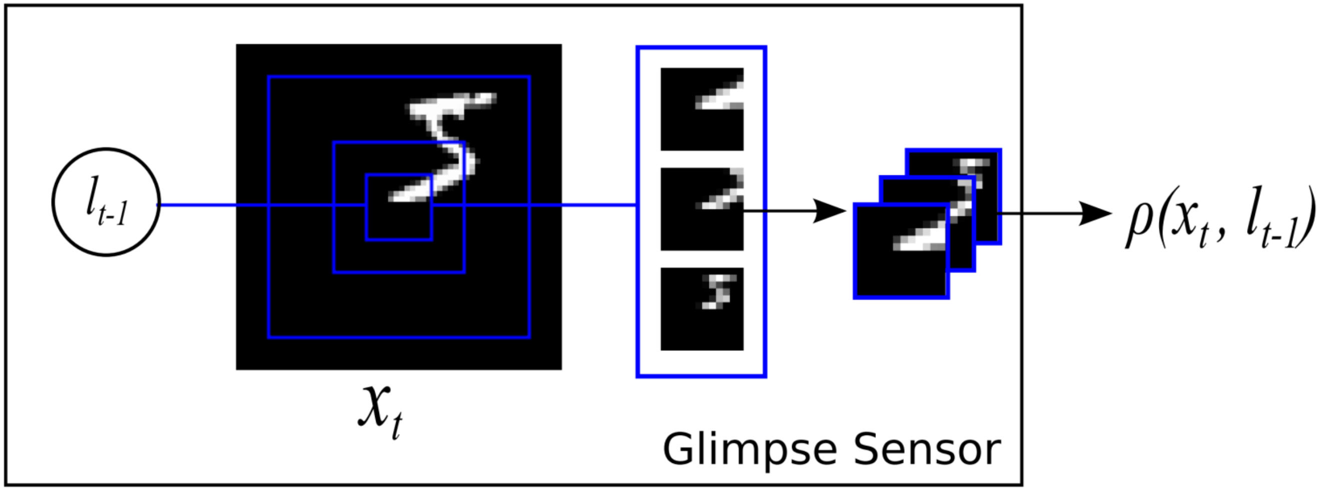
[Recurrent Models of Visual Attention, Mnih, 2014]
[Multiple Object Recognition with Visual Attention, Ba, 2015]

Hard attention examples



Glimpse

- Looking at a part of an image at different scales

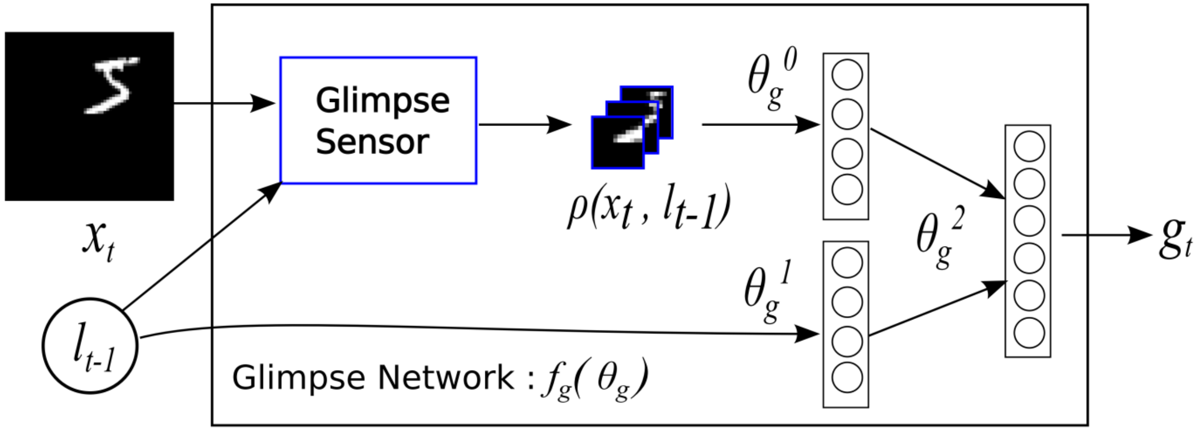


- At a number of different scales combined to a single multichannel image (human retina like representation)
- Given a location l_t output an image summary at that location

[Recurrent Models of Visual Attention, Mnih, 2014]

Glimpse network

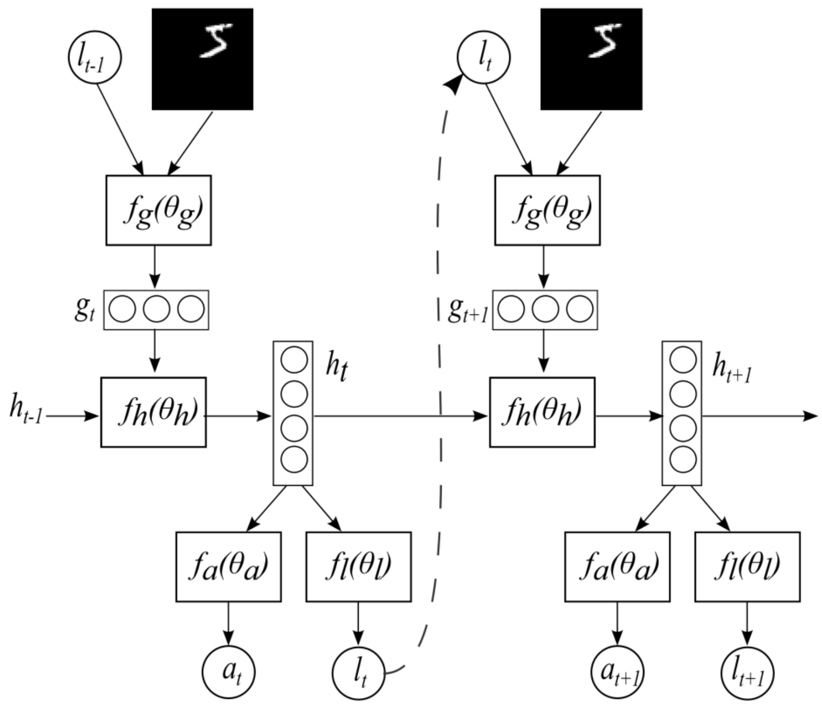
- Combining the Glimpse and the location of the glimpse into a joint network



- The glimpse is followed by a feedforward network (CNN or a DNN)
- The exact formulation of how the location and appearance are combined varies, the important thing is combining **what** and **where**
- Differentiable with respect to glimpse parameters but not the location

Emission network

- Given an image a glimpse location l_t, and optionally an action a_t
- Action can be:
  - Some action in a dynamic system – press a button etc.
  - Classification of an object
  - Word output
- This is an RNN with two output gates and a slightly more complex input gate!



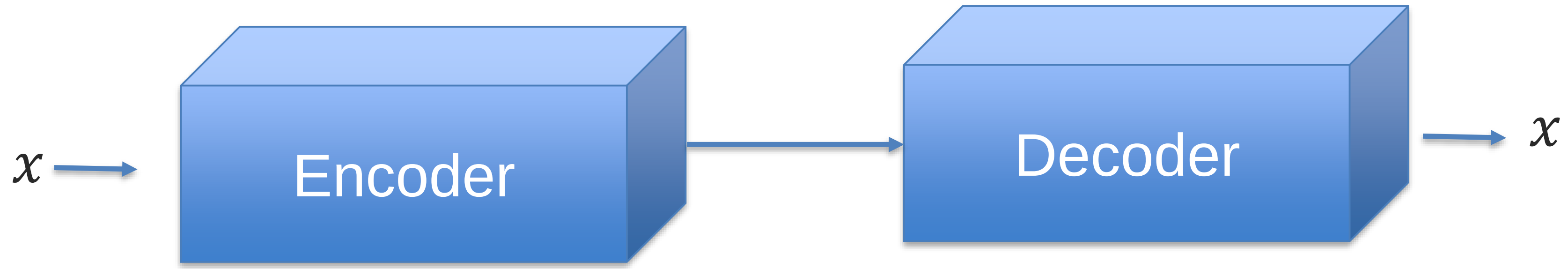
Other Example of Hard Attention

- http://proceedings.mlr.press/v37/xuc15.pdf

Variational Autoencoders

Auto-encoder

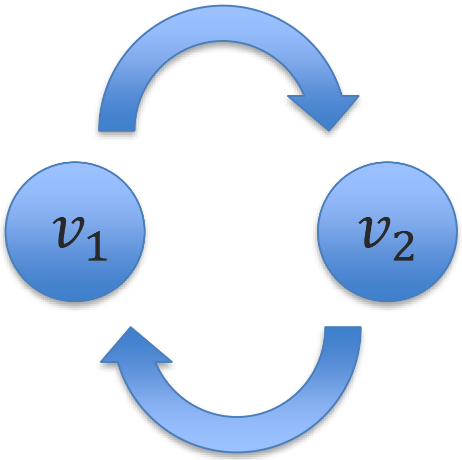
- A combination of an Encoder and a Decoder encoding x and decoding x



- The loss reconstruction error of x.

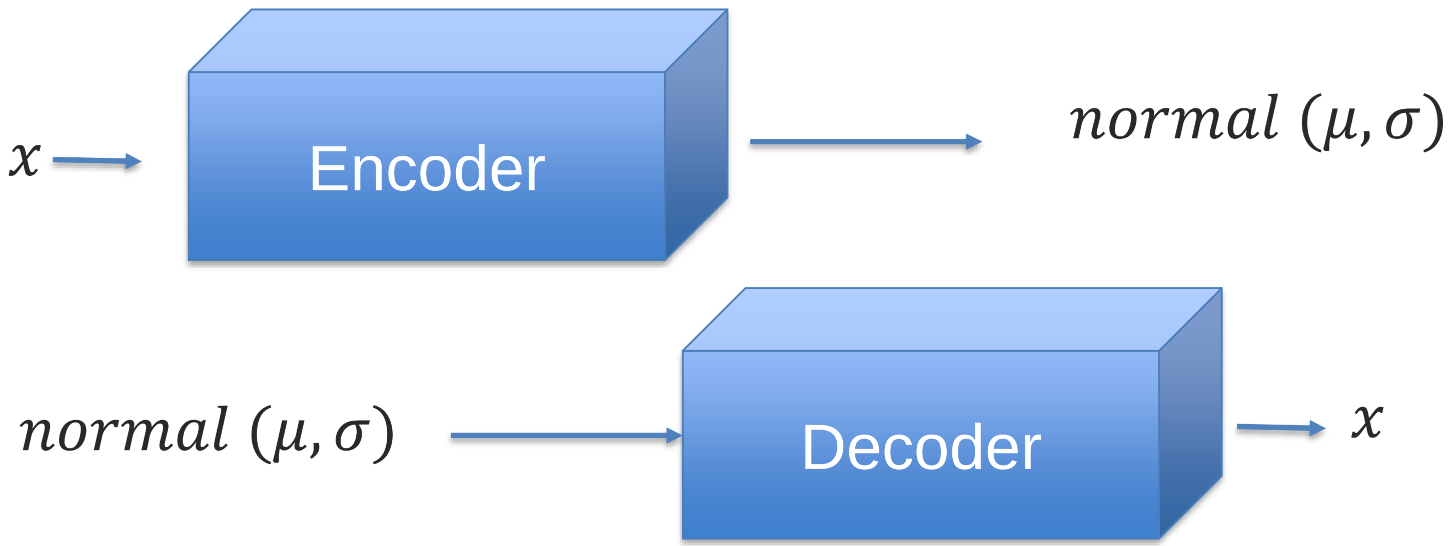
Variational Inference

- When inference is not possible
  - Either relax the problem
  - Or use variational methods
- Variational inference:
  - Unroll through time (MCMC, Gibbs) – RBM
  - Mean-field Approximation (Fully Connected CRF)
- Both cases we have an approximation of the variables.



