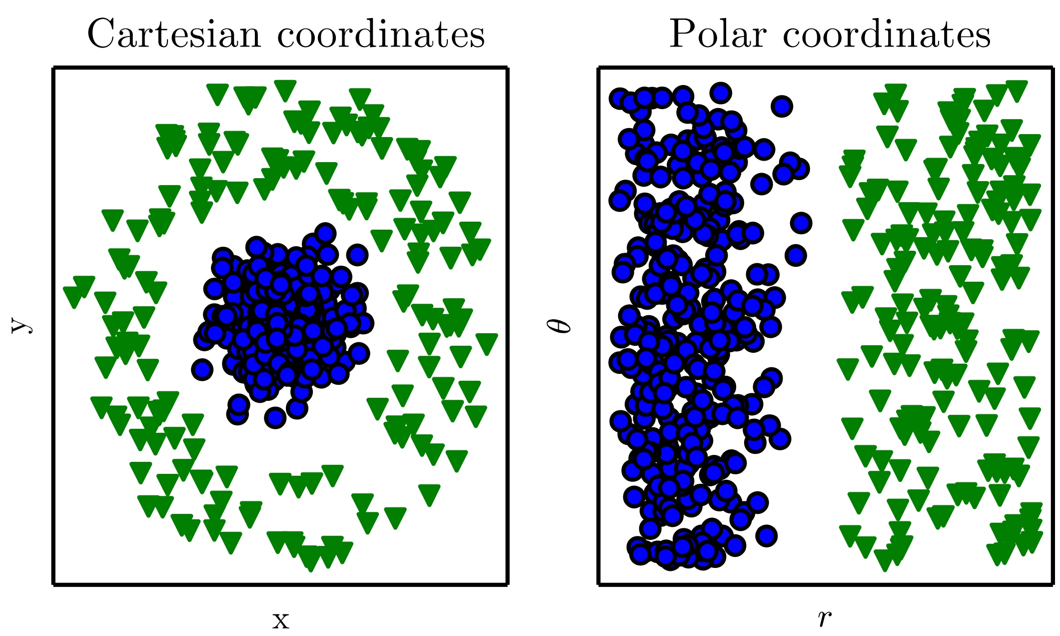
Introduction to Deep Learning
CMPT 733

Steven Bergner

Overview

- Renaissance of artificial neural networks
  - Representation learning vs feature engineering
- Background
  - Linear Algebra, Optimization
  - Regularization
- Construction and training of layered learners
- Frameworks for deep learning

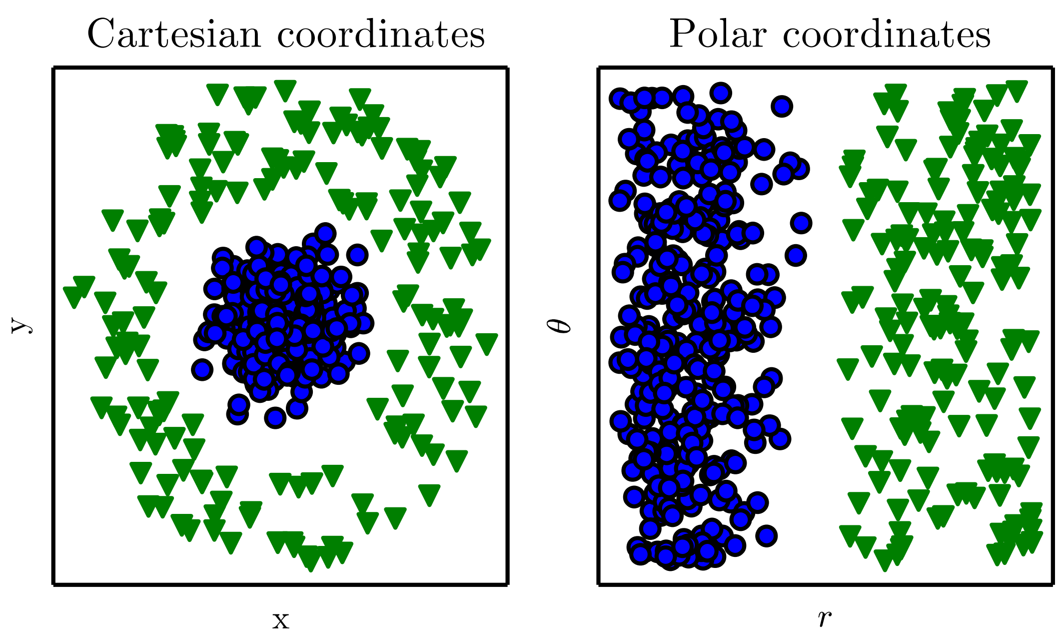
Representations matter



- Transform into the right representation
- Classify points simply by threshold on radius axis

[Goodfellow, Bengio, Courville 2016]

