Neural Networks

Iaroslav Shcherbatyi

Agenda

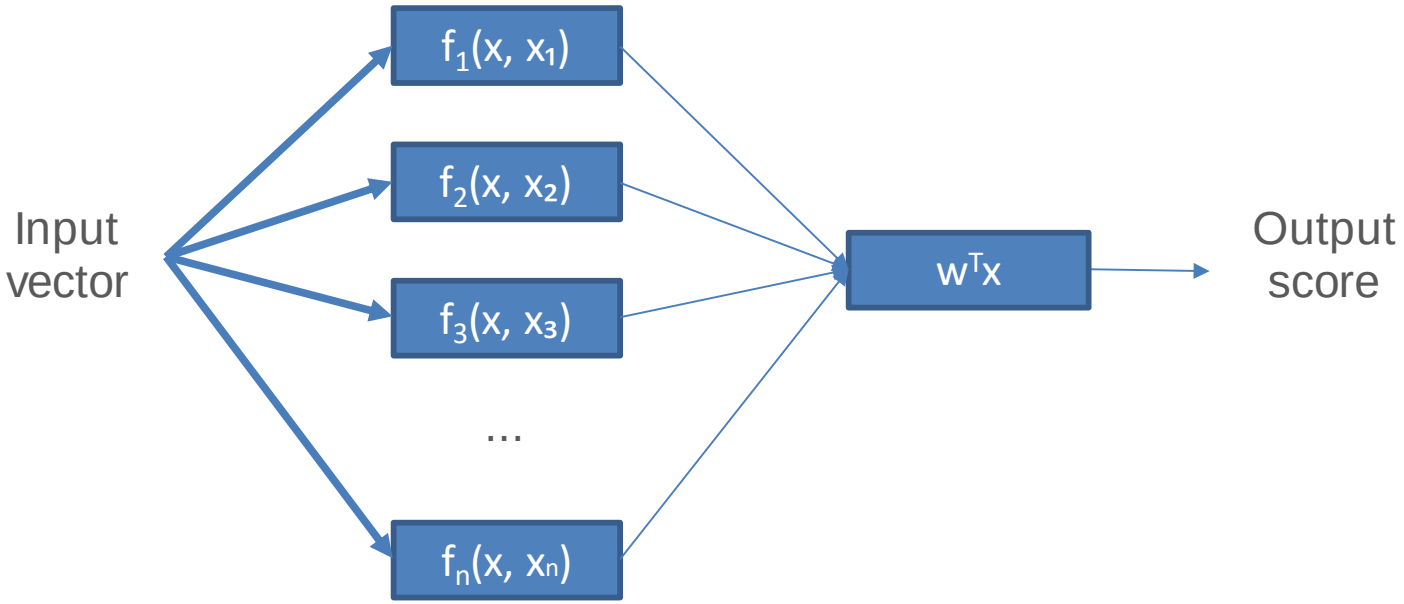
Neural networks

- Formal definition
  - Shallow neural networks
  - Live demo
  - Deep fully connected neural networks
  - Recurrent neural networks
  - Convolutional neural networks
- Advantages / Disadvantages

Model for Kernel SVM

Kernel SVM: Weighted sum of similarities with training points.

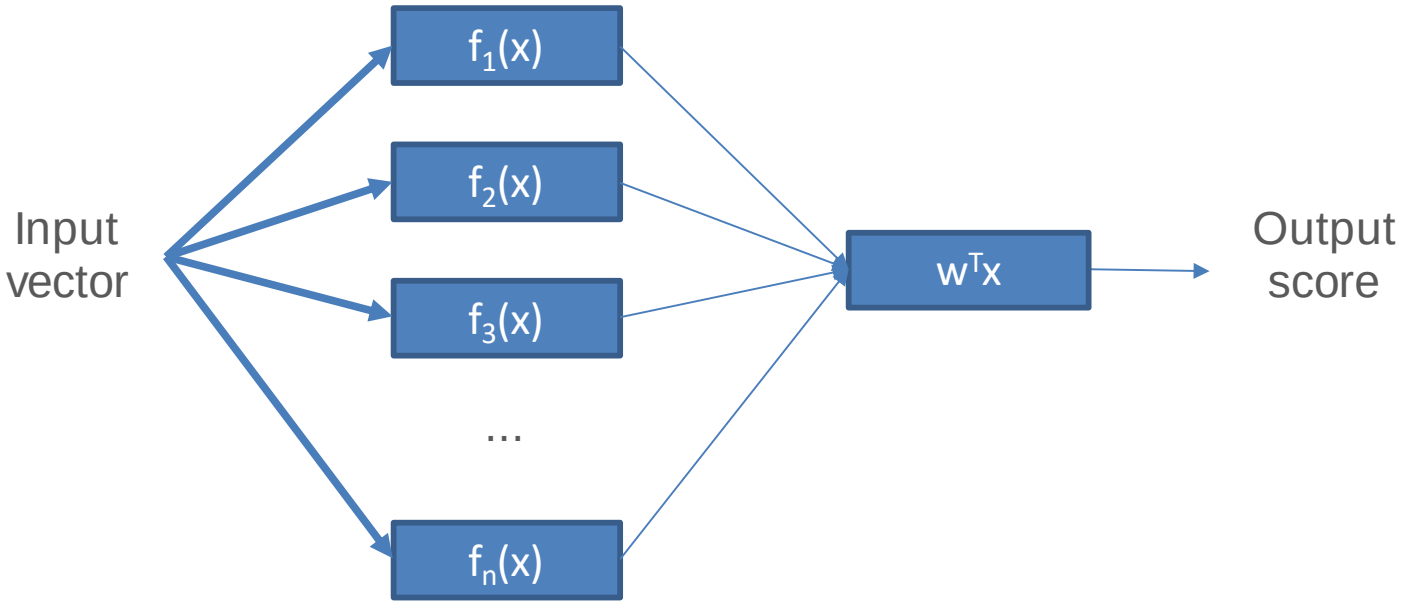
For every function block below, its input is denoted as x. Fat input arrows denote vector inputs, thin – scalar.



Model for Boosting

Boosting: [Weighted] sum of outputs of multiple weak learners.

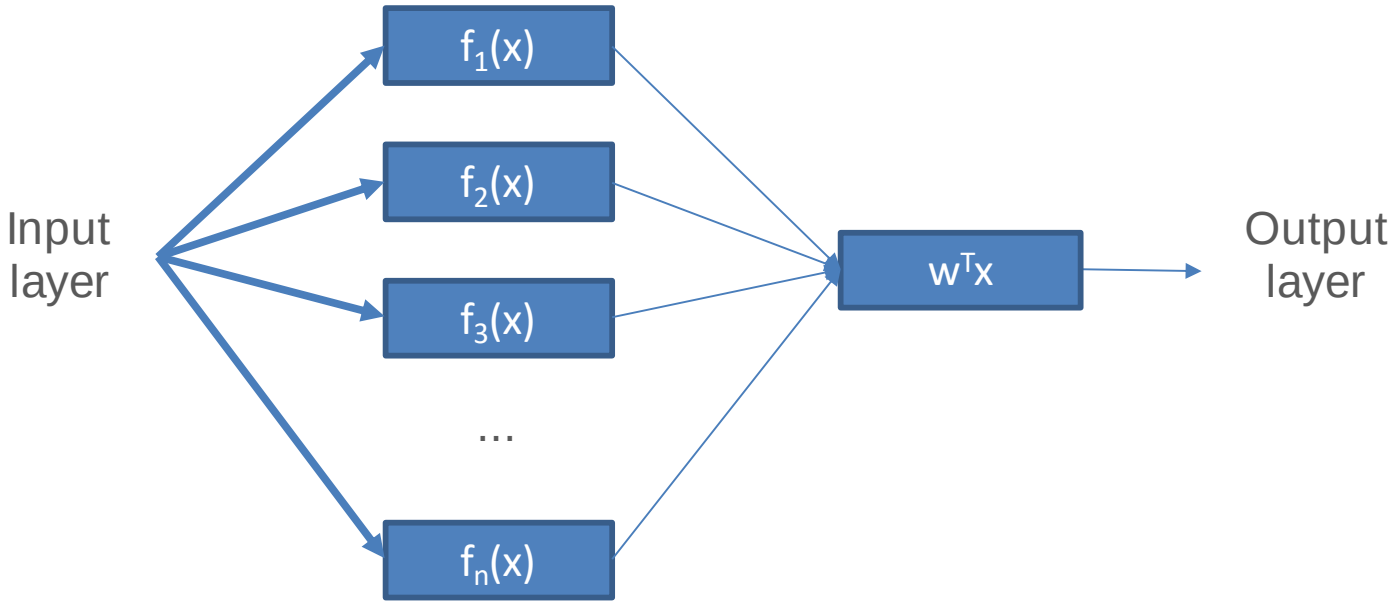
For every function block below, its input is denoted as x. Fat input arrows denote vector inputs, thin – scalar.



Shallow neural network

Shallow ANN: Weighted sum of outputs of multiple neurons.

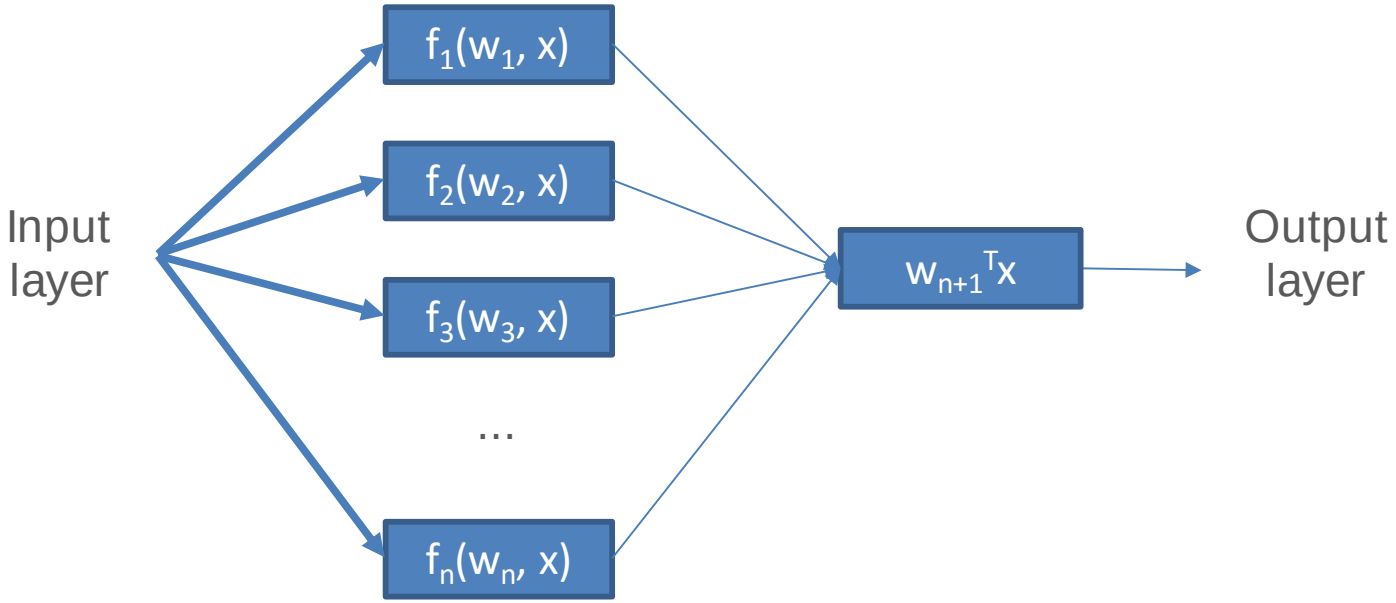
For every function block below, its input is denoted as x. Bold input arrows denote vector inputs, thin – scalar.