Variational Auto-encoder

- We assume exact inference is not possible but approximation is possible.

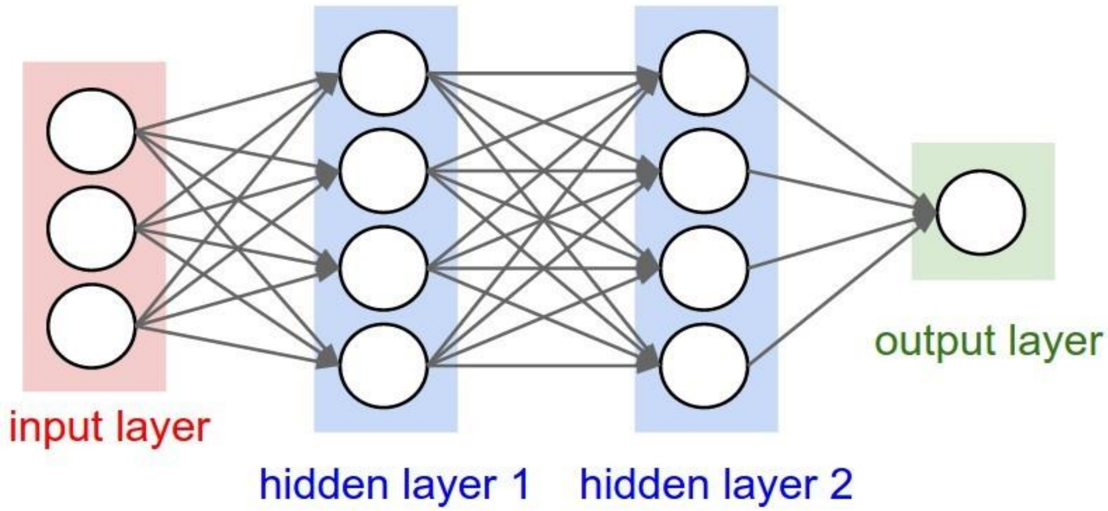


Variational Auto-encoder

- A probability controls the encoder space
  - More meaningful representations
- Space is split in euclidean-meaningful representations.
- The normal distributions have nice properties.

Background-MLP

- Multilayer Perceptron
  - Superset of CNNs, LSTMs, GRUs.
  - Reminder: A recursive application of affine transformations and nonlinearities
  - Can solve everything but often a headache to optimize



Stochastic Gradient Descent

- Loss functions:
  - Mean Squared Error: $L = ||f(x) - f^*(x)||$
  - Categorical Cross-Entropy (surrogate)

$$L_i = \boxed{-\log\left(\boxed{\frac{e^{f_{y_i}}}{\sum_j e^{f_{y_j}}}}\right)}$$

Softmax function

Minimizing the negative log likelihood.

Stochastic Gradient Descent

- Stochastic since updates are done based on a random subset of training data:

For i=1,2,3, …, N:

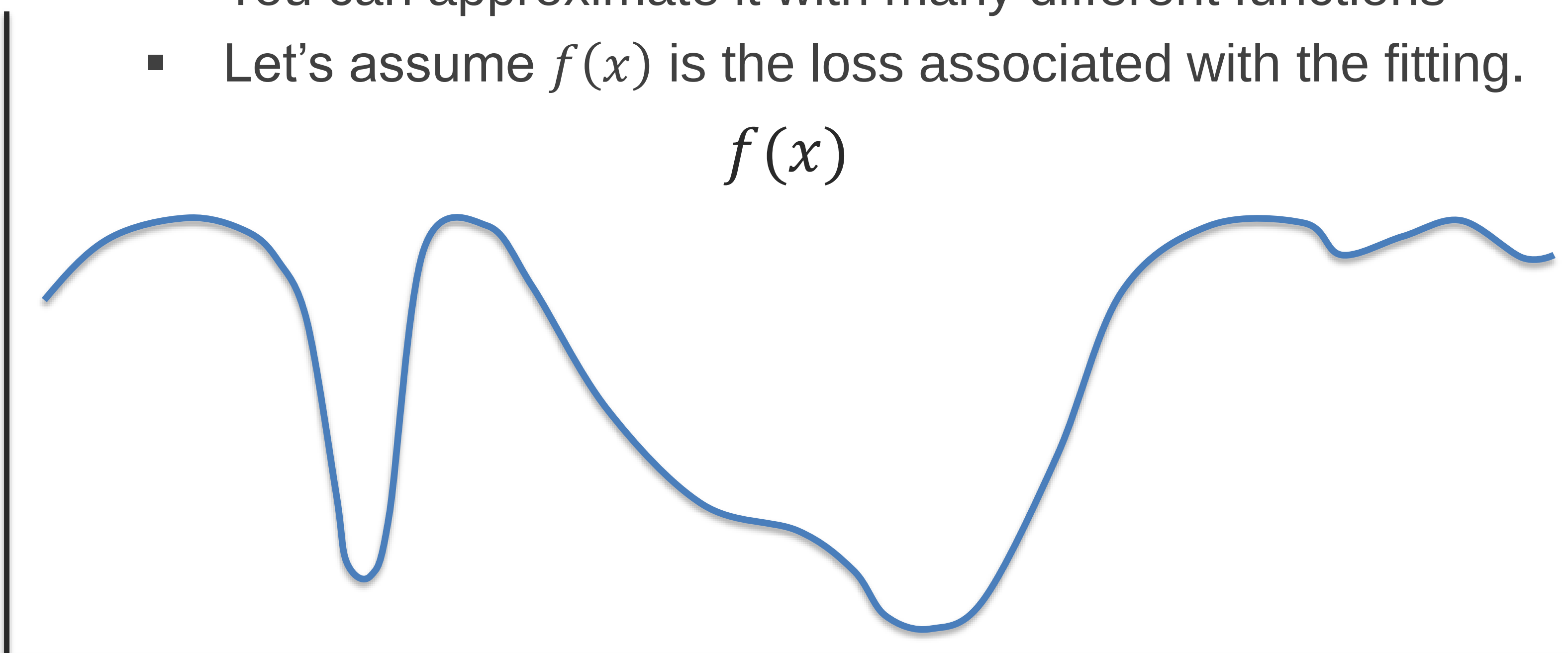
$$w_{t+1} = w_t - \propto_t \nabla f(w_t)$$

Lecture Objectives

- Practical Deep Model Optimization
  - Background
  - Optimization and Iterative approaches
  - Learning rate and Momentum
  - Regularization
  - Co-adaptation
  - Multimodal Optimization
- Variational Methods
  - Variational AE

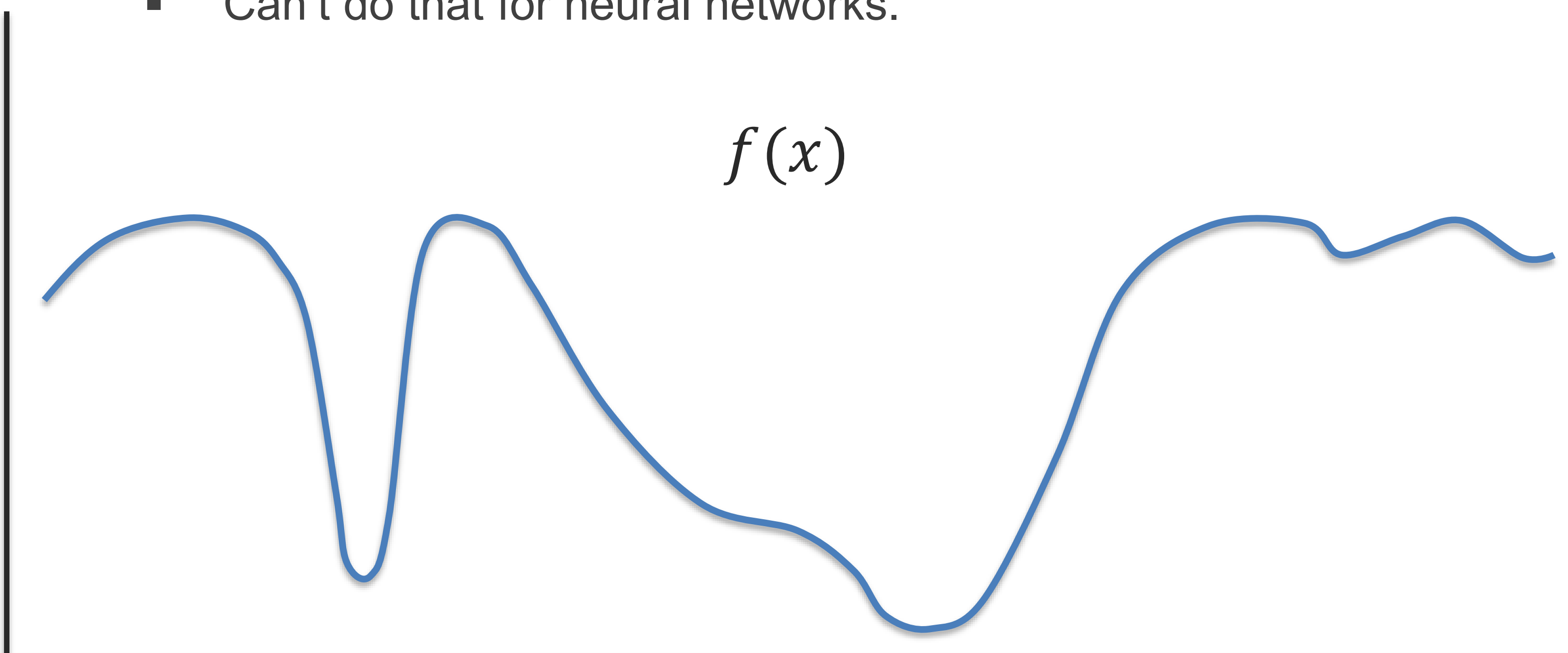
Optimization

- $g(x)$ is an unknown distribution in the nature
  - Probability of shark given fin length
  - You can sample–go get sharks and size the fins. Good luck!
  - You can approximate it with many different functions
  - Let's assume $f(x)$ is the loss associated with the fitting.