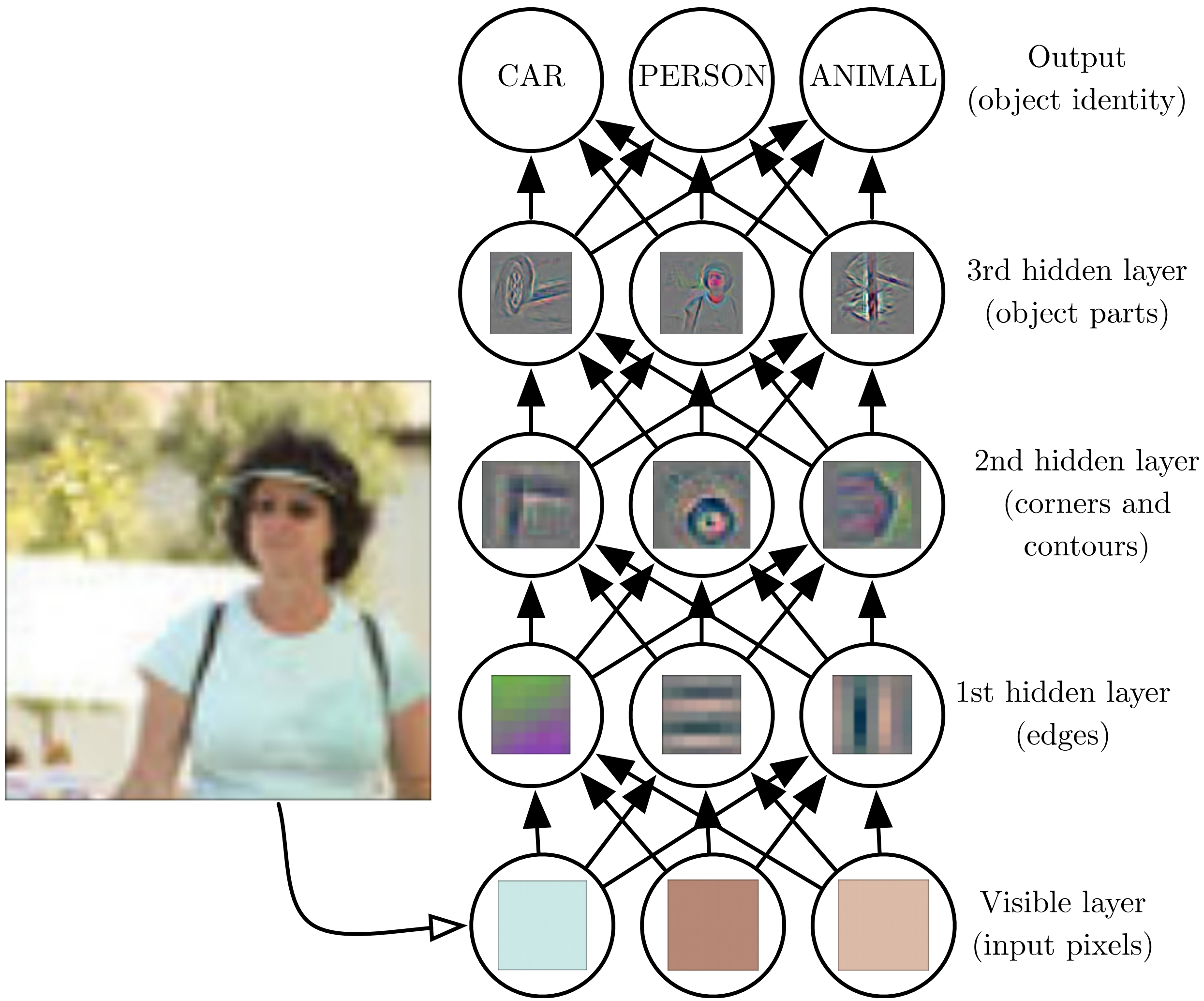
Representations matter



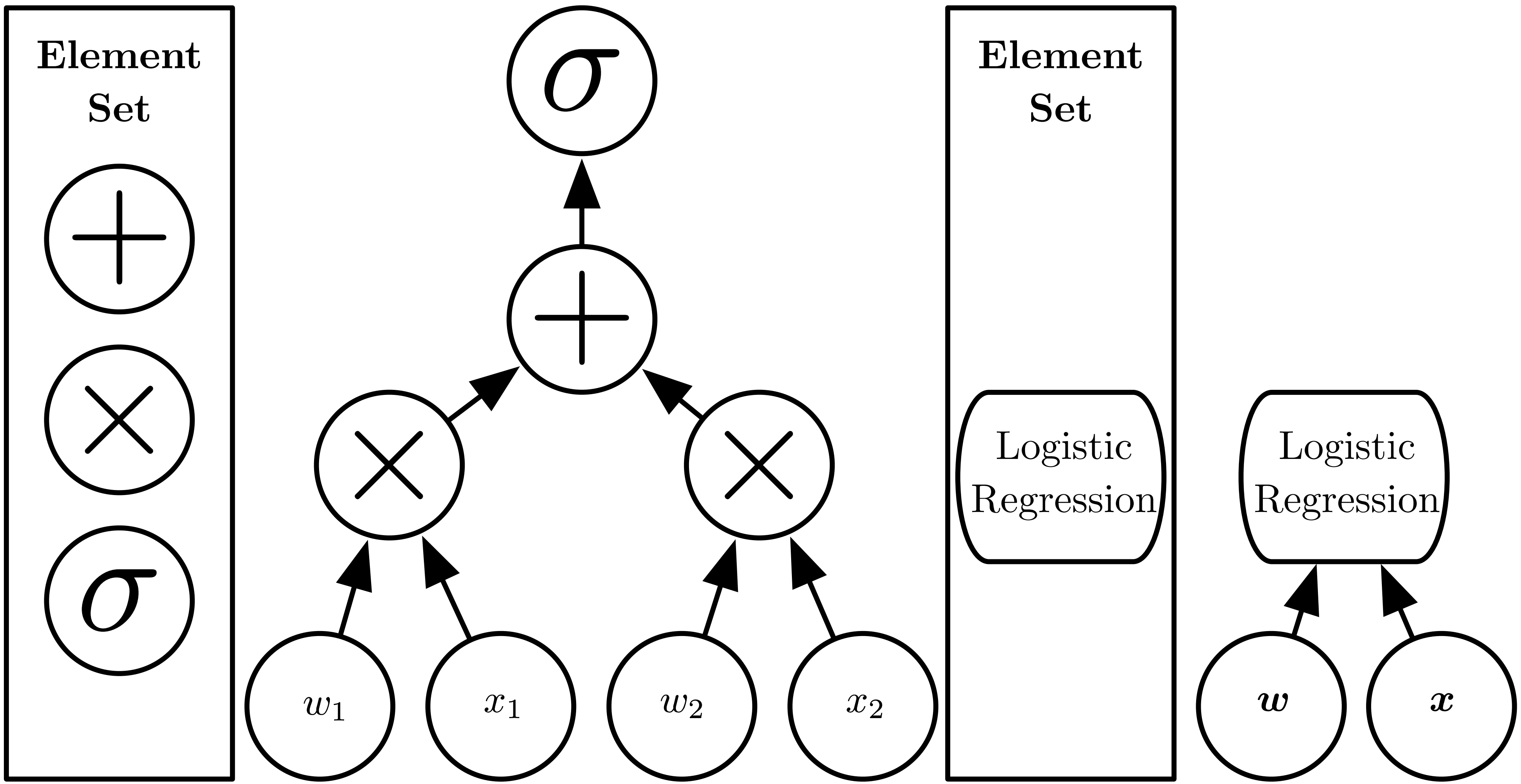
- Transform into the right representation
- Classify points simply by threshold on radius axis
- Single neuron with non-linearity can do this

[Goodfellow, Bengio, Courville 2016]