Shallow neural network

Parameters of **all** neurons are optimized simultaneously.

Here w_i are parameters of i-th block.



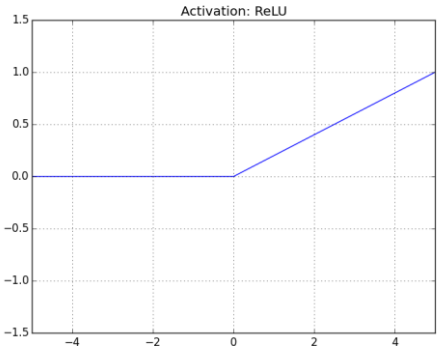
NN's neurons

Neurons are typically functions of the following form are considered:

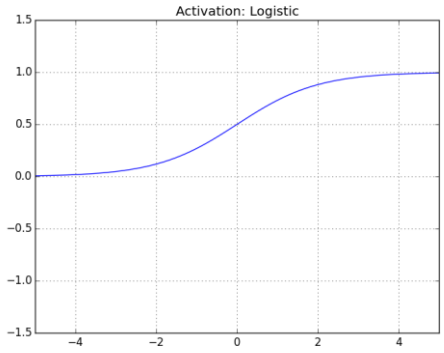
$$f(w,x)=\sigma(w_p^T x+w_b)$$

Sigma is called "squashing" function; If it is non – linear, than NN (even shallow) is a universal approximator.

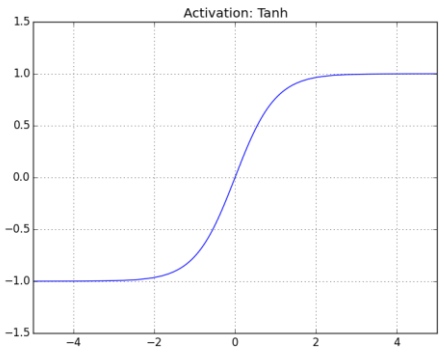
$$\sigma(a) = max(0, a)$$



$$\sigma(a) = 1/(1 + exp(-x))$$



$$\sigma(a) = tanh(a)$$



Training of NN

A typical approach – find parameters which minimize:

$$\min_{w \in W} C \sum_{i=1...n} l(f(w, x_i), y_i) + r(w)$$

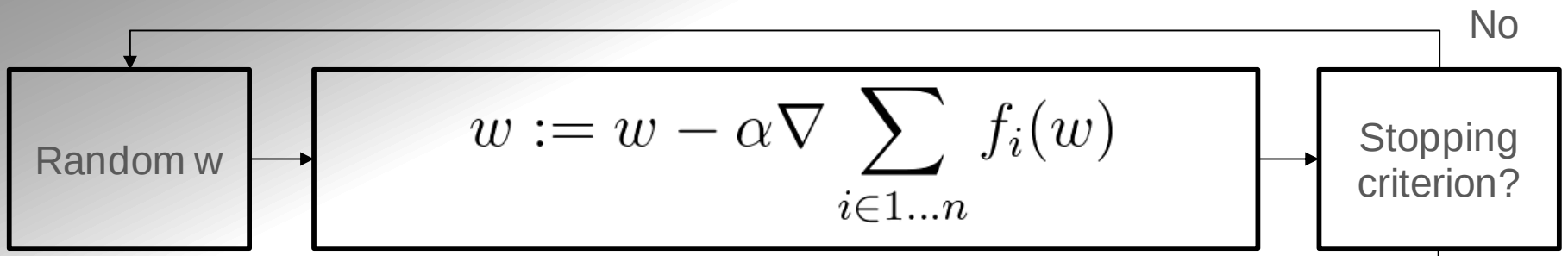
For example, for regression and with L2 regularization:

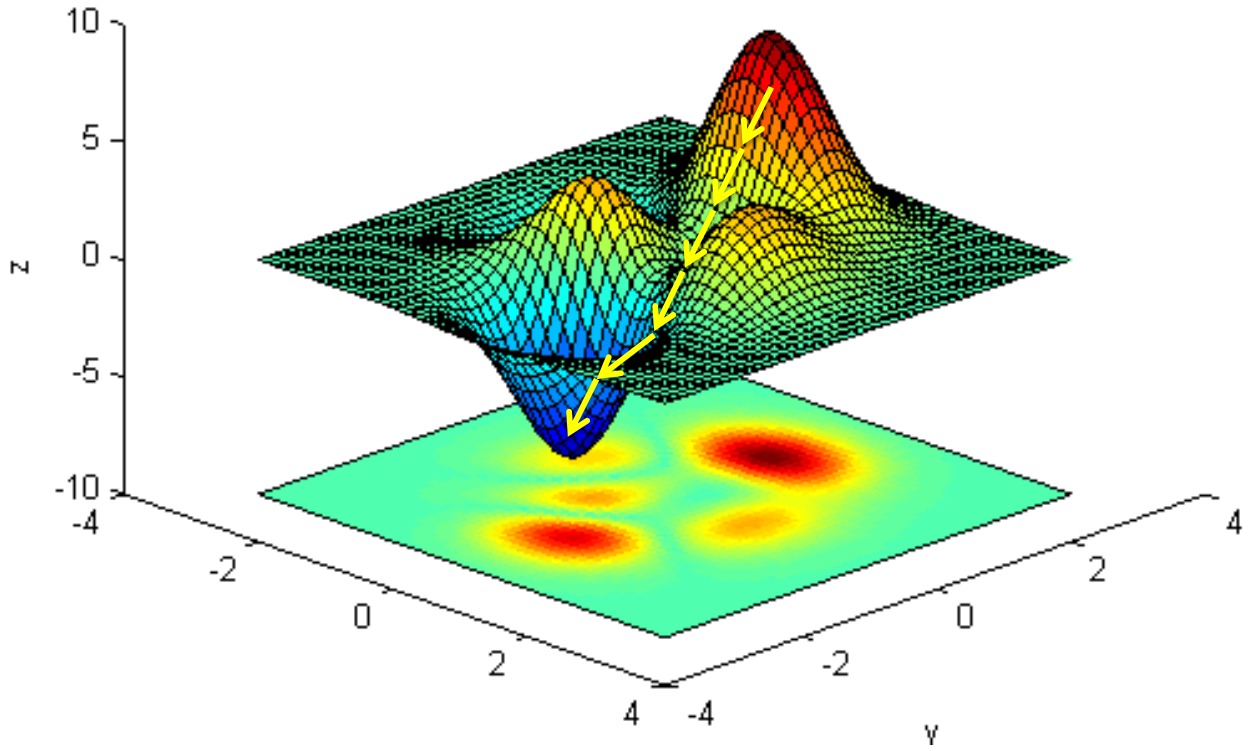
$$\min_{w \in W} C \sum_{i=1...n} (f(w, x_i) - y_i)^2 + ||w||_2^2$$

Here w contains all the weights of neural net.

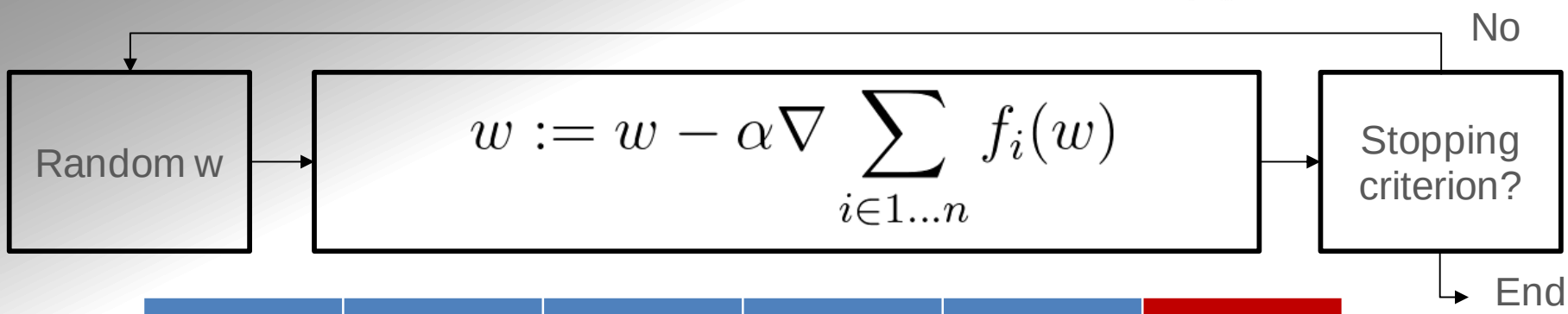
Usually solved using some flavor of Stochastic Gradient Descent.