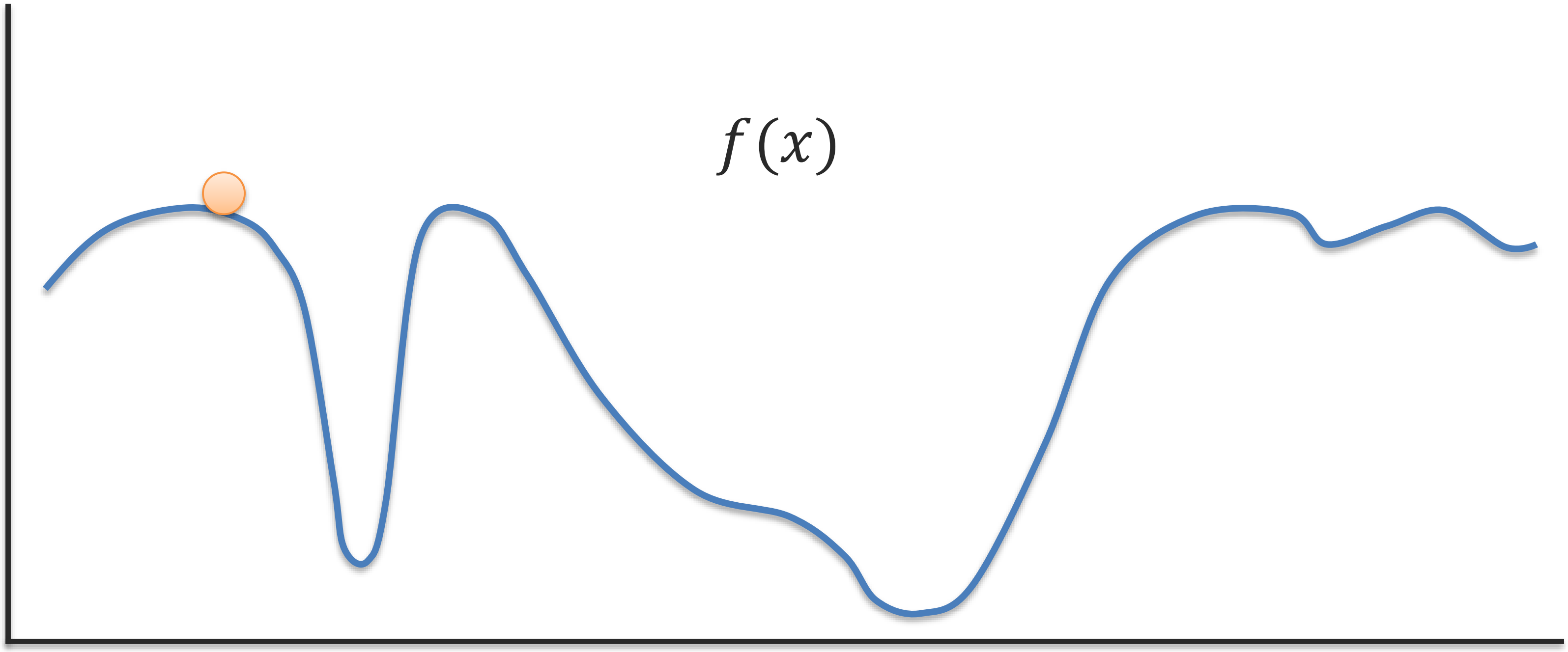
Optimization

- Assume that approximation function is $\bar{g}(x;\, \theta)$
- Goal is to minimize $f(x)$: $\bar{g}(x;\, \theta) \backsimeq g(x)$
- One easy way: $\frac{\partial f}{d\theta} = 0$ solve for θ.
  - Can't do that for neural networks.

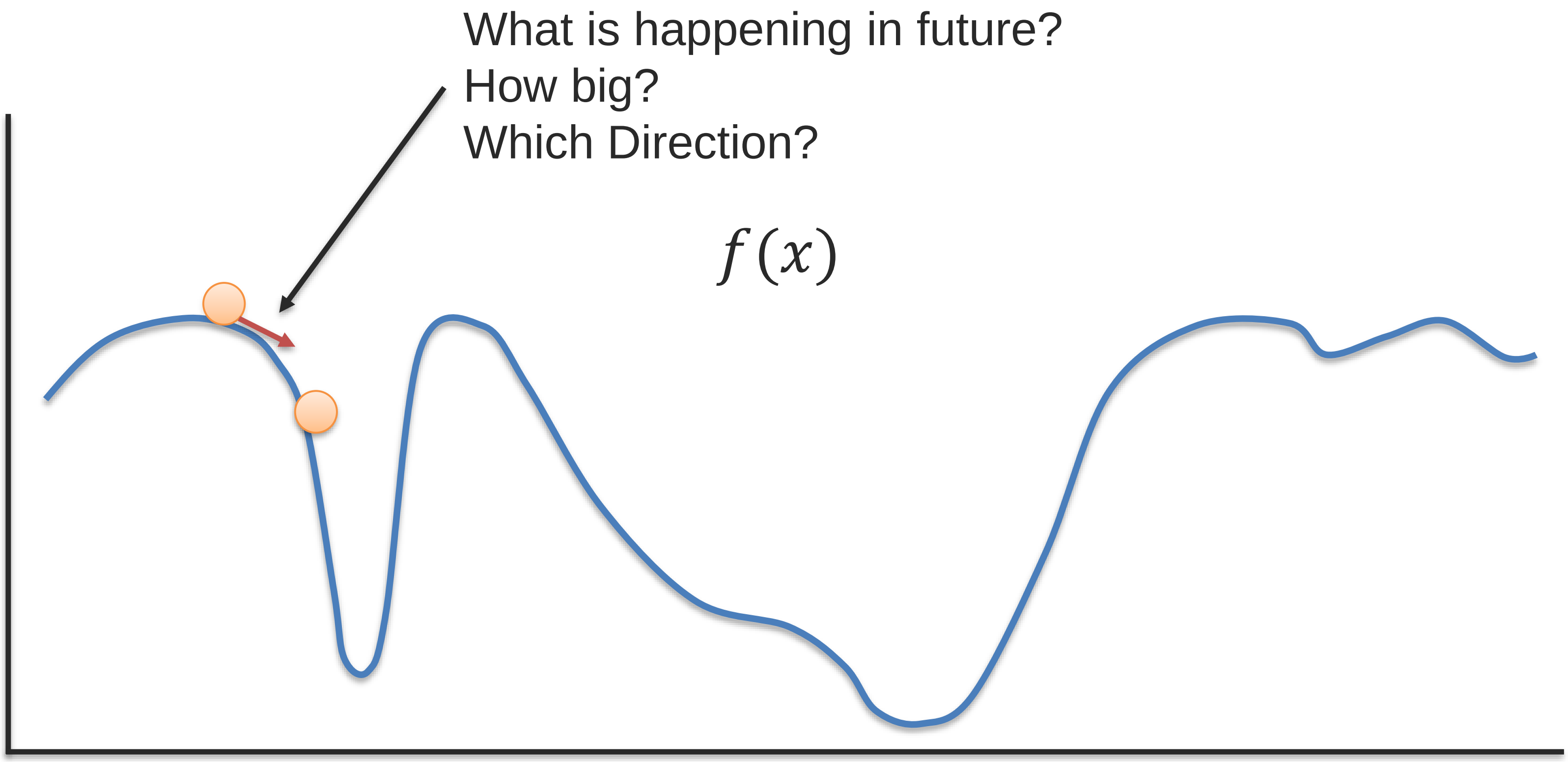


Iterative Approaches

- Start from a point on $f(x)$ namely x_1.
- Test which direction gives you a better value on $f(x)$.
- Go to the new position.



Iterative Approaches



Let's mathematically formulate this

Iterative Approaches

- Any differentiable loss function $f(x)$
- Taylor's expansion: $f(x) = \sum_0^\infty \frac{f^{(n)}(x_1)}{n!}(x - x_1)^n$
- First order: $f(\mathbf{x}) = f(\mathbf{x}_1) + \nabla f(\mathbf{x}_1) \cdot (\mathbf{x} - \mathbf{x}_1) + O(||\mathbf{x} - \mathbf{x}_1||^2)$

$$f(\mathbf{x}_1 + h\mathbf{u}) - f(\mathbf{x}_1) = h\nabla f(\mathbf{x}_1) \cdot \mathbf{u} + h^2 O(1)$$

$$\mathbf{u} = -\nabla f(\mathbf{x}_1)/||\nabla f(\mathbf{x}_1)|| \longleftarrow$$

Best unit solution – go in the direction of gradient

- Big step size h is unrealistic in most cases
- So, GD is: For $t = 1, 2, \ldots, N_{max}$:

Step size is a simple function! Bad! $\longrightarrow$

$$\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t - \alpha_t \nabla f(\mathbf{x}_t)$$

Iterative Approaches

- Taylor's expansion: $f(x) = \sum_0^\infty \frac{f^{(n)}(x_1)}{n!}(x - x_1)^n$
- Higher order: Second order Taylor expansion (Newton methods)
- Learning rate can be approximated
  - Bigger steps can be taken as order goes higher.
  - Step size now a function of higher order derivatives
- However, we don't have enough resources
- Maybe Quasi-Newton approaches?
- So let's stick to first order …

Gradient Descent

- Most practical optimization method so far for deep learning:
  - Linear in space and time complexity per number of parameters
  - Applicable to SIMD Parallelization Paradigm
  - Nice attributes for deep learning
- This is all good but how should the updates happen on the model?

SGD vs GD

GD

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$Update$$

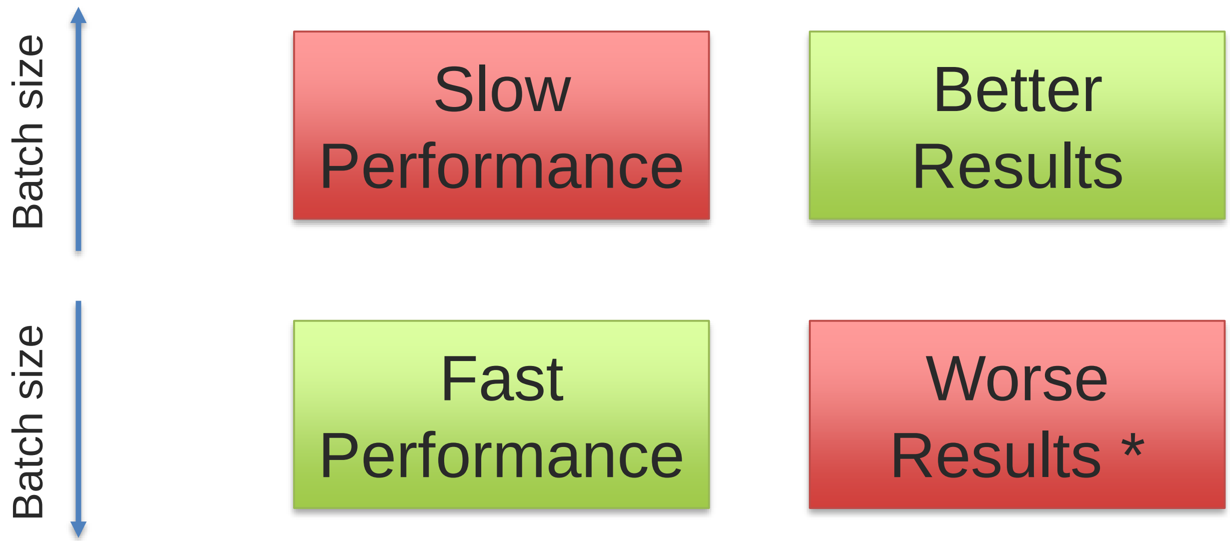
SGD

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$3 * Update$$

SGD vs GD - Tradeoff

- GD takes a global decision for updating the gradient
- SGD takes a local decision for updating the gradient



* Or even no results

SGD vs GD - Tradeoff

- Let's go speed! Going one sample at a time! How bad can it be?