Depth: layered composition



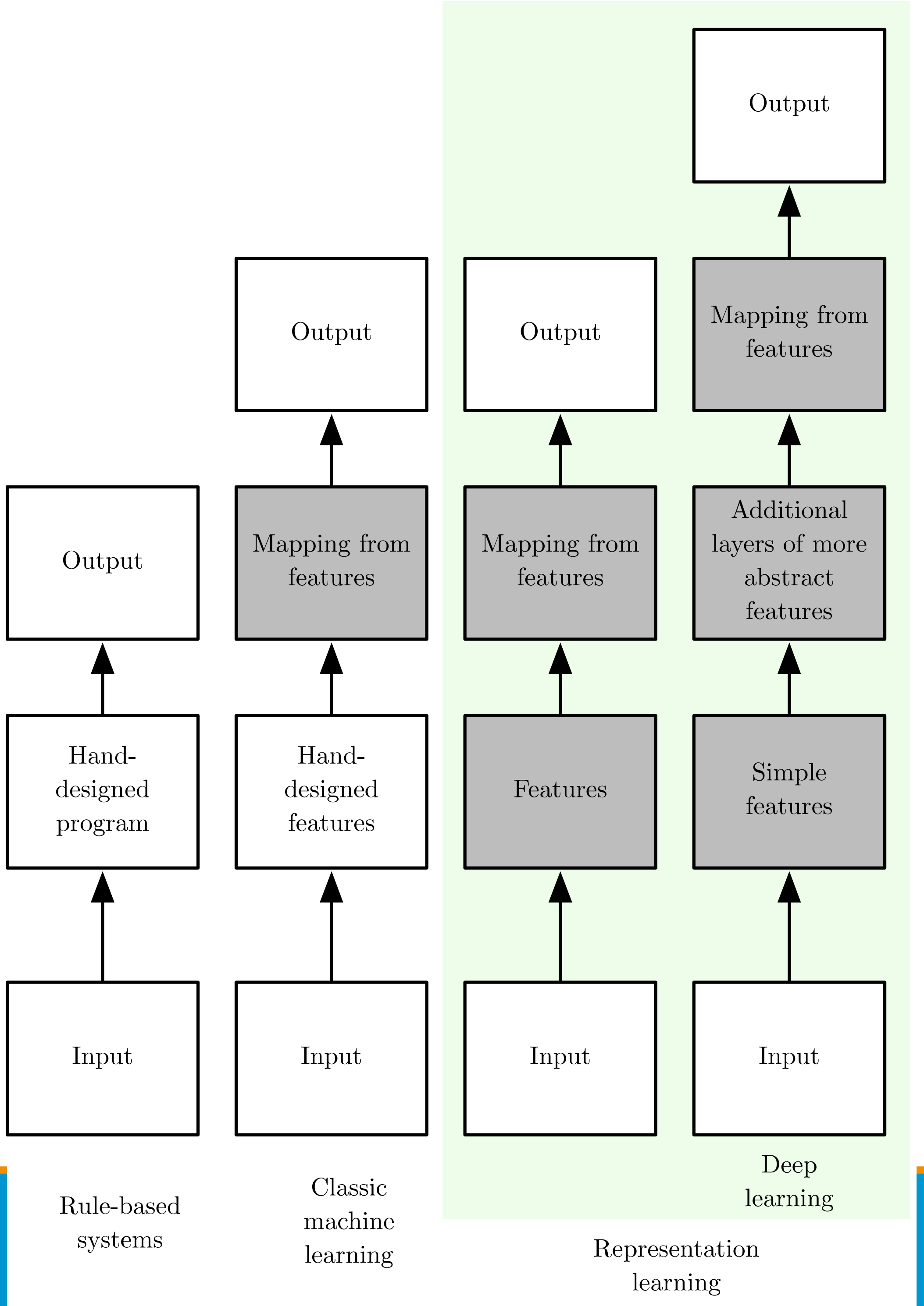
Computational graph



[Goodfellow, Bengio, Courville 2016]