Gradient Descent





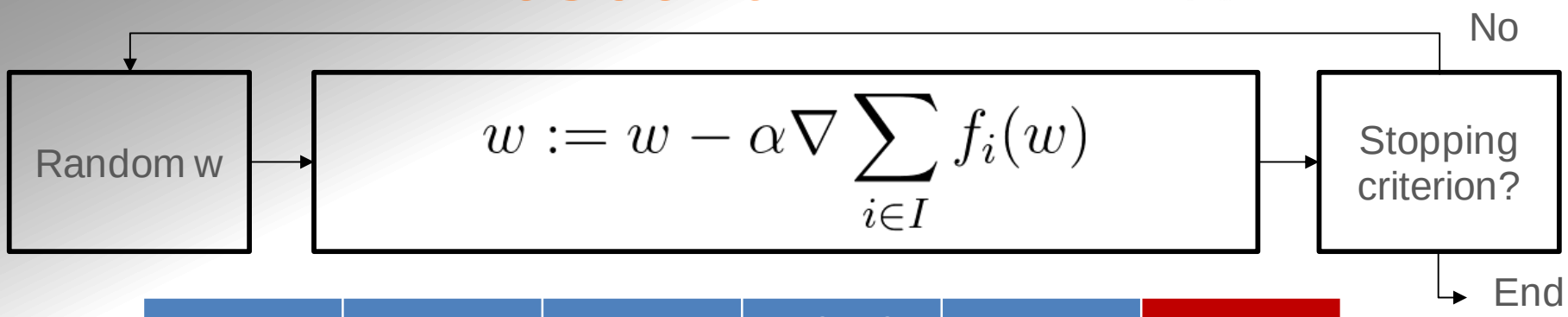
Gradient Descent



Age	Gender	Pain type	Blood pressure	Oldpeak	Sick
70	1	4	130	2.4	yes
67	0	3	115	1.6	no
57	1	2	124	0.3	yes
64	1	4	128	0.2	no
74	0	2	120	0.2	no
65	1	4	120	0.4	no
56	1	3	130	0.6	yes

Use whole dataset

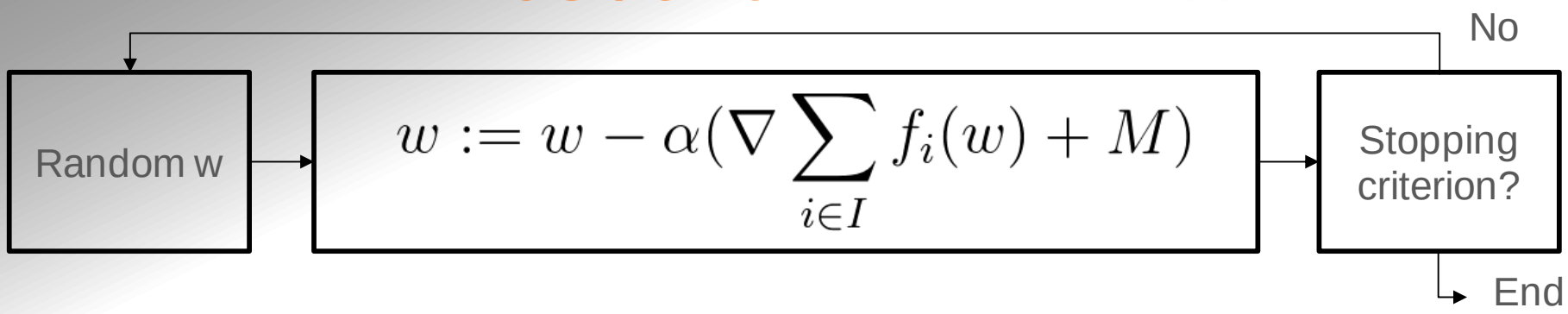
Stochastic Gradient Descent

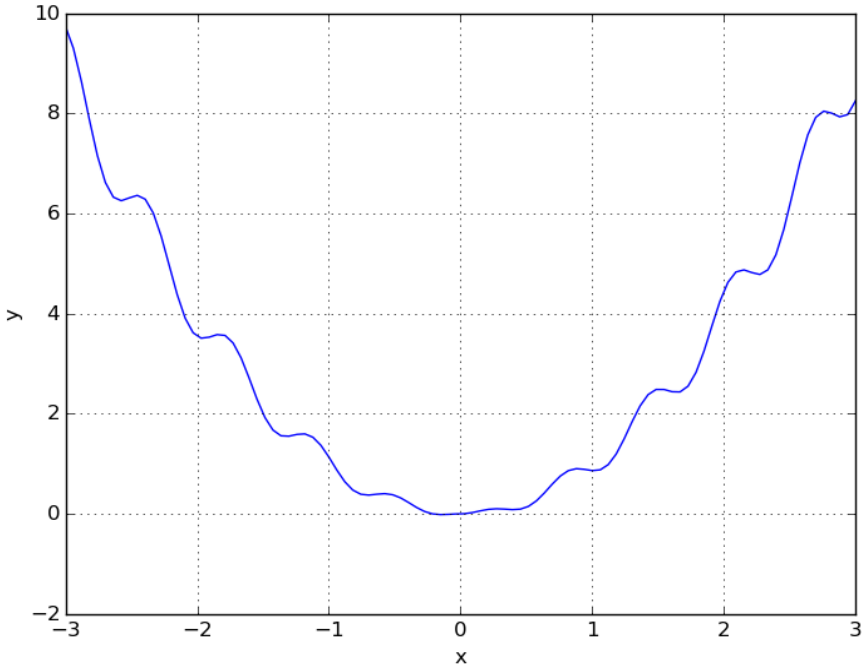


Age	Gender	Pain type	Blood pressure	Oldpeak	Sick
70	1	4	130	2.4	yes
67	0	3	115	1.6	no
57	1	2	124	0.3	yes
64	1	4	128	0.2	no
74	0	2	120	0.2	no
65	1	4	120	0.4	no
56	1	3	130	0.6	yes