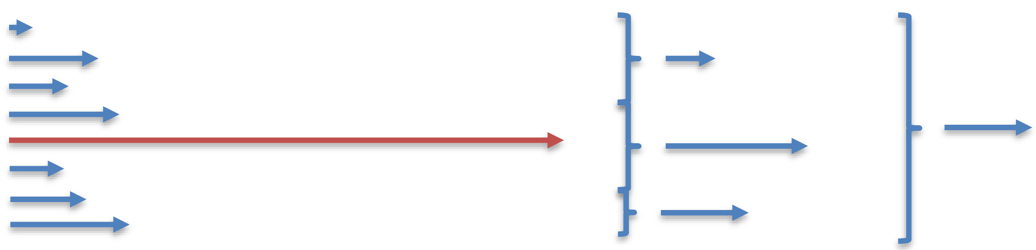
$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

- But it's just one sample point!
- It really only takes one sample point to send Sigmoids to no man's land!

SGD vs GD - Tradeoff

- Ok let's not do that, how bad is my approximation of global gradient if I use a batch instead?

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$



1 sample

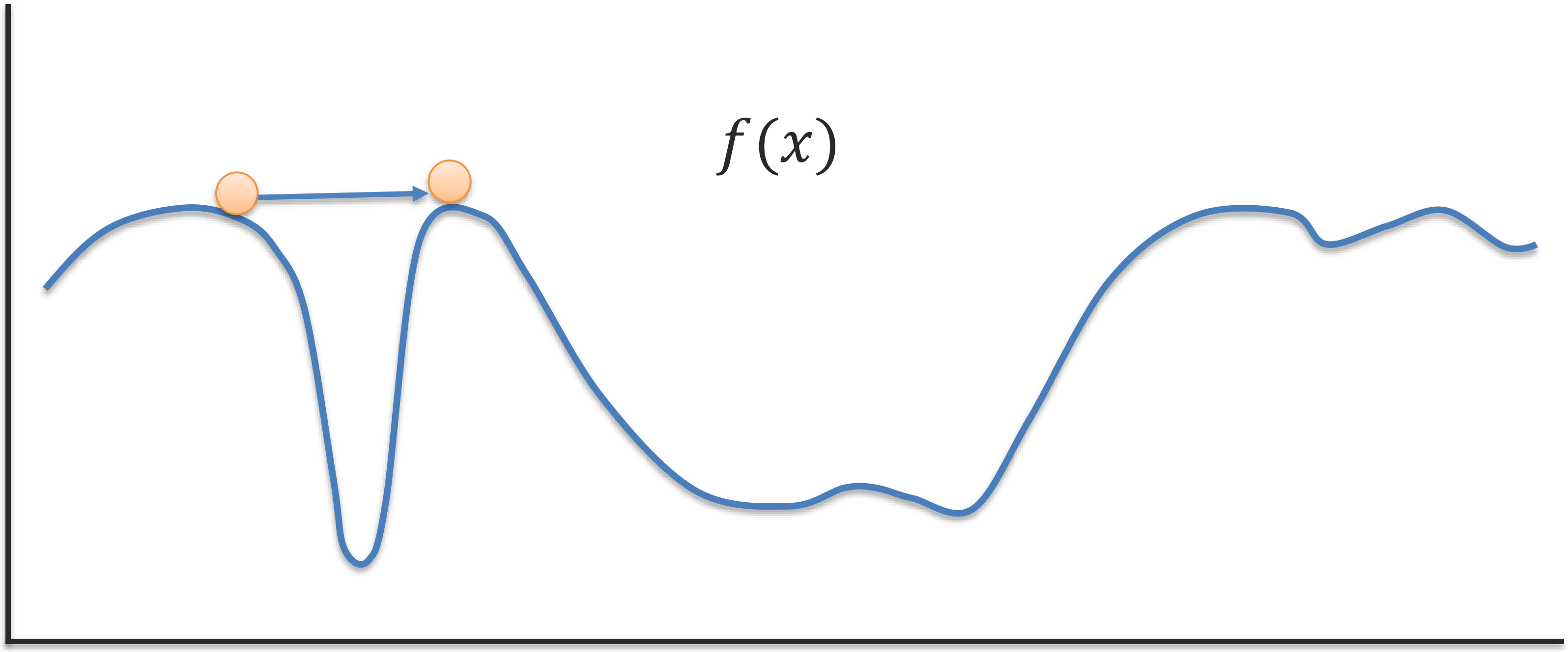
batch

Entire dataset

- Law of averages helps a lot.
  - Gradient of 32 points is already extremely close to that of all the dataset.

Learning rate decay

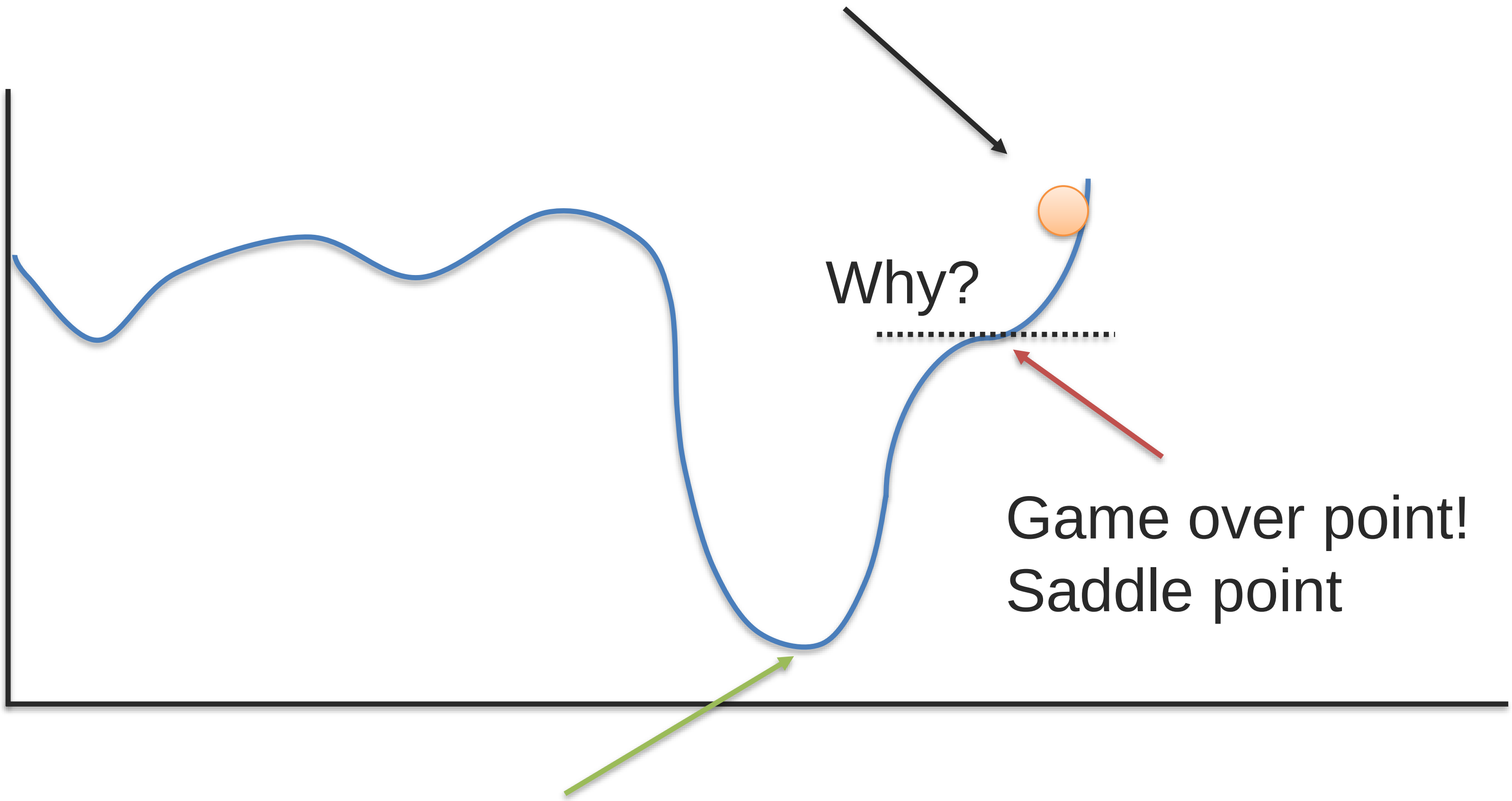
- Jumping over good solutions:
  - Decay learning rate
  - We won't let model jump around too much after some time



Example

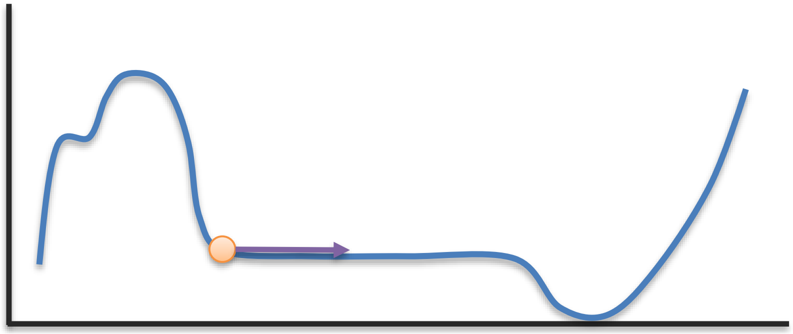
Game over zone! Plateau

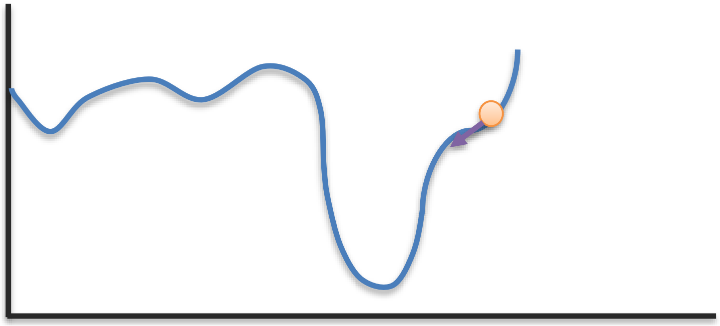
Example



Momentum

- Both situations avoidable if we had higher order derivatives. Which we don't!
- Let's jump over/speed through them.



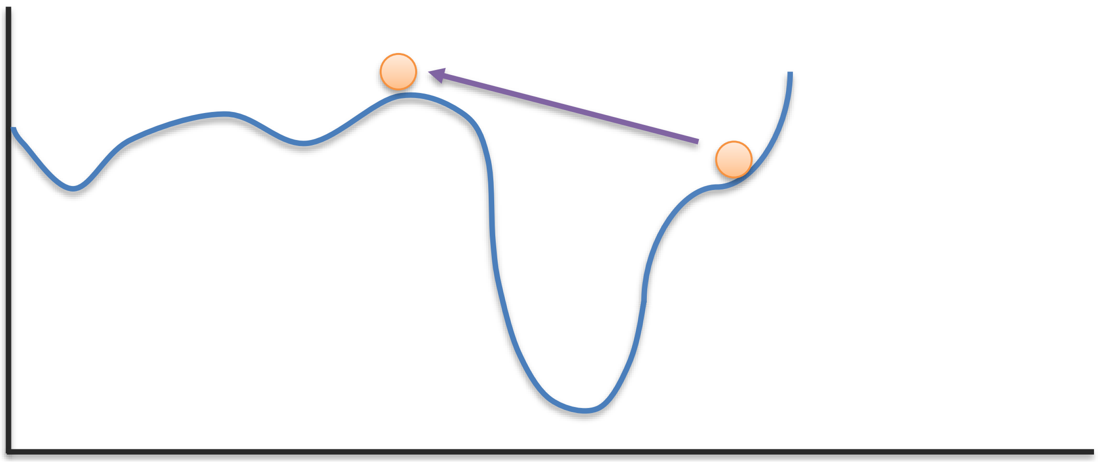


Momentum

SGD: $w_{t+1} = w_t - \propto_t \nabla f(w_t)$

SGD+M: $w_{t+1} = w_t - (\propto_t \nabla f(w_t) + m \nabla w\,(t))$

Momentum can backfire!