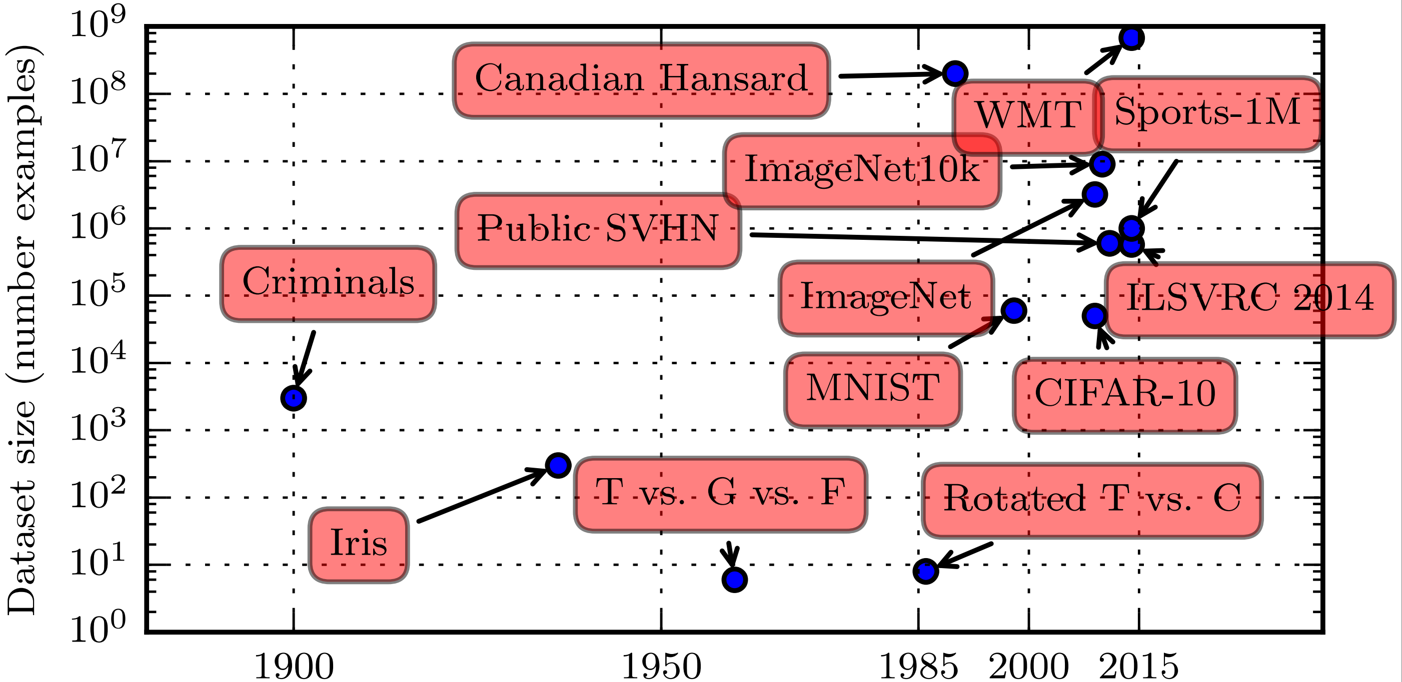
Components of learning

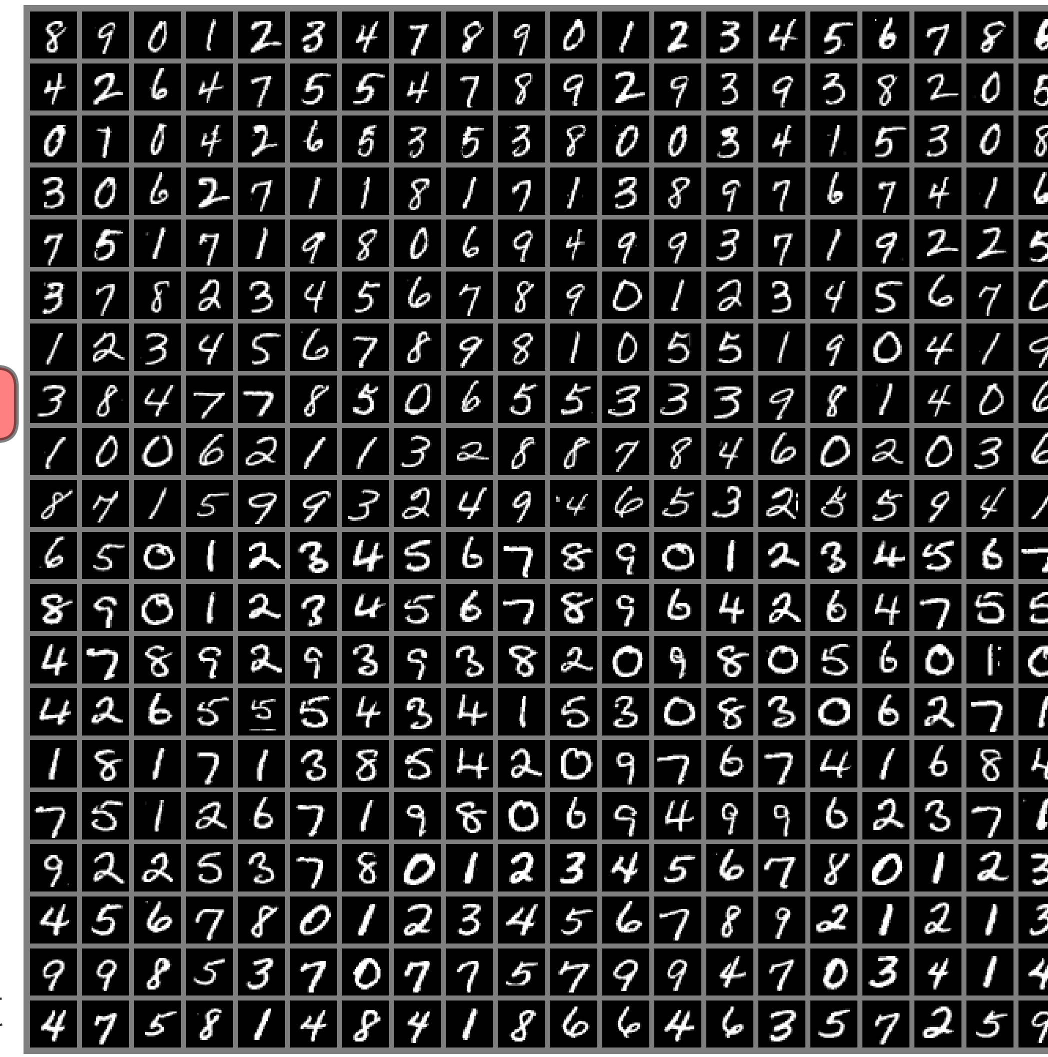
- Hand designed program
  - Input → Output
- Increasingly automated
  - Simple features
  - Abstract features
  - Mapping from features



[Goodfellow, Bengio, Courville 2016]

Growing Dataset Size





MNIST dataset

[Goodfellow, Bengio, Courville 2016]

Basics

Linear Algebra and Optimization

Linear Algebra

- Tensor is an array of numbers
  - Multi-dim: 0d scalar, 1d vector, 2d matrix/image, 3d RGB image
- Matrix (dot) product $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ $C_{i,j} = \sum_k A_{i,k} B_{k,j}$