Use subset of dataset

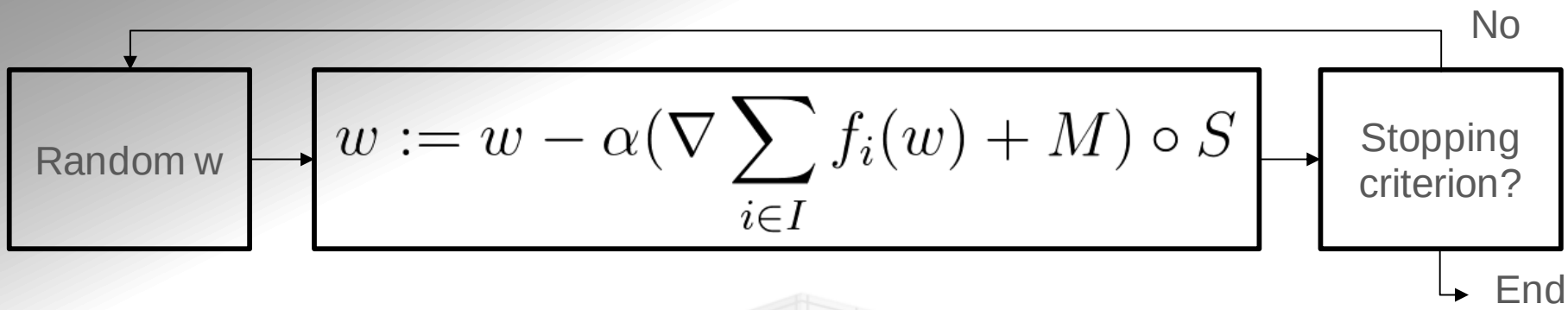
Stochastic Gradient Descent

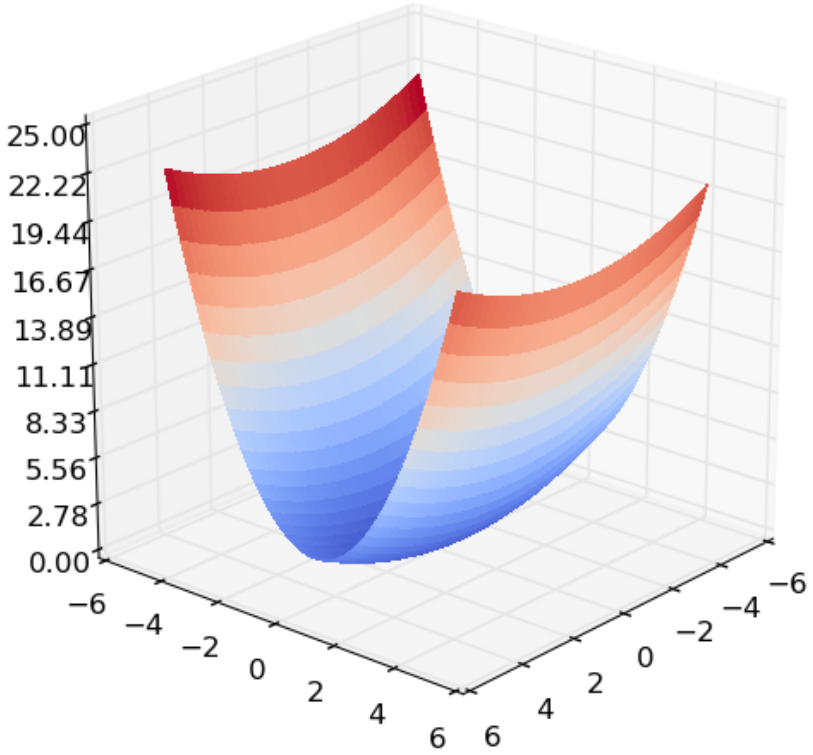




Momentum helps to avoid local minima

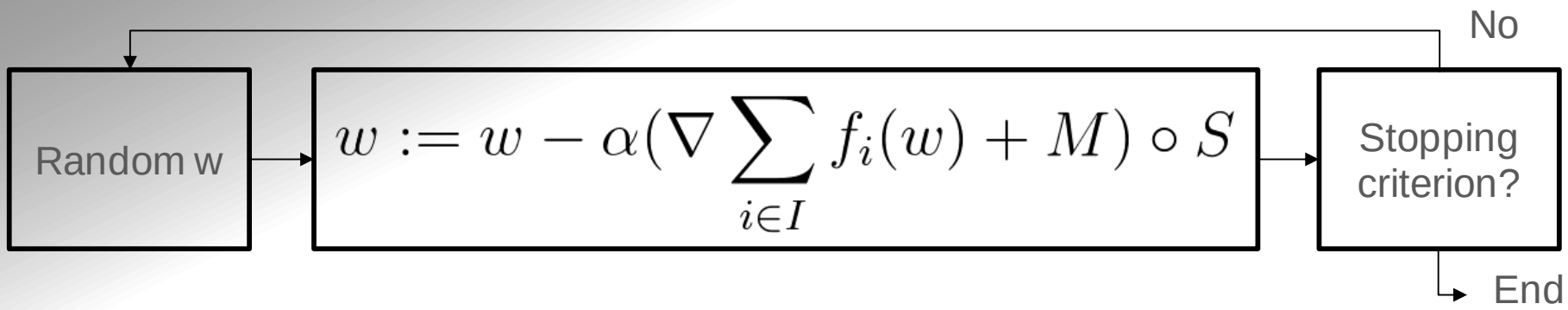
Stochastic Gradient Descent

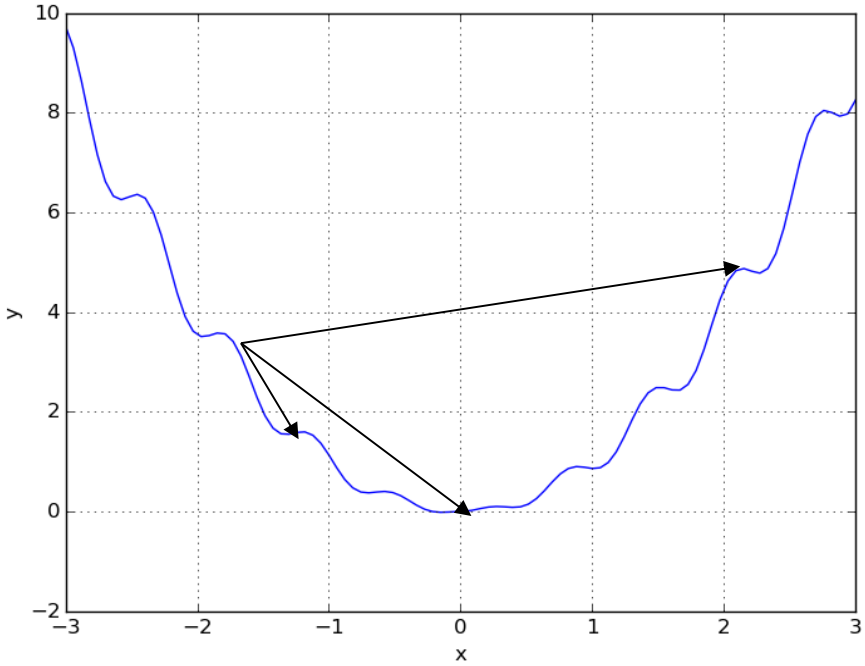




Scaling has effect similar to normalization

Stochastic Gradient Descent





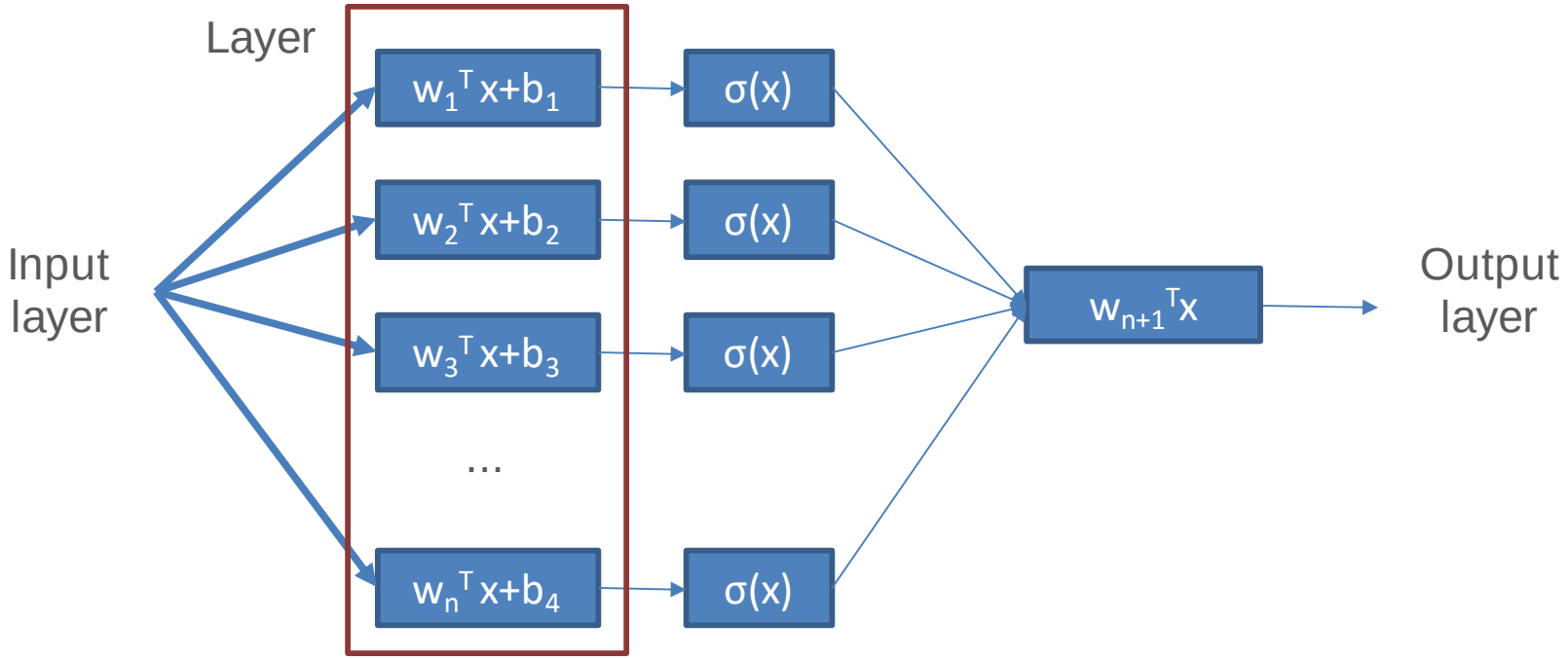
Too small alpha: slow convergence

Too large alpha: divergence

Layer-wise representation of NN's

Popular architectures can be represented conveniently as a sequence of layers.

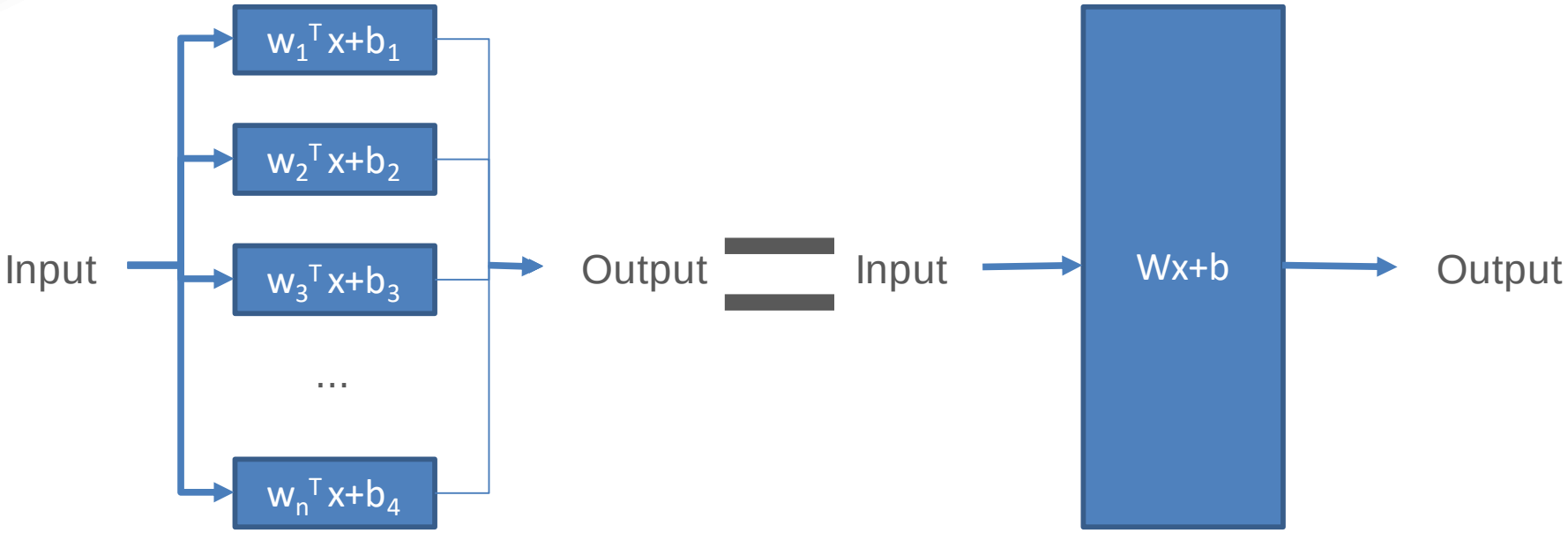
Layer is a vector valued function of vector input.



Linear function layer

Linear layer is a convenient representation of outputs of linear functions used inside neurons.

$W^T x$ is a matrix multiplication of matrix W and vector x.

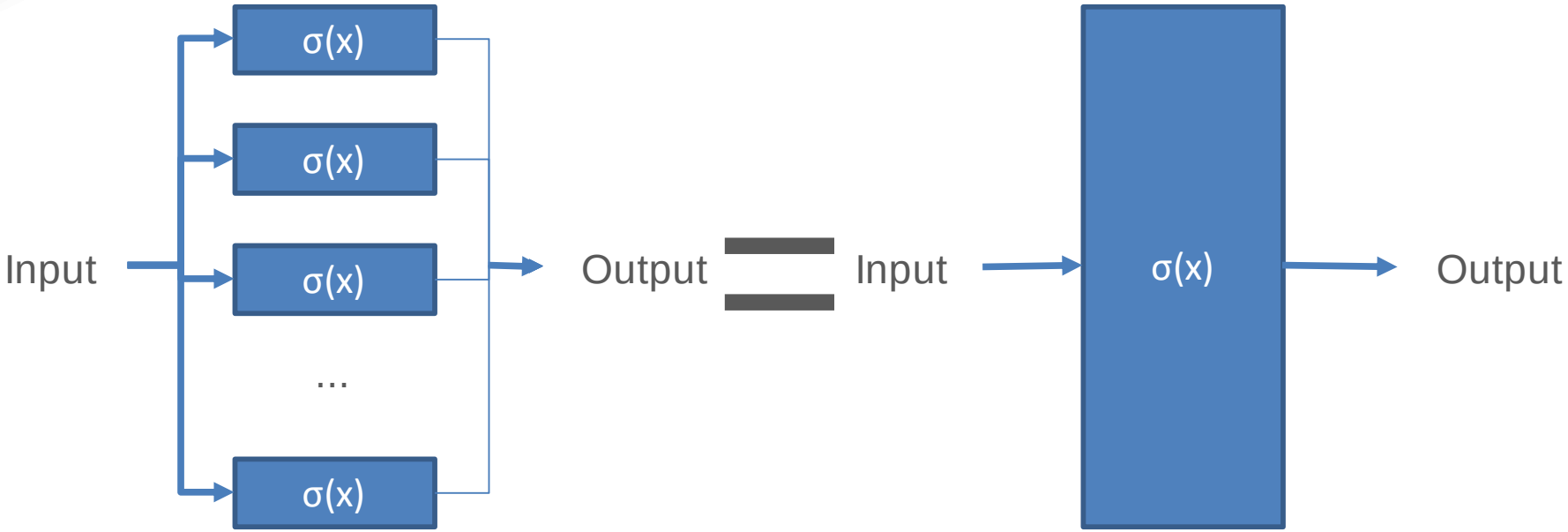


Linear function layer

Activation layer takes as input a vector x, and outputs vector y of same size as x, where it holds for every i = 1 … n

$$y_i = \sigma(x_i)$$

where n is a size of input vector x.