Adaptive Learning Rate

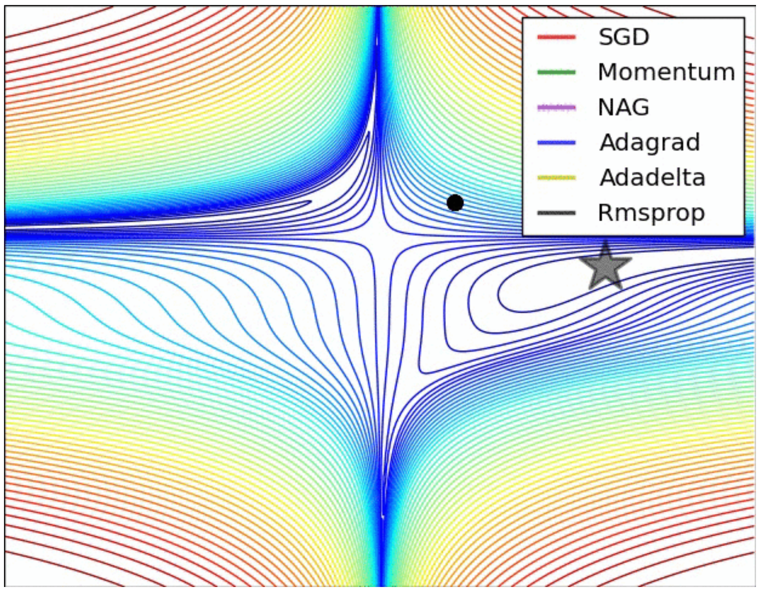
- General Idea: Let neurons who just started learning have huge learning rate.
- Adaptive Learning Rate is an active area of research:
  - Adadelta
  - RMSProp

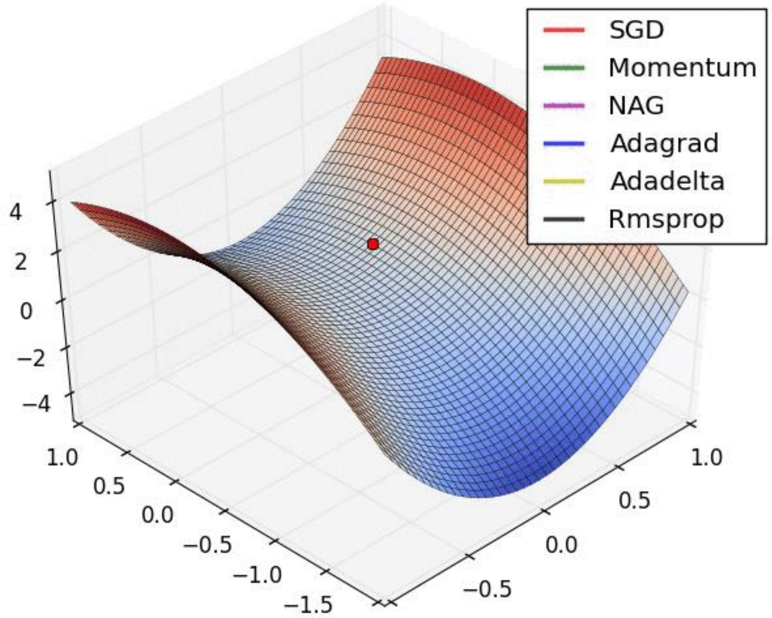
    ```
    cache = decay_rate * cache + (1 - decay_rate) * dx**2
    x += - learning_rate * dx / (np.sqrt(cache) + eps)
    ```

  - Adam

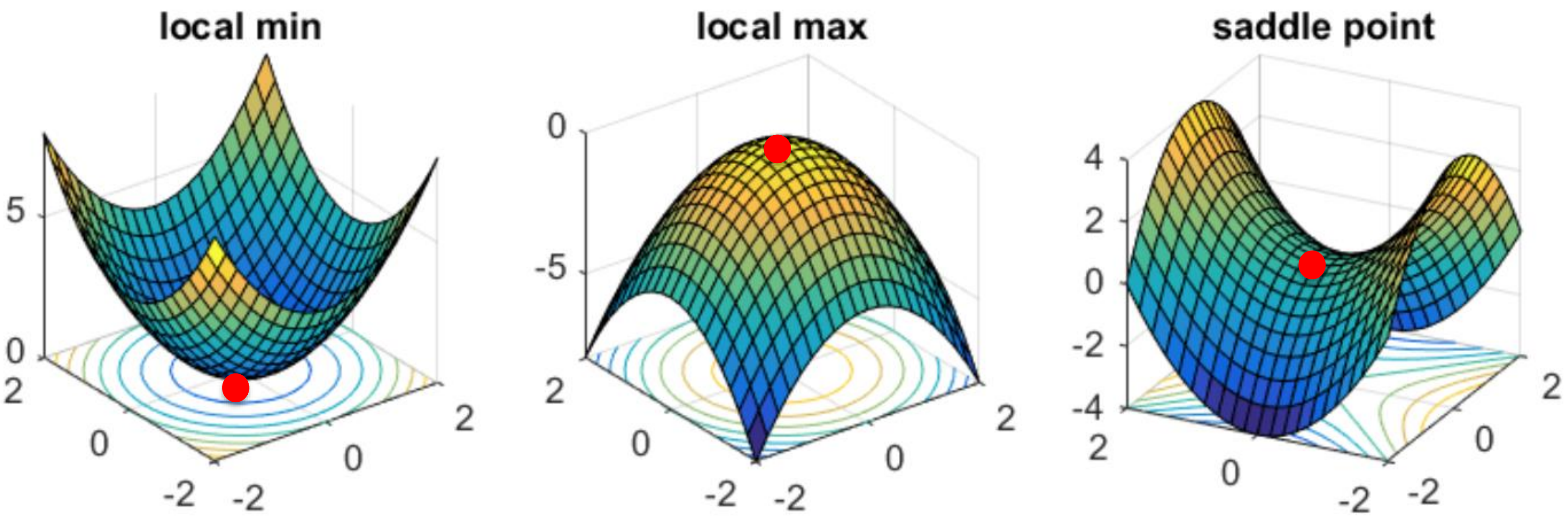
    ```
    m = beta1*m + (1-beta1)*dx
    v = beta2*v + (1-beta2)*(dx**2)
    x += - learning_rate * m / (np.sqrt(v) + eps)
    ```

Comparison


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

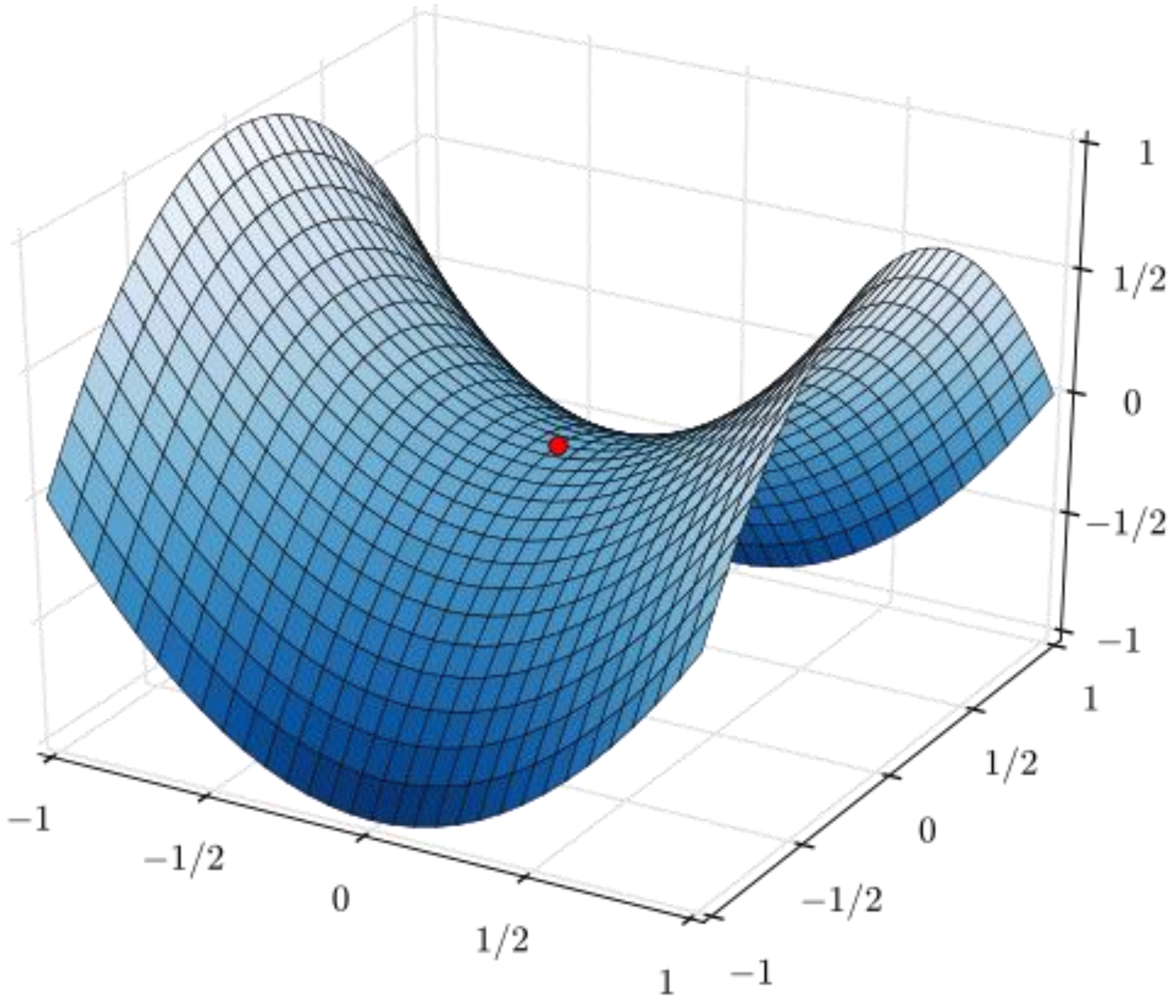
Critical Points



Saddle Points

- Deep Learning Optimization:
  - Deep Learning problems in general have many local minimas
  - Many (not all) of them are actually almost as good as global minima due to parameter permutation
  - However it is NP-hard to even find a local minima
- Lots and lots of saddles in many deep learning problems.