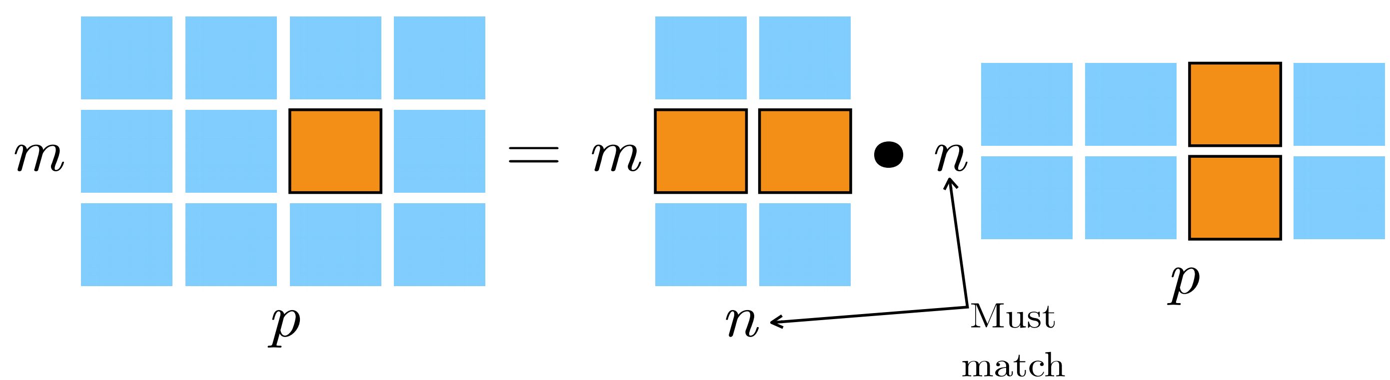
- Dot product of vectors A and B

  (m = p = 1 in above notation, n=2)

[Goodfellow, Bengio, Courville 2016]

Linear Algebra

- Tensor is an array of numbers
  - Multi-dim: 0d scalar, 1d vector, 2d matrix/image, 3d RGB image
- Matrix (dot) product $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ $C_{i,j} = \sum_k A_{i,k} B_{k,j}$



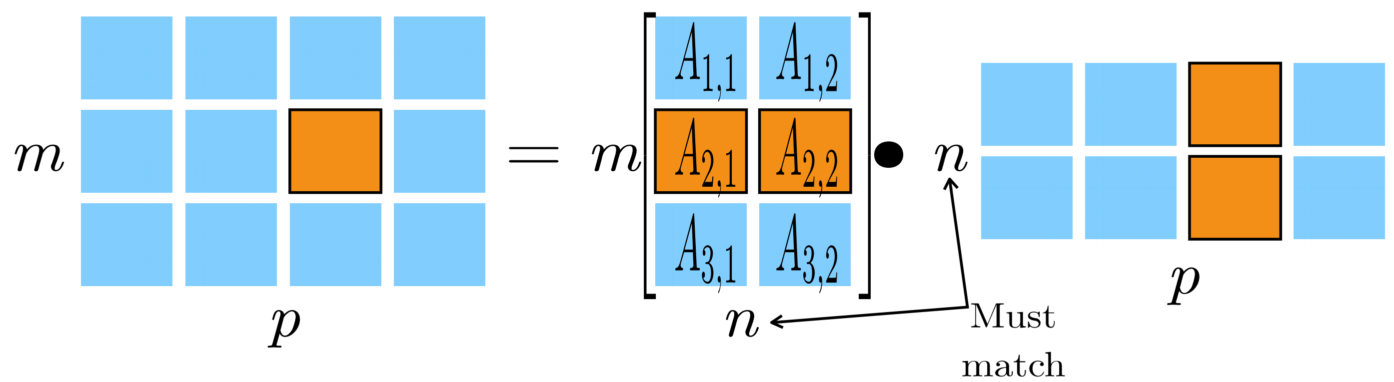
- Dot product of vectors A and B

  (m = p = 1 in above notation, n=2)

[Goodfellow, Bengio, Courville 2016]

Linear Algebra

- Tensor is an array of numbers
  - Multi-dim: 0d scalar, 1d vector, 2d matrix/image, 3d RGB image
- Matrix (dot) product $\boldsymbol{C} = \boldsymbol{AB}$ $\quad C_{i,j} = \sum_k A_{i,k} B_{k,j}$