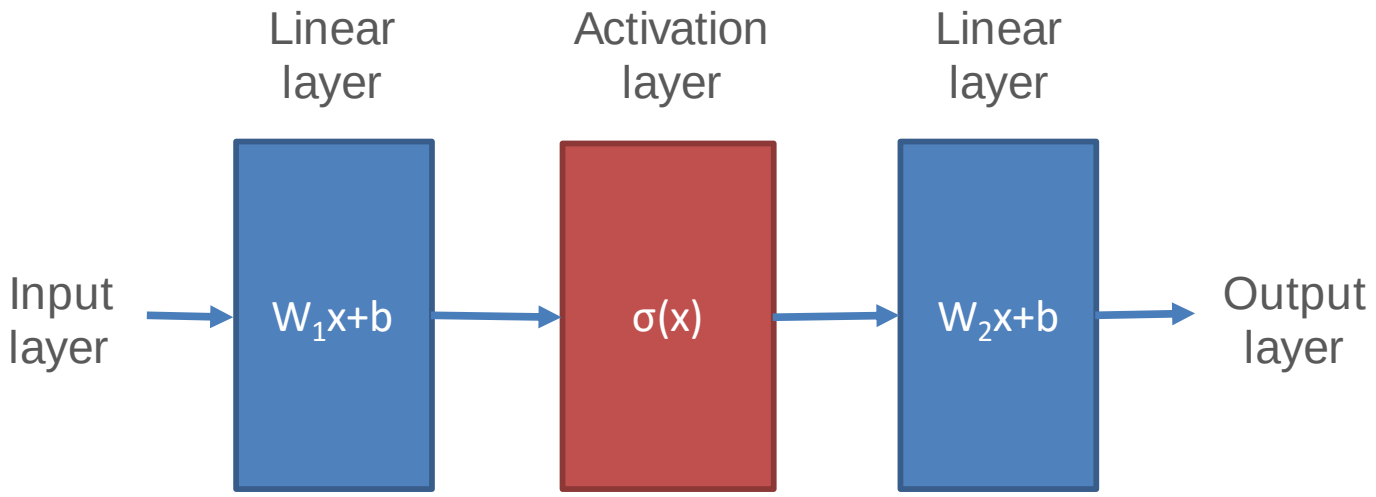


Layer representation of NN

Architecture selection for neural networks is (typically) done on the layer level.

Layers below are represented as blocks.

Red blocks: no parameters, blue: with parameters.

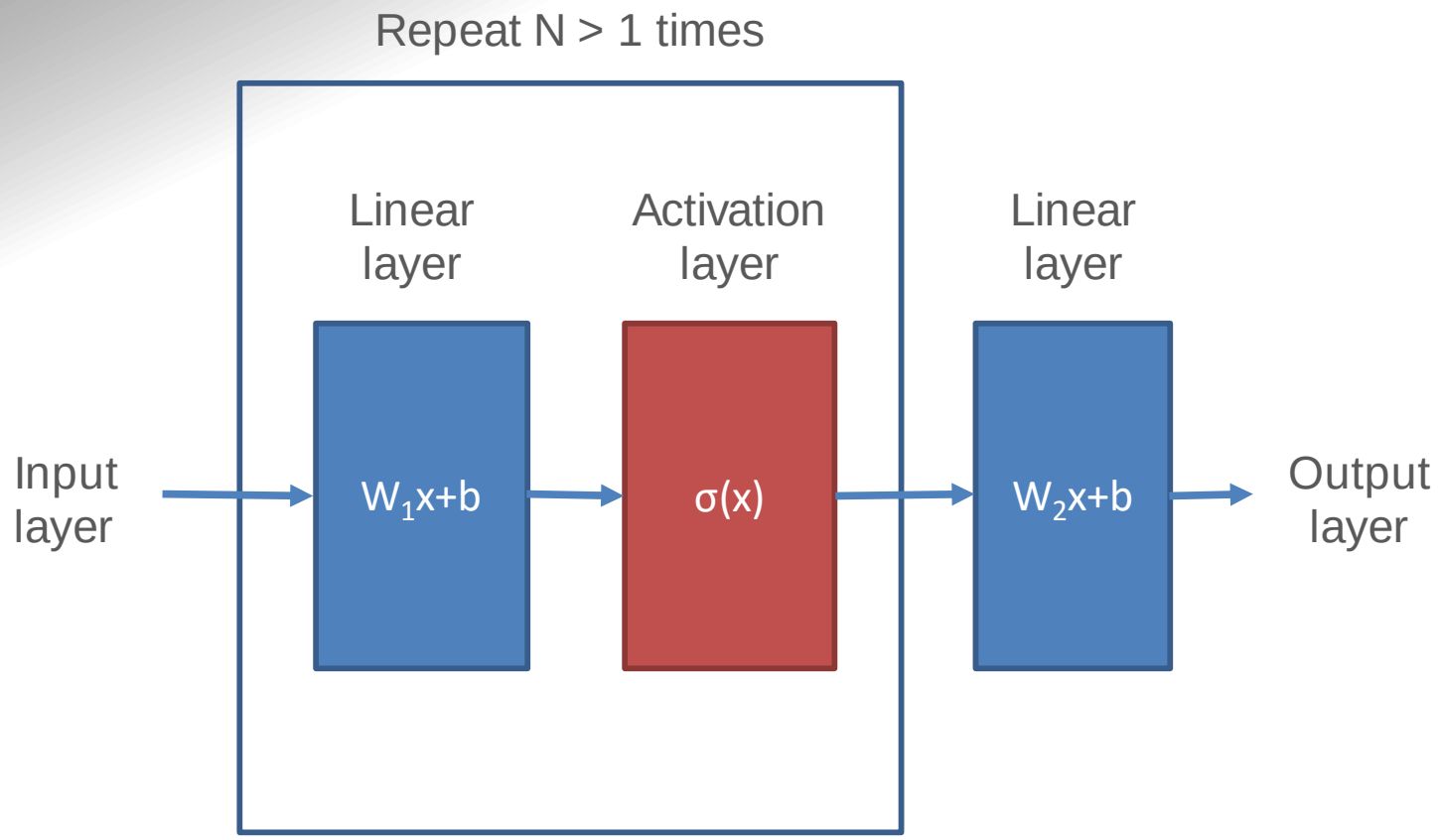


Online demo

Jupyter notebook demo

nn_example.ipynb

Deep neural networks

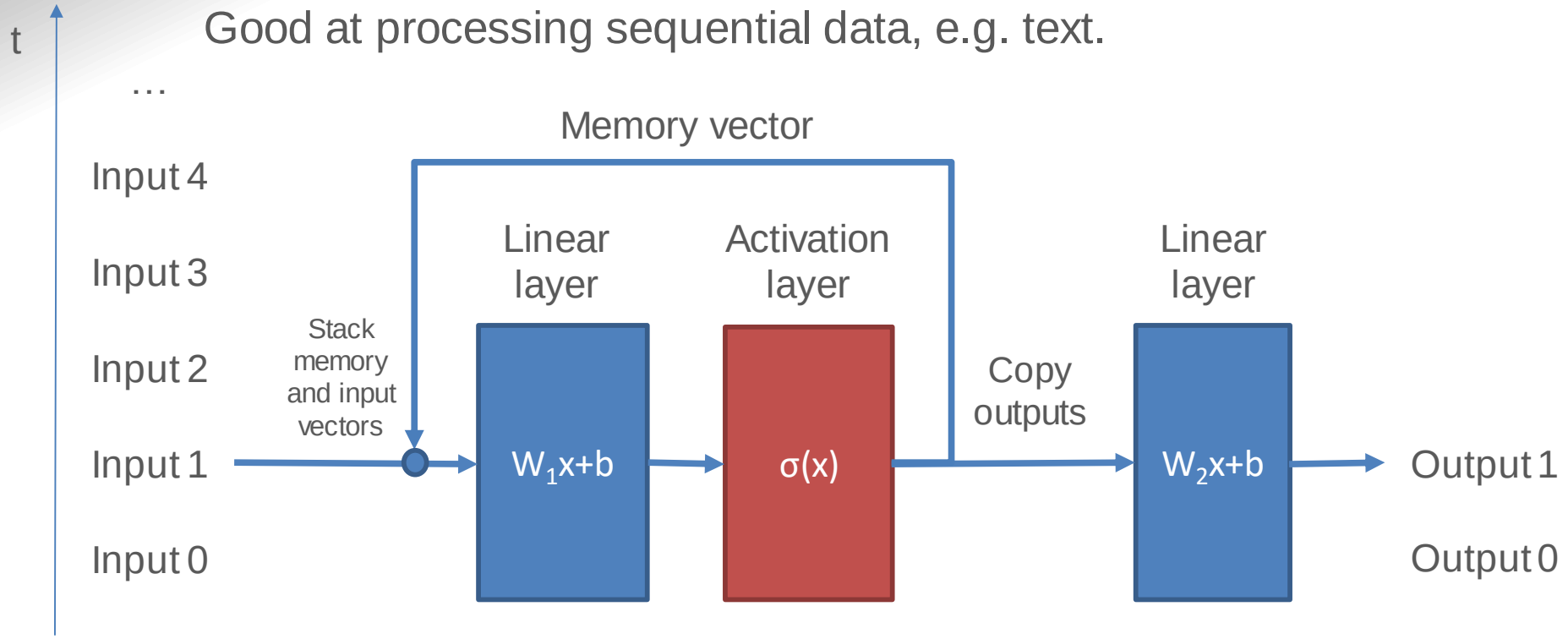


Recurrent neural networks

Store activations between different inputs (“have memory”).