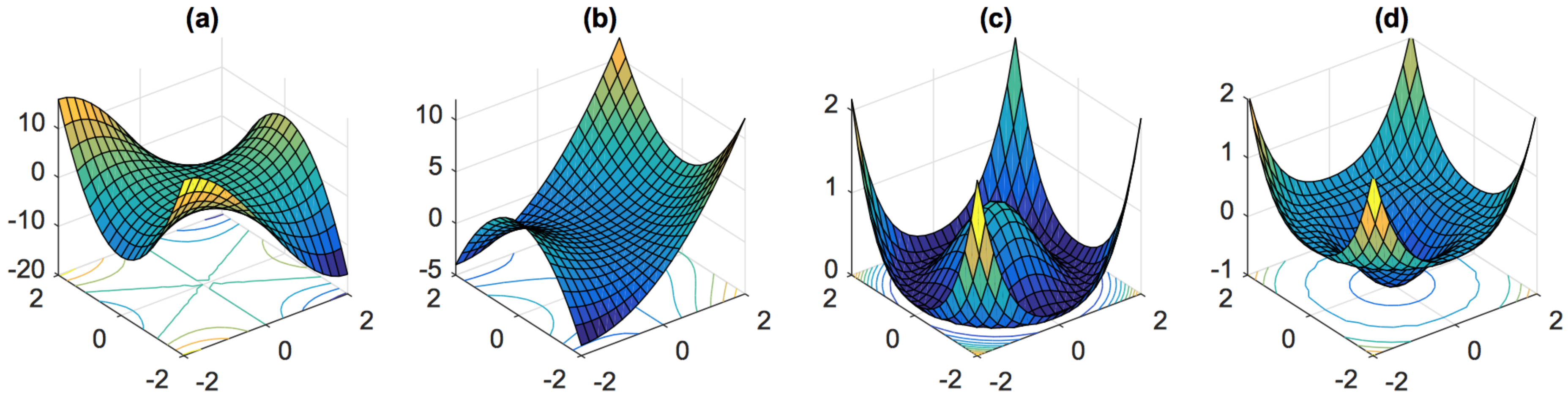
Why Saddles are Bad



Detecting Saddles

- One way to detect saddles:
  - Calculate Hessian at point x
  - If Hessian is indefinite you have a saddle for sure.
  - If Hessian is not indefinite you really can't tell.
- My loss isn't changing:
  - You are definitely close to a critical point
    - You may be in a saddle point
    - You may be in the local minima/maxima
  - One trick: quickly check the sorrounding
    - Best practical trick if Hessian is not indefinite.

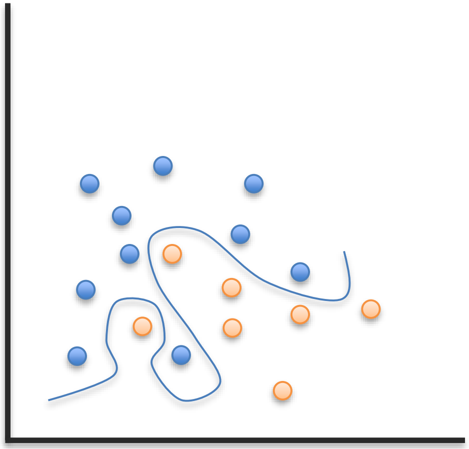
Bad Saddle Points



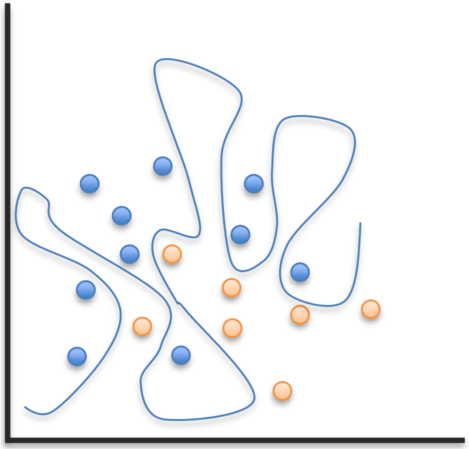
https://arxiv.org/pdf/1602.05908.pdf

Example

Real

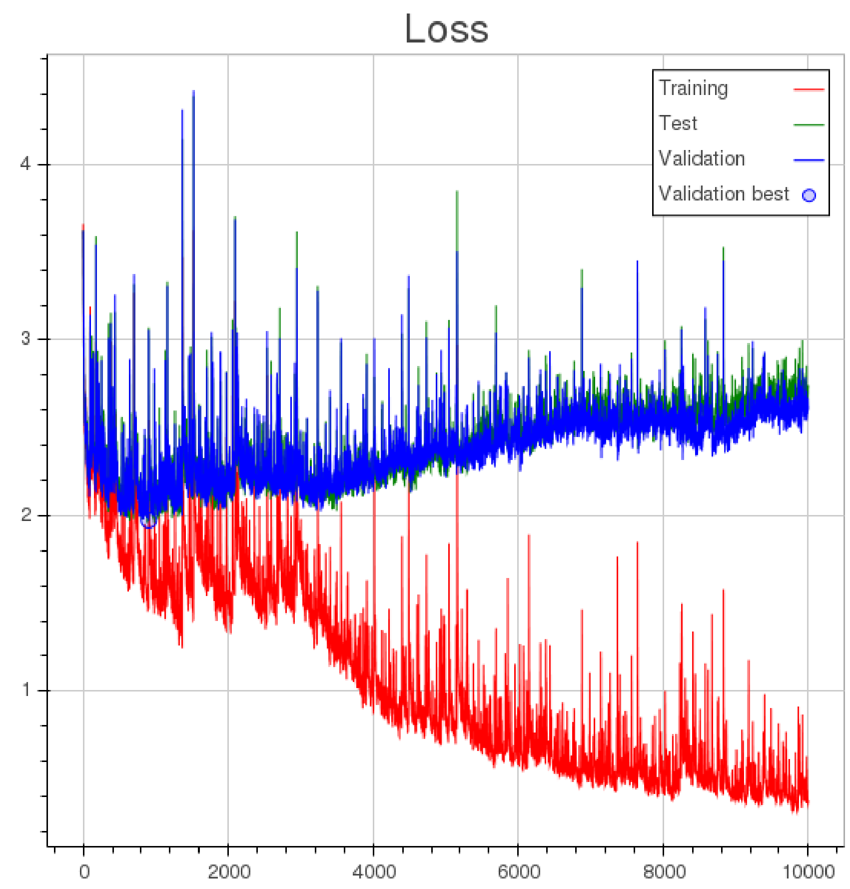


Our Model



Not the fault of learning rate or momentum

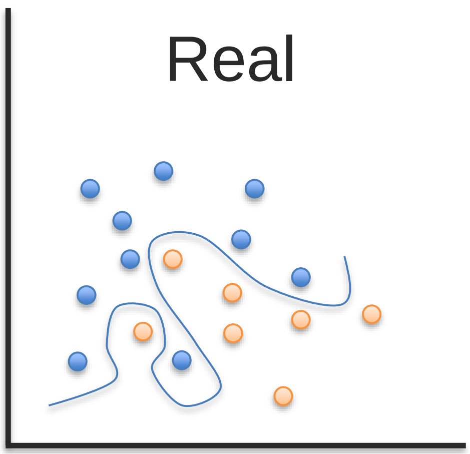
Example

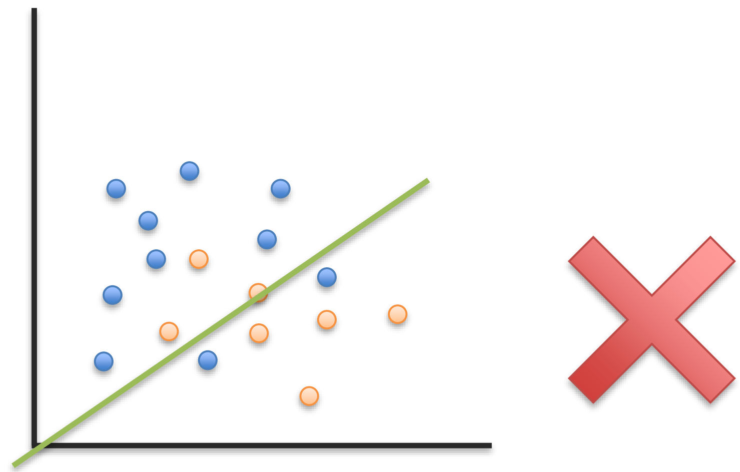

Loss
Training
Test
Validation
Validation best
4
3
2
1
0
2000
4000
6000
8000
10000

Bias-Variance

- Problem of bias and variance
  - Simple models are unlikely to find the solution to a hard problem, thus probability of finding the right model is low.

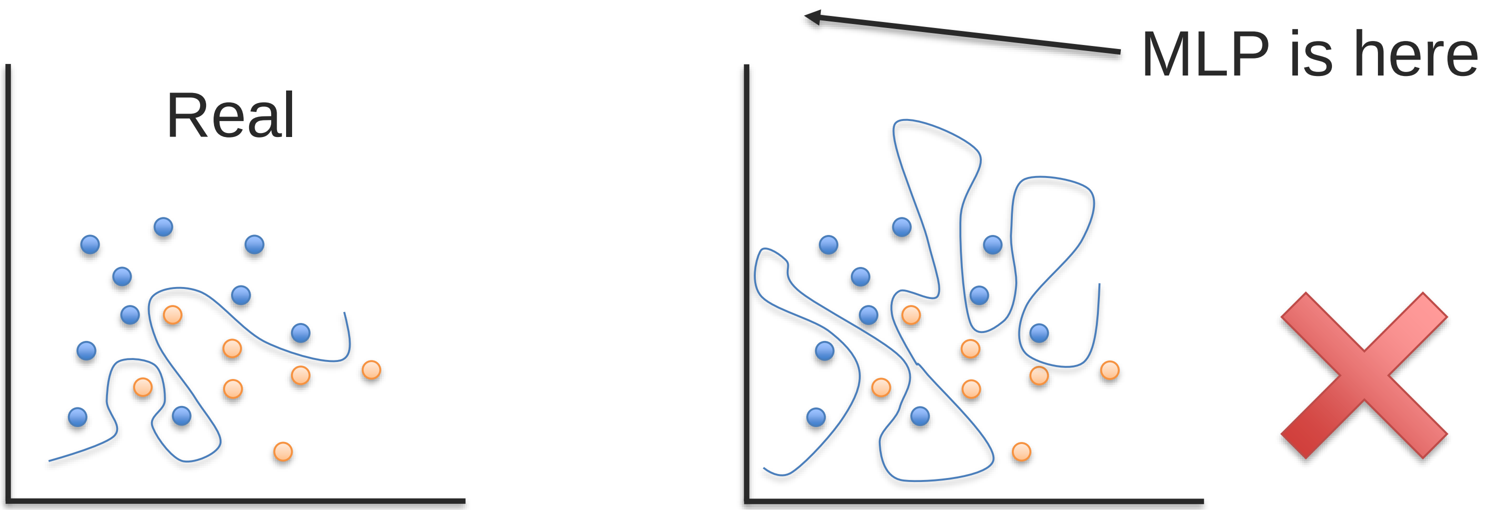
No longer SOT!





Bias-Variance

- Problem of bias and variance
  - Simple models are unlikely to find the solution to a hard problem, thus probability of finding the right model is low.
  - Complex models find many solutions to a problem, thus probability of finding the right model is again low.