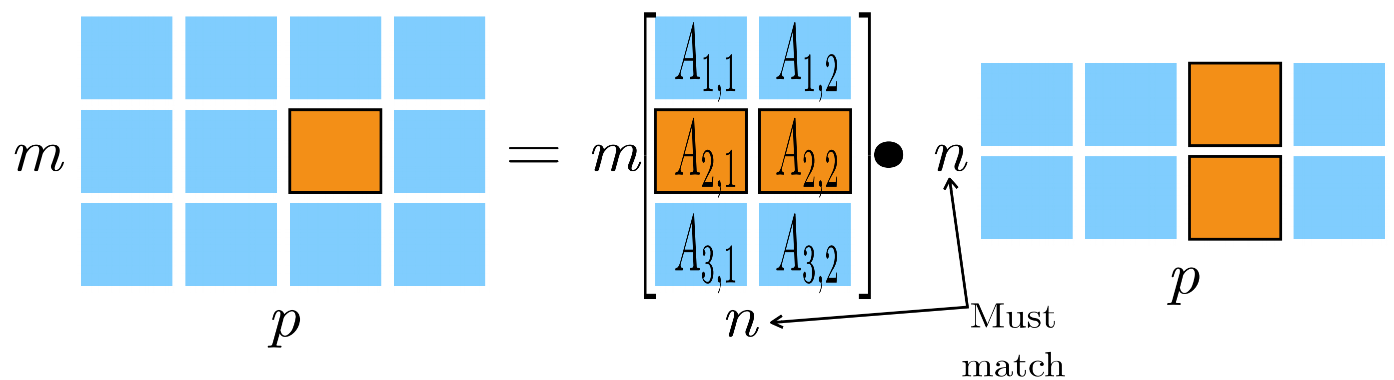
- Dot product of vectors A and B

  (m = p = 1 in above notation, n=2)

[Goodfellow, Bengio, Courville 2016]

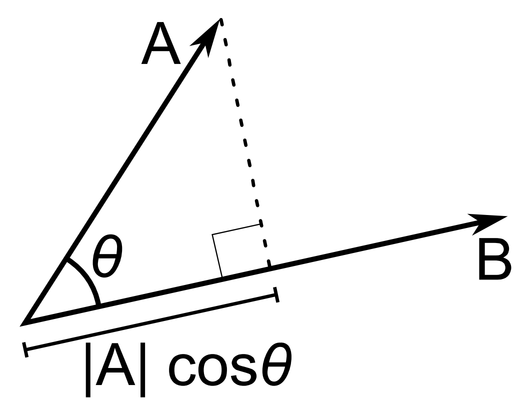
Linear Algebra

- Tensor is an array of numbers
  - Multi-dim: 0d scalar, 1d vector, 2d matrix/image, 3d RGB image
- Matrix (dot) product $\boldsymbol{C} = \boldsymbol{AB}$ $C_{i,j} = \sum_k A_{i,k} B_{k,j}$



- Dot product of vectors A and B

  (m = p = 1 in above notation, n=2)



Linear algebra: Norms

- L^p norm

$$||\boldsymbol{x}||_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

- Most popular norm: L2 norm, p=2

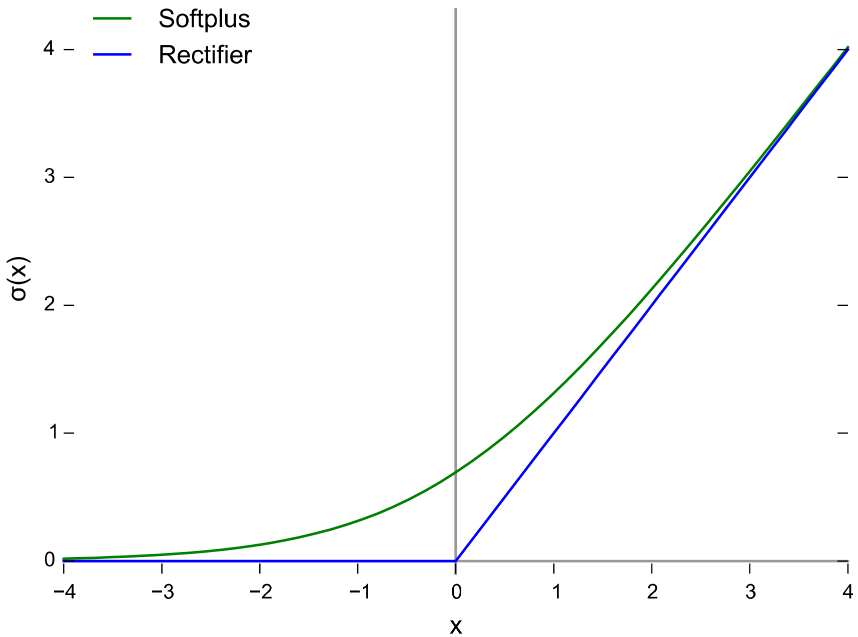
- L1 norm, p=1: $||\boldsymbol{x}||_1 = \sum_i |x_i|.$