Work with sequences of arbitrary size. Memory vector is reset between sequences.

Good at processing sequential data, e.g. text.



Sentiment analysis: use case for RNN

Is the customer feedback positive or negative?

This film isn't for all people. That's to say about a lot of movies in general of course, but this one in particular brings up a big clashing point between critics; What do we want to see in our movies? What is more important, to portray a fictional setting for the sake of giving people a mind blowing visual experience or to amuse and amaze them with clever plot twists and intelligent dialogs?

First lets analyze what exactly this film is made of. Basically, the whole thing is just one epic fighting scene after another. Most noticeably is the camera work and the visual effects. Every shot seems like it was intended to be a work of art. The colors, the characters, the costumes, the backgrounds... every little detail has been given so much attention. During the big fights you'll also instantly notice the unique editing. There are a lot of "time slowdowns" throughout the battles which show what exactly is happening. Fatal wounds that slowly leak blood spatters in the air, decapitated heads traveling in slow-motion across the screen... it's all there.

https://www.kaggle.com/c/word2vec-nlp-tutorial/data

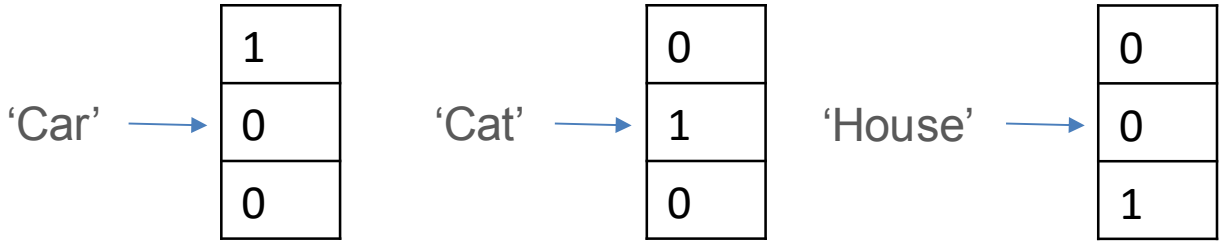
Text encoding for RNN

Words in encoded form are fed to the neural network. To obtain such encoding, the following steps are taken:

1. Generate vocabulary of all unique words from the training text
2. To encode the word with its index in the vocabulary
3. Use one hot encoding on word indexes

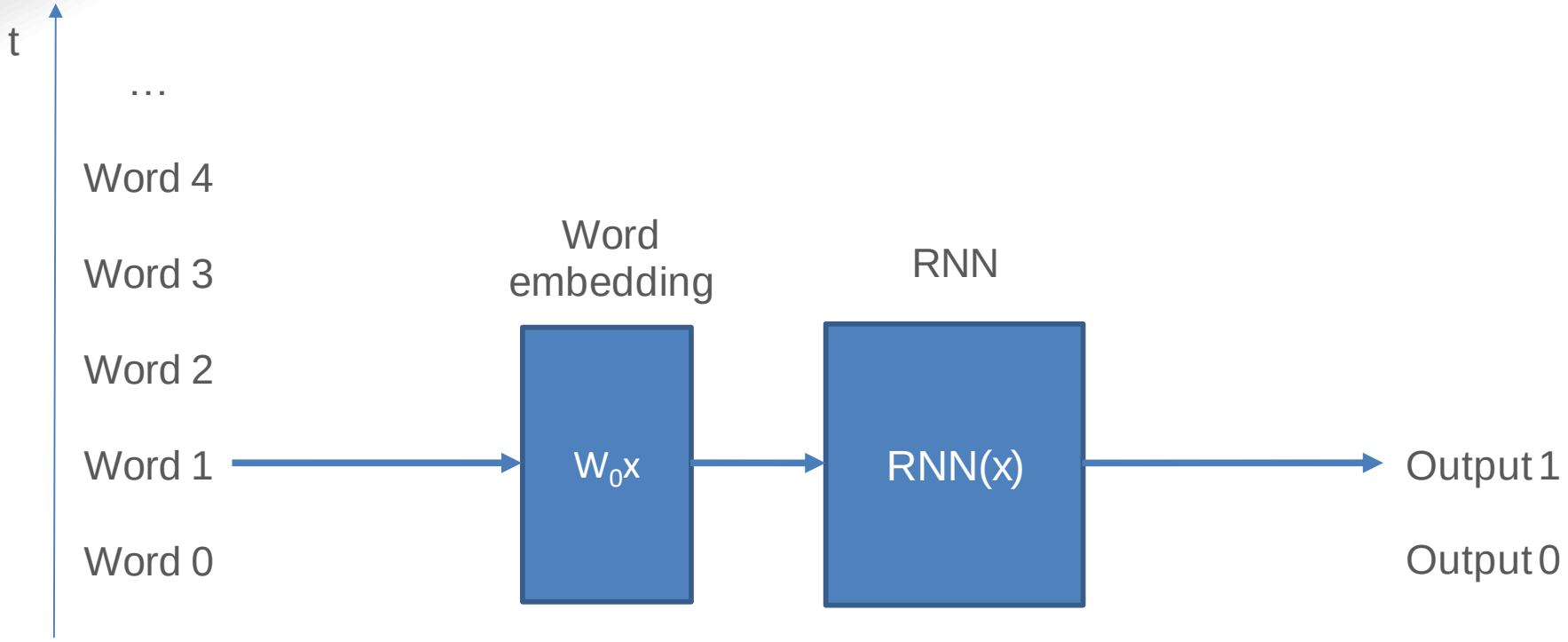
Either create a category for all words missing in dictionary or skip them.

Vocabulary:
{'Car', 'Cat','House'}



Word embedding

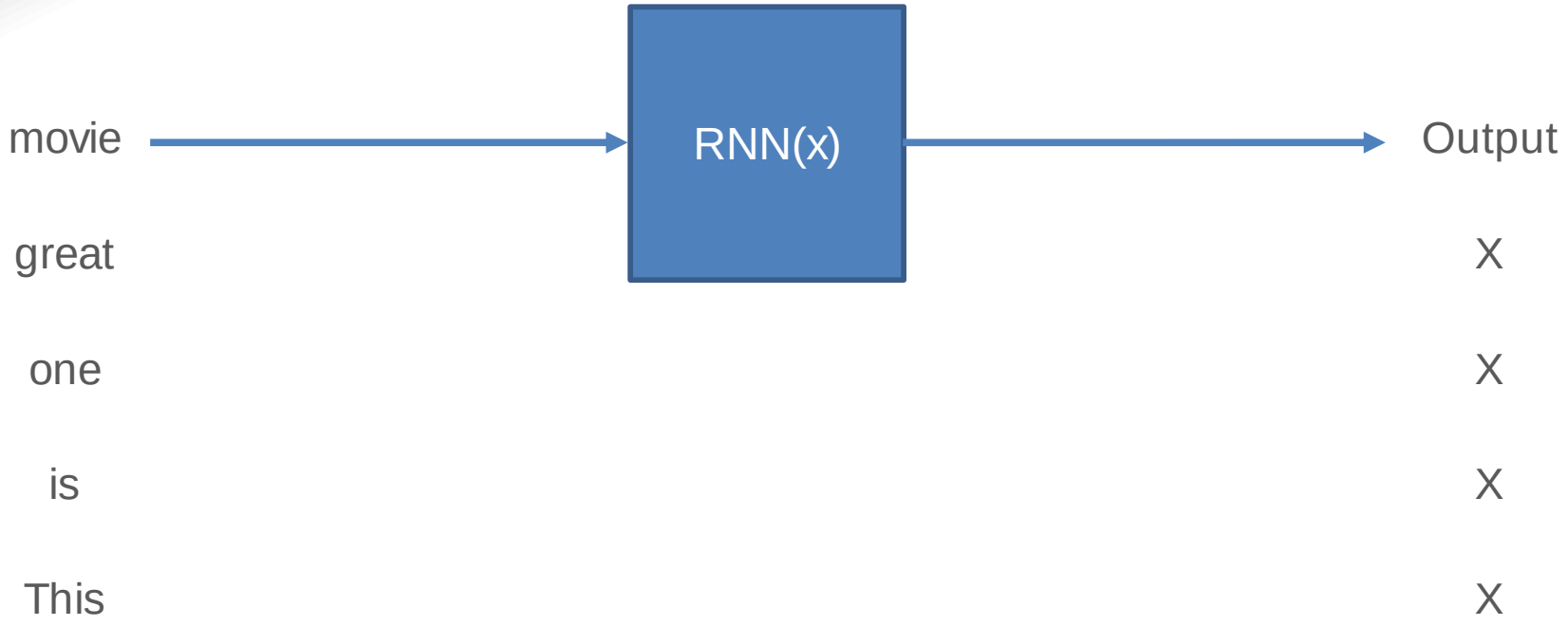
Word encoding can produce very high dimensional vectors. Word embedding reduces this dimension by applying a linear transformation to the word vectors



Estimations with RNN

RNN as final layer:

Reset RNN memory. Then, allow RNN to “read” the whole text, and after it is done, take the output.



Types of RNN

“Vanilla” RNN: the most simple type of recurrent neural networks. Has relatively few parameters, but does not work well with long sequences.