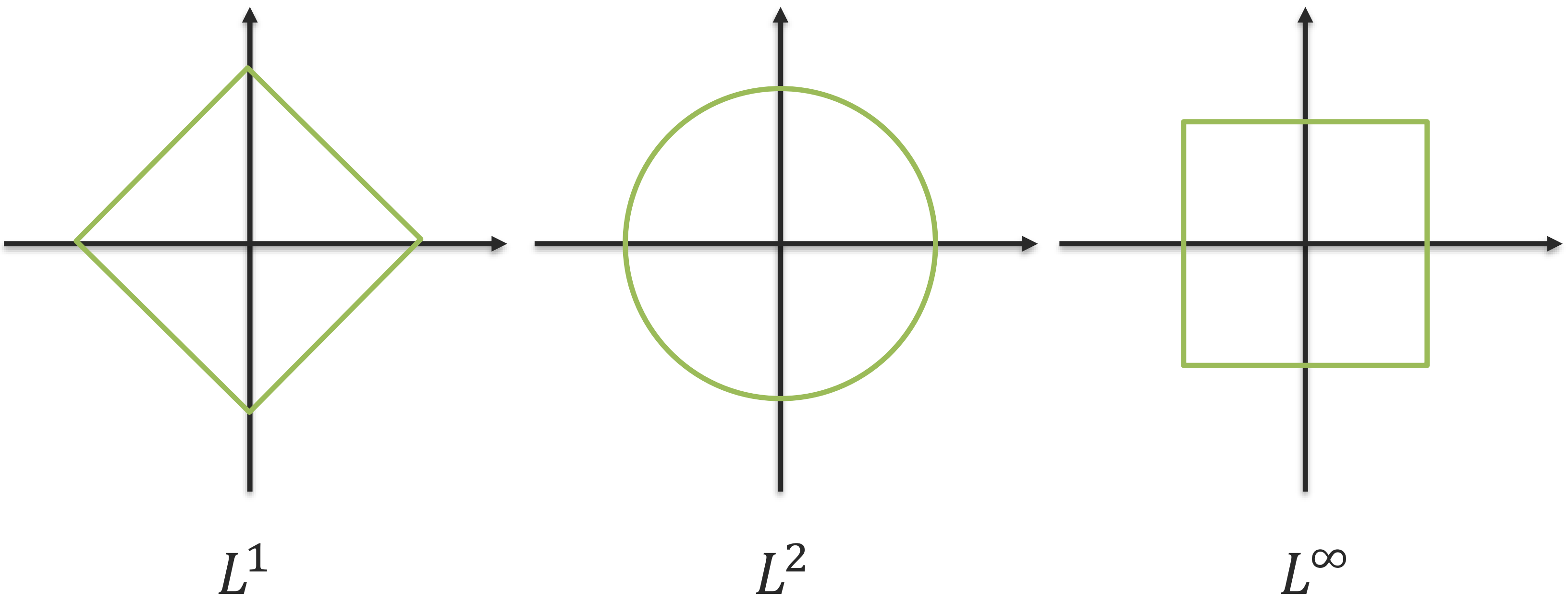


Lecture Objectives

- Practical Deep Model Optimization
  - Background
  - Optimization and Iterative approaches
  - Learning rate and Momentum
  - Regularization
  - Co-adaptation
  - Multimodal Optimization
- Variational Methods
  - Variational AE

Regularization

- Parameter Regularization:
  - Adding prior to the network parameters
  - L^p Norms



Minimize: $Loss(x;\theta) + \propto ||\theta||$

Parameter Regularization

- Parameter Regularization:
  - L^1(Lasso) and L^2 (Ridge) are the most famous norms used. Sometimes combined (Elastic)
  - Other norms are computationally ineffective.
- Maximum a posteriori (MAP) estimation:
  - Having priors one the model parameters
  - L^2 can be seen as a Gaussian prior on model parameters θ
  - A generalization of L^2 is called Tikhonov Regularization with Multivariate Gaussian prior on model parameters.
    - Assuming Correlation between parameters one can build a Mahalanobis variation of Tikhonov Regularization.

Structural Regularization

- Lots of models can learn everything.
- Go for simpler ones.
- Use task specific models:
  - CNNs
  - RecNNs
  - LSTMs
  - GRUs

Occam's razor

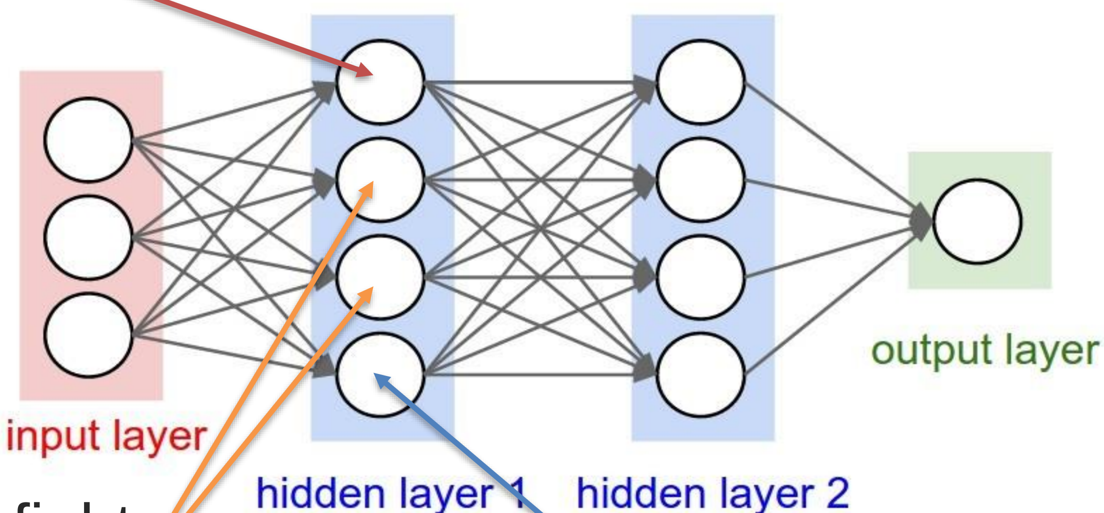
Lecture Objectives

- Practical Deep Model Optimization
  - Background
  - Optimization and Iterative approaches
  - Learning rate and Momentum
  - Regularization
  - Co-adaptation
  - Multimodal Optimization
- Variational Methods
  - Variational AE

Example

- A neuron learns something that is not useful:
  1. Learn something useful
  2. Other neurons learn to mitigate it.

Useless neuron

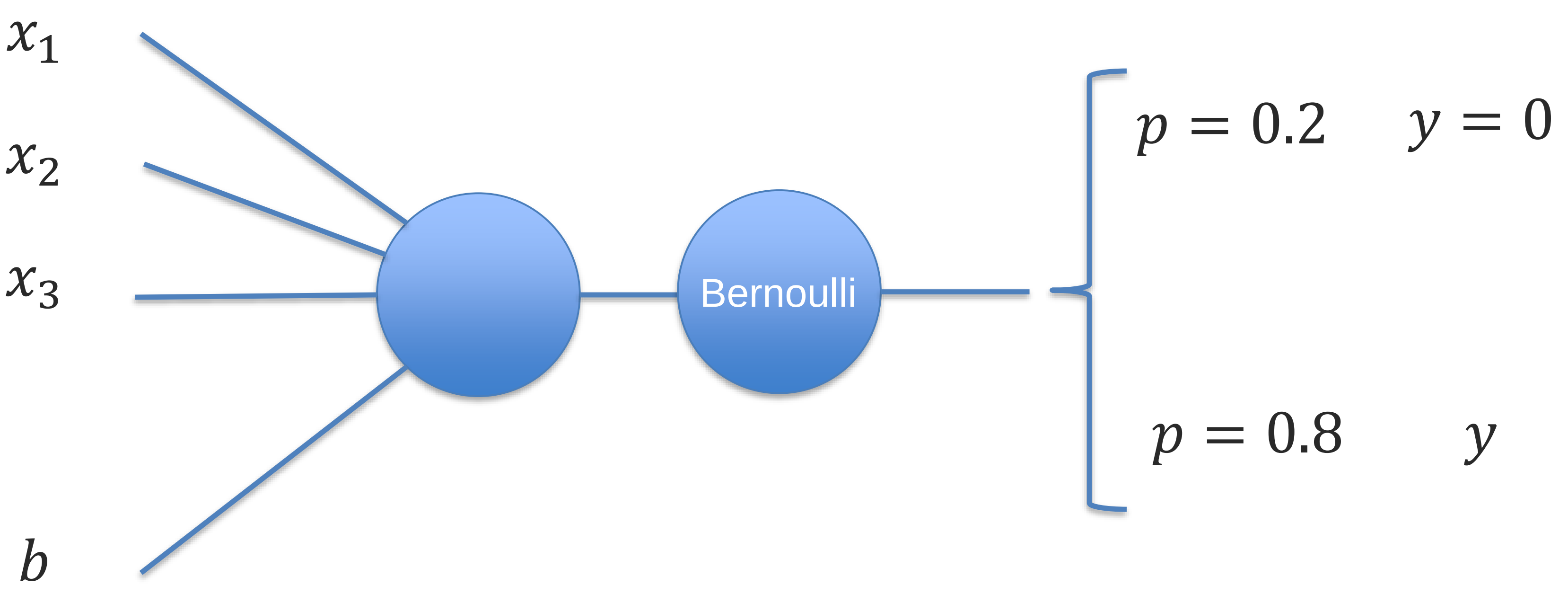


Learning to fight useless neuron

Actually learning something

Dropout

- Simply multiply the output of a hidden layer with a mask of 0s and 1s (Bernoulli)



Dropout

Forward step: multiply with a Bernoulli distribution per epoch, batch or sample point. Question: which one works better?

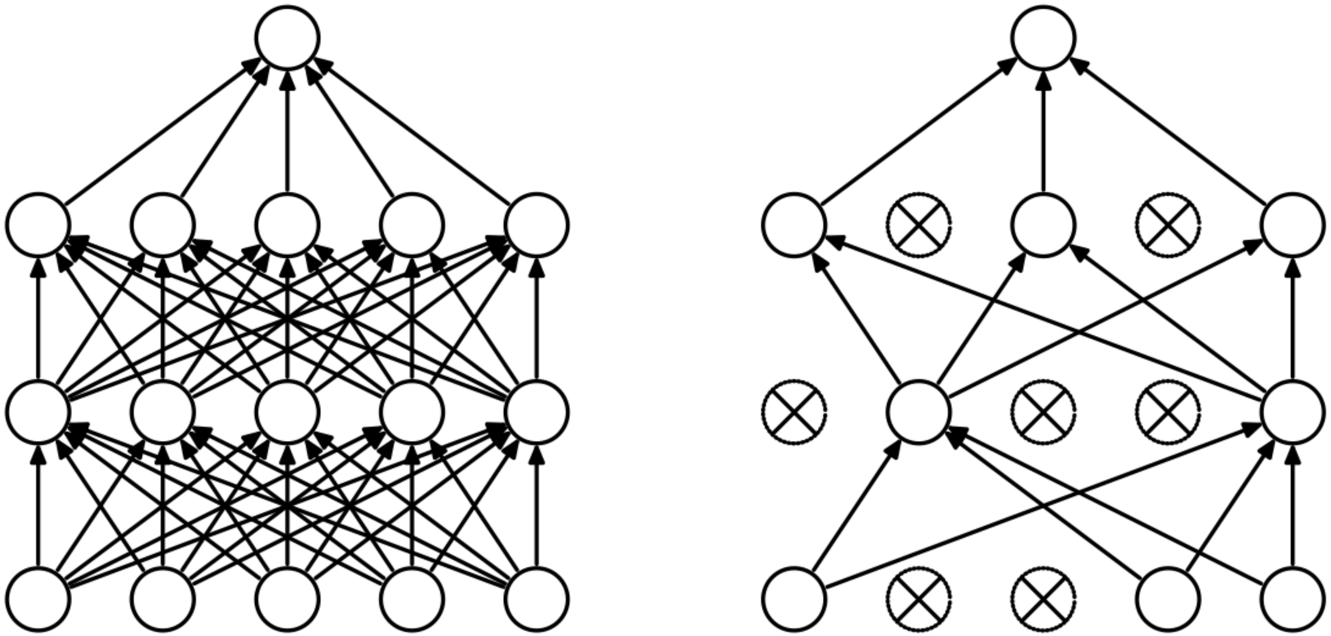
Backward step: just calculate the gradients same as before. Question: some neurons are out of the network, so how does this work?