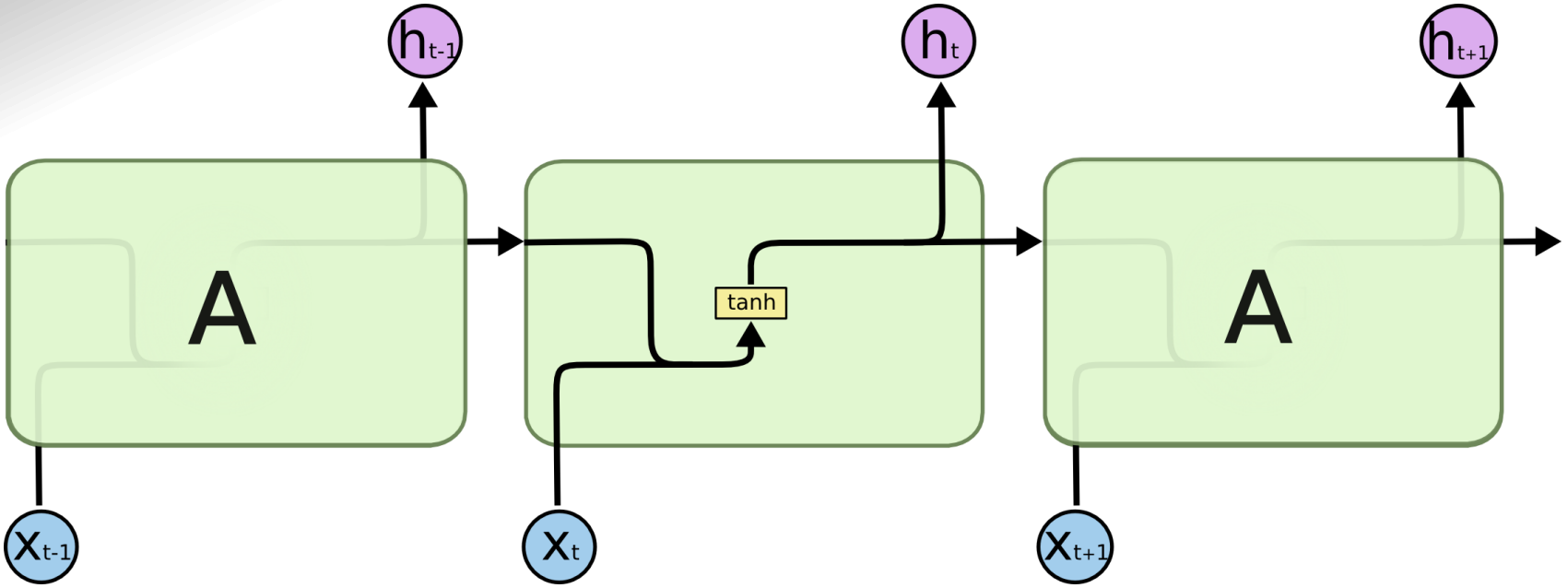
Long Short Term Memory units: have more parameters per block compared to simple RNN, but in practice work better with long sequences.

Gated recurrent units: work better than simple RNN on long sequences, but have less parameters compared to LSTM.

Is an active research area.

'Vanilla' RNN

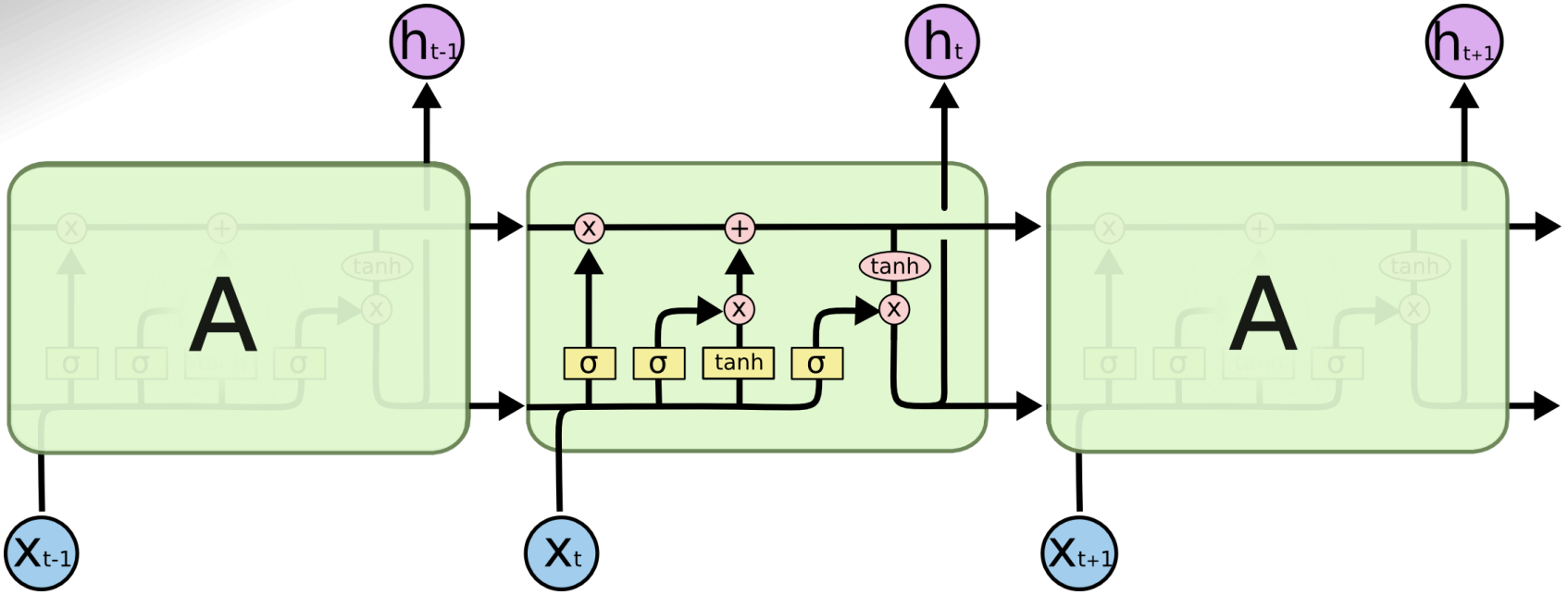
Shallow NN with recurrent connection.



Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short Term Memory Networks

Work better with longer sequences compared to Vanilla RNN

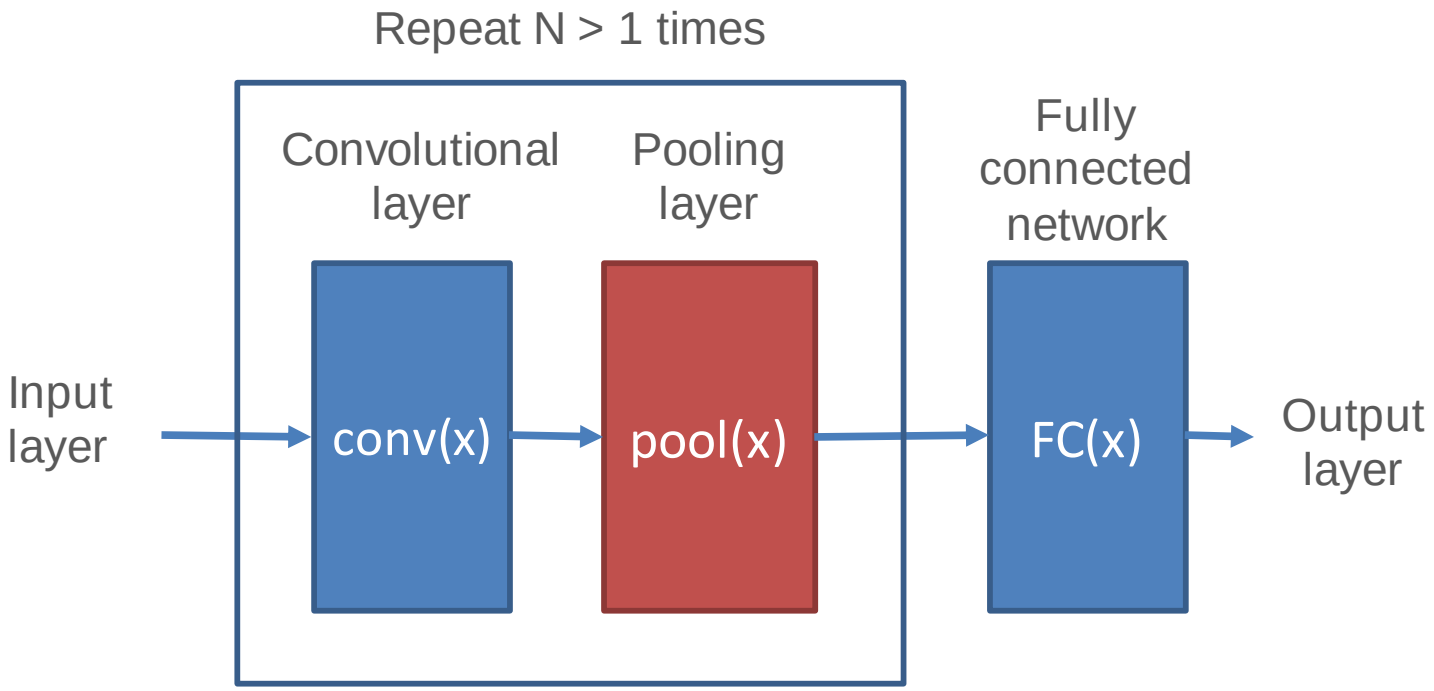


Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Convolutional neural networks

Operates directly on images.

Have “special” layers – convolutional and pooling. Pooling layer: downsize input image.



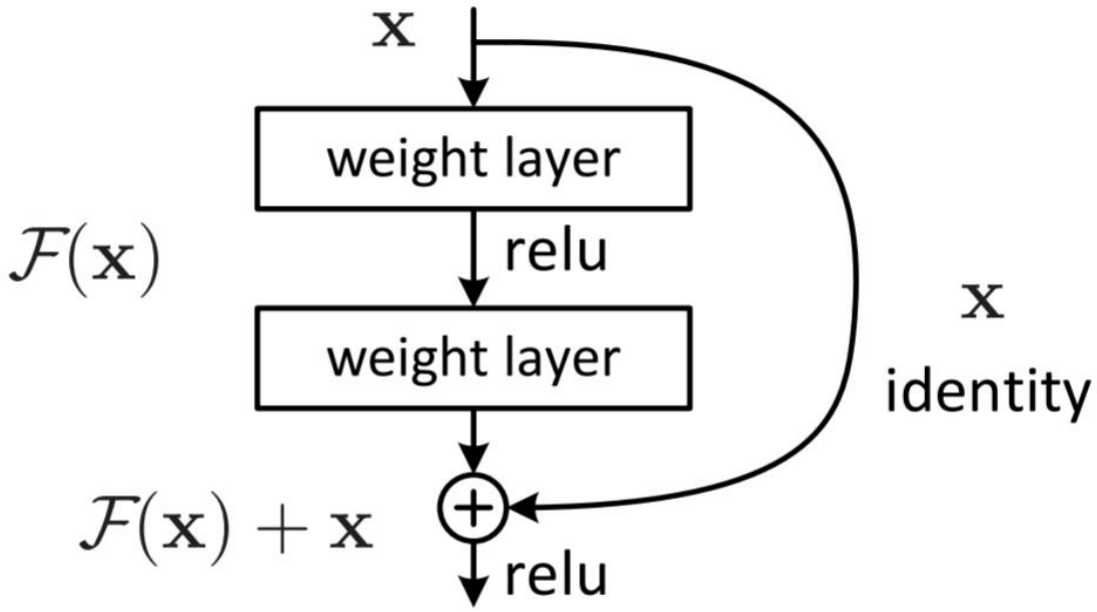
Convolutional neural networks

Convolutions demo

Taken from Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition

Network topology

Not only a simple sequence of layers.