All good? Nope

Multiply the weights by $1 - p_i$

Dropout

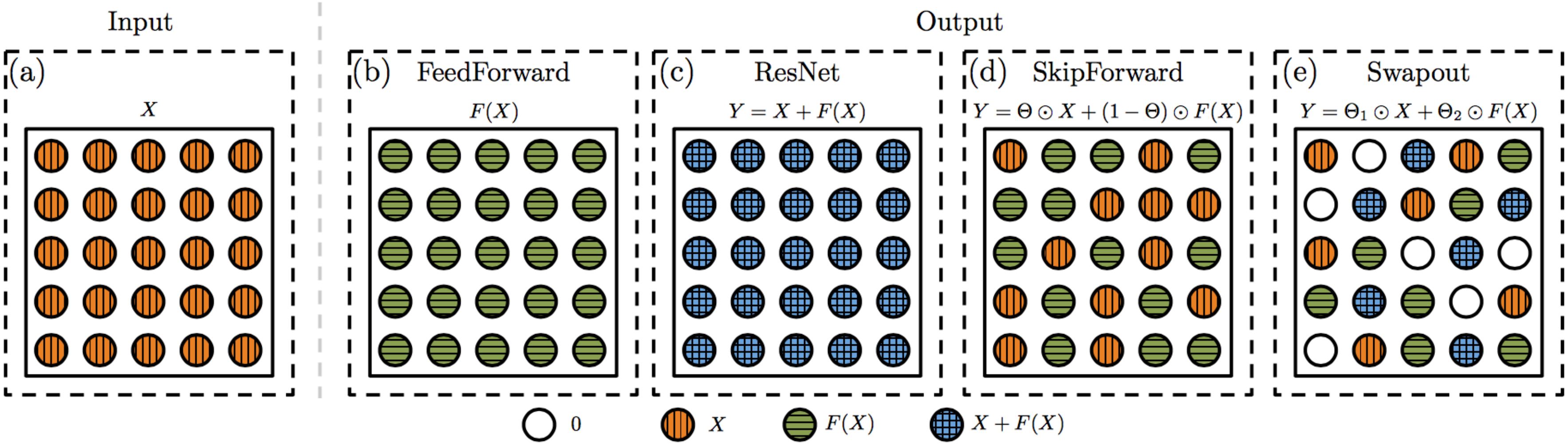
Stop co-adaptation + learn ensemble



Other variations

- Gaussian dropout: instead of multiplying with a Bernoulli random variable, multiply with a Gaussian with mean 1.

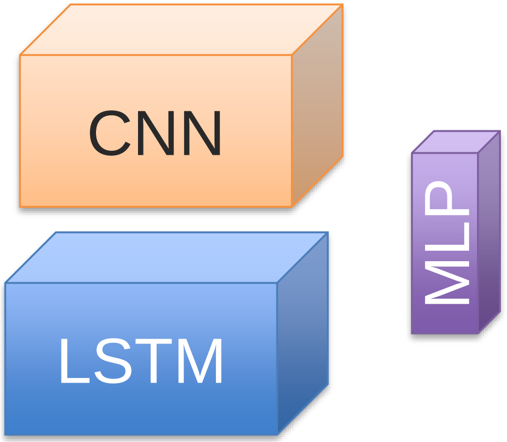
- Swapout: Allow skip-connections to happen



Lecture Objectives

- Practical Deep Model Optimization
  - Background
  - Optimization and Iterative approaches
  - Learning rate and Momentum
  - Regularization
  - Co-adaptation
  - Multimodal Optimization
- Variational Methods
  - Variational AE

Multimodal Optimization

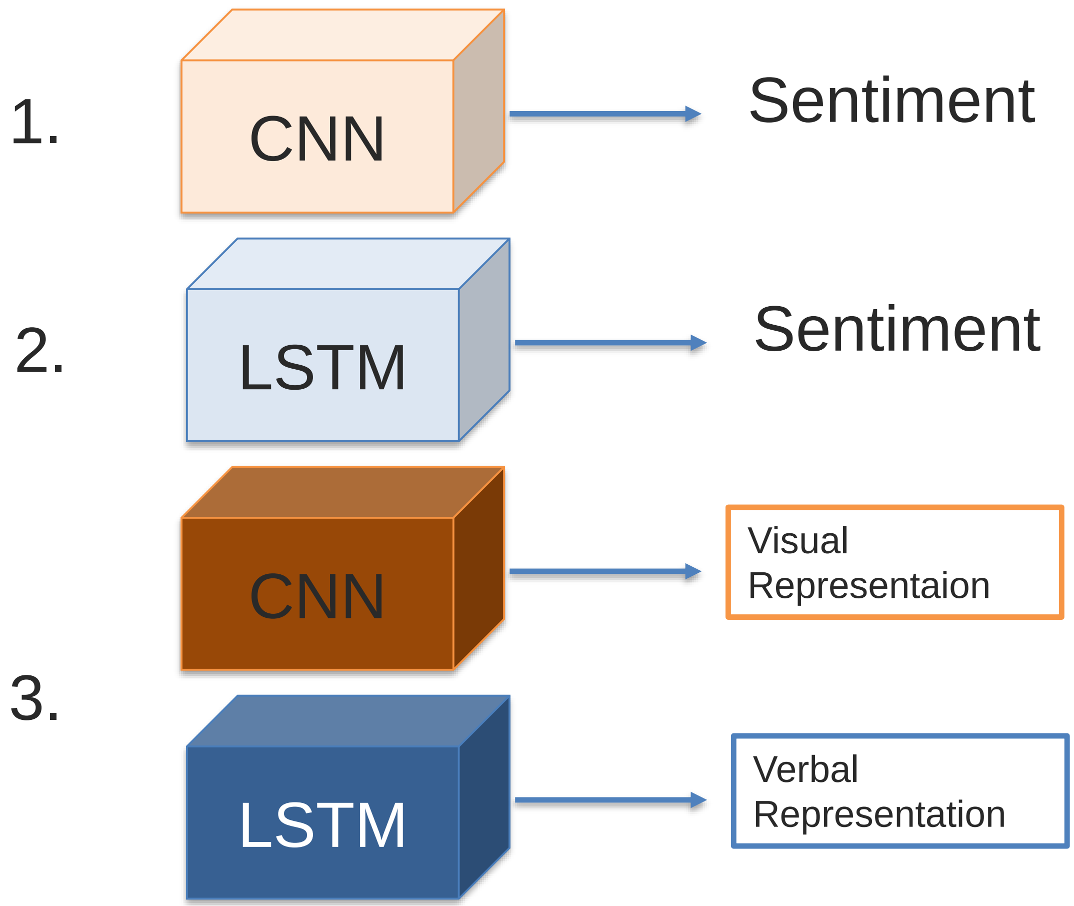


- Biggest Challenge:
  - Data from different sources
  - Different networks
- Example:
  - Question Answering: LSTM(s) connected to a CNN
  - Multimodal Sentiment: LSTM(s) fused with MLPs and 3D-CNNs
- CNNs work well with high decaying learning rate
- LSTMs work well with adaptive methods and normal SGD
- MLPs are very good with adaptive methods

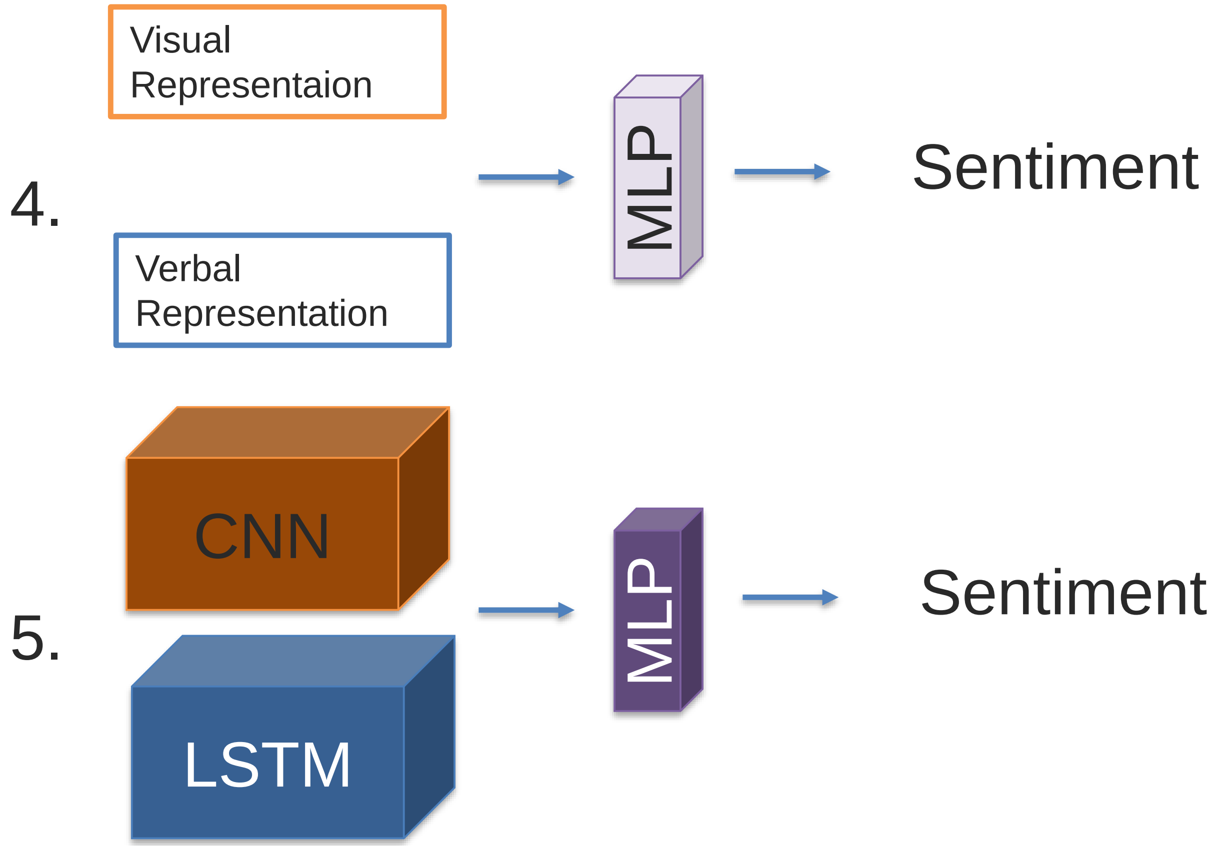
Multimodal Optimization

- How to work with all of them?
- Pre-training is the most straight forward way:
  - Train each individual component of the model separately
  - Put together and fine tune
- Example: Multimodal Sentiment Analysis

Pre-training



Pre-training



Pre-training Tricks

- In the final stage (5), it is better to not use adaptive methods such as Adam.
  - Adam starts with huge momentum on all the networks parameters and can destroy the effects of pretraining.
  - Simple SGD mostly helpful.
- Initialization from other pre-trained models:
  - VGG for CNNs
  - Language models for RNNs
  - Layer by layer training for MLPs

Questions?