He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Regularization

- "Classic" regularization: use L1 or L2 regularization functions.

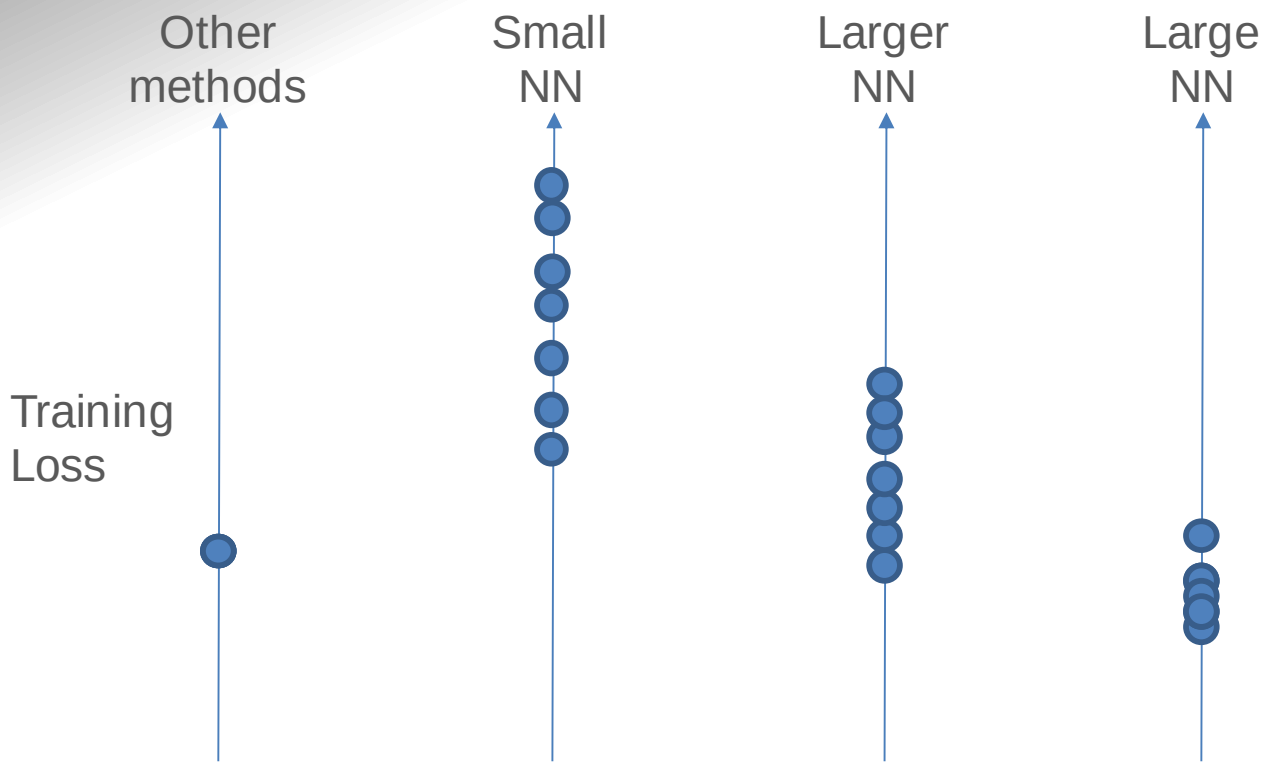
$$\min_{w \in W} C \sum_{i=1...n} l(f(w, x_i), y_i) + r(w)$$

- Dropout regularization: during SGD weight update, set random subset of outputs to 0.

Example layer outputs: (1.3, 2.1, 0.5, 2.3)
Random mask: (1, 0, 0, 1)
Final output: (1.3, 0.0, 0.0, 2.3)

Wager, Stefan, Sida Wang, and Percy S. Liang. "Dropout training as adaptive regularization." *Advances in neural information processing systems*. 2013.

Larger NN are easier to train



Choromanska, Anna, et al. "The loss surfaces of multilayer networks." *Artificial Intelligence and Statistics*. 2015.

Deep Learning Software

Most of it in python, and can run on both GPU or CPU:

- Theano
- TensorFlow
- Keras
- PyTorch
- Torch
- MXNet
- Lasagne
- Autograd
- Chainer
- Deeplearn.js
- …

Advantages

Well suited for big data

- Is not as computationally expensive as Kernel SVM or KNN for large amounts of data.

Unreasonably flexible

- Can take as input directly images, text, structured data etc. Can output directly images, text, structured data etc.

State of the art

- Recently dedicated neural networks beat other methods on applications of computer vision and natural language processing -> great hype right now.

Disadvantages

Not very well understood

- Does not have a strong theoretical background like SVM.

Requires a lot of data

- For neural network to converge properly, large number of parameters are required. For large number of parameters, large amount of data is needed to not overfit.

Black box model

- Interpretation is complicated due to complexity of models.

References

- Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." *Neural networks* 61 (2015): 85-117.
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *Nature*521.7553 (2015): 436-444.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

Cool stuff

Meta learning – learn to x!

Andrychowicz, Marcin, et al. "Learning to learn by gradient descent by gradient descent." *Advances in Neural Information Processing Systems*. 2016.

Li, Ke, and Jitendra Malik. "Learning to optimize." *arXiv preprint arXiv:1606.01885* (2016).

Wang, Jane X., et al. "Learning to reinforcement learn." *arXiv preprint arXiv:1611.05763* (2016).

Wang, Dilin, and Qiang Liu. "Learning to draw samples: With application to amortized mle for generative adversarial learning." *arXiv preprint arXiv:1611.01722* (2016).

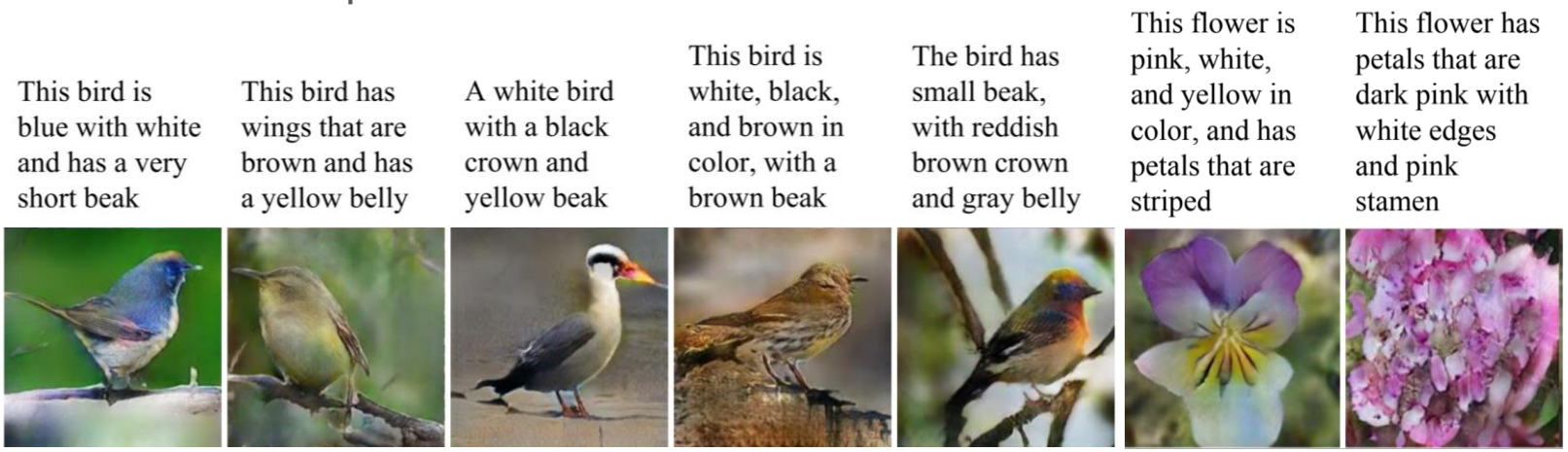
Chen, Yutian, et al. "Learning to Learn without Gradient Descent by Gradient Descent." *arXiv preprint arXiv:1611.03824* (2016).

Cool stuff

Seen so far: deterministic output

Age	Blood pressure	Oldpeak	Sick
70	130	2.4	yes
67	115	1.6	no

GM: Stochastic output



https://arxiv.org/pdf/1612.03242.pdf

Hands on

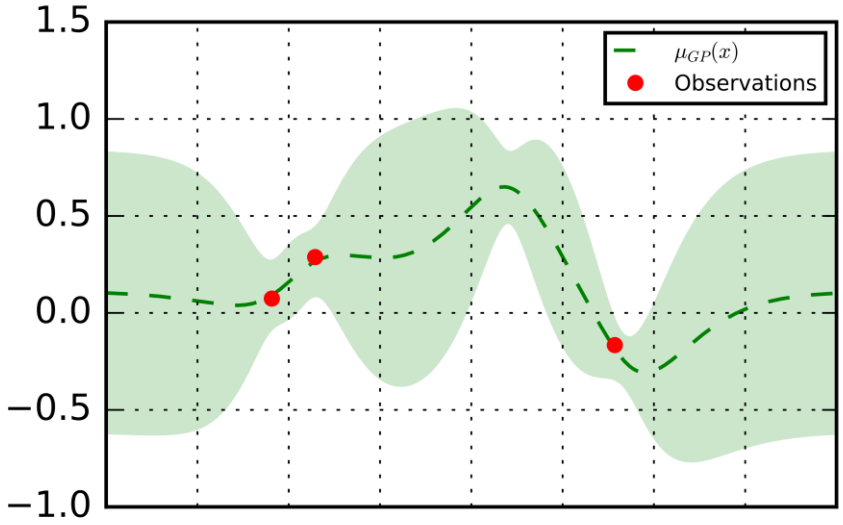
Source: https://github.com/iaroslav-ai/ed3s-2017

Deep Learning: many parameters

- Eg: # of layers, # of neurons, learning rate, momentum, type of neuron activation, batch size, epochs to train, …
- Hard to optimize by hand
- Humans are expensive and biased

Sequential Model Based Optimization

- Global optimality guarantees
- Fast convergence rates
- Support noisy objectives
- No closed form required



Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams. "Practical bayesian optimization of machine learning algorithms." Advances in neural information processing systems. 2012.

Image source: https://github.com/scikit-optimize/scikit-optimize/blob/master/examples/bayesian-optimization.ipynb

We will use this oss library: https://github.com/scikit-optimize/scikit-optimize