- Max norm, infinite p: $||\boldsymbol{x}||_\infty = \max_i |x_i|.$

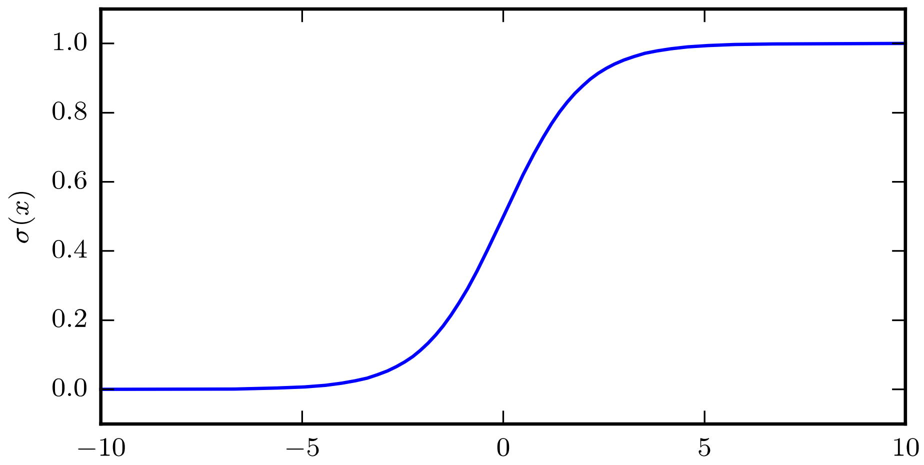
[Goodfellow, Bengio, Courville 2016]

Nonlinearities

- ReLU
- Softplus
- Logistic Sigmoid

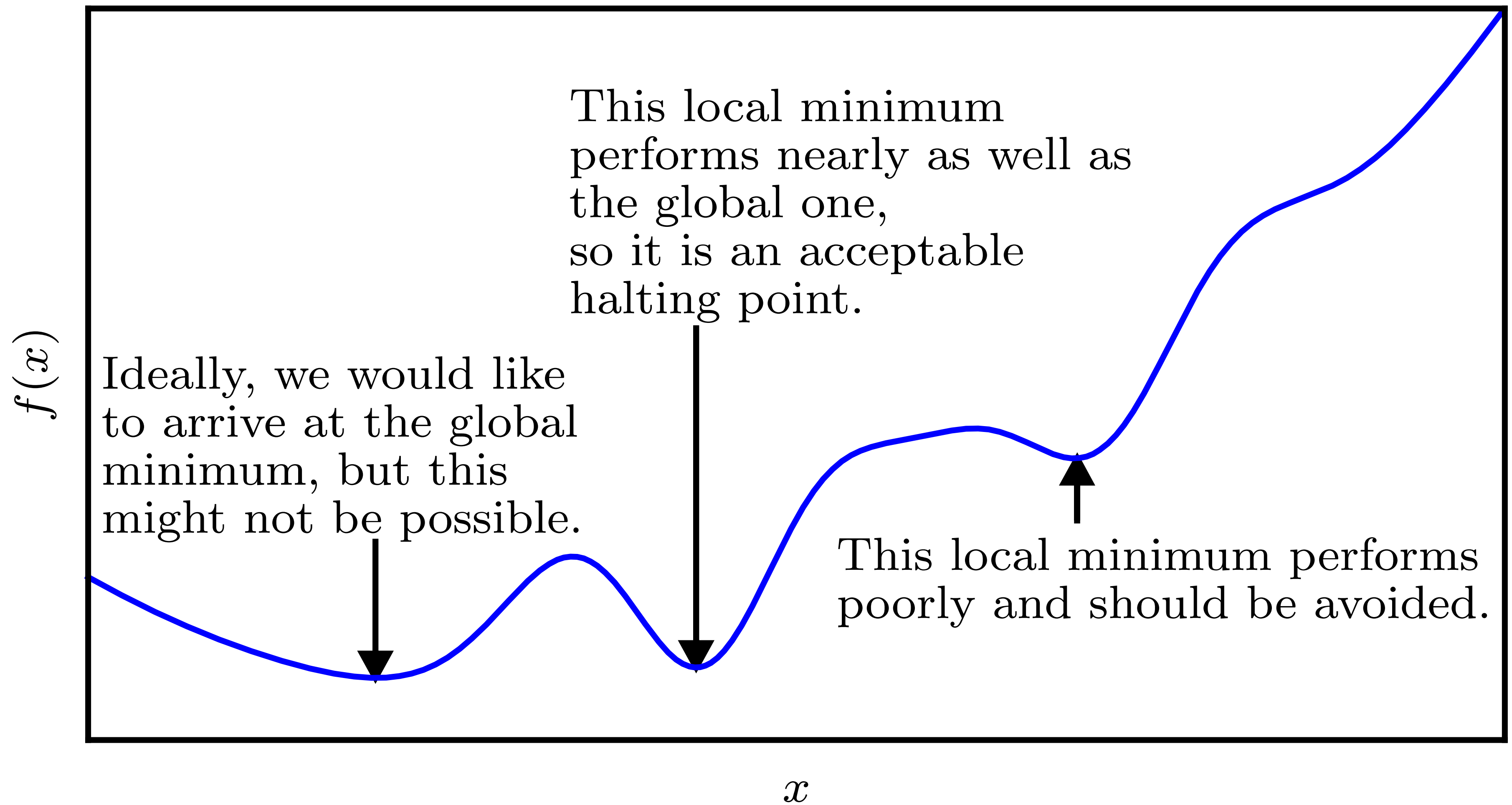


[(c) public domain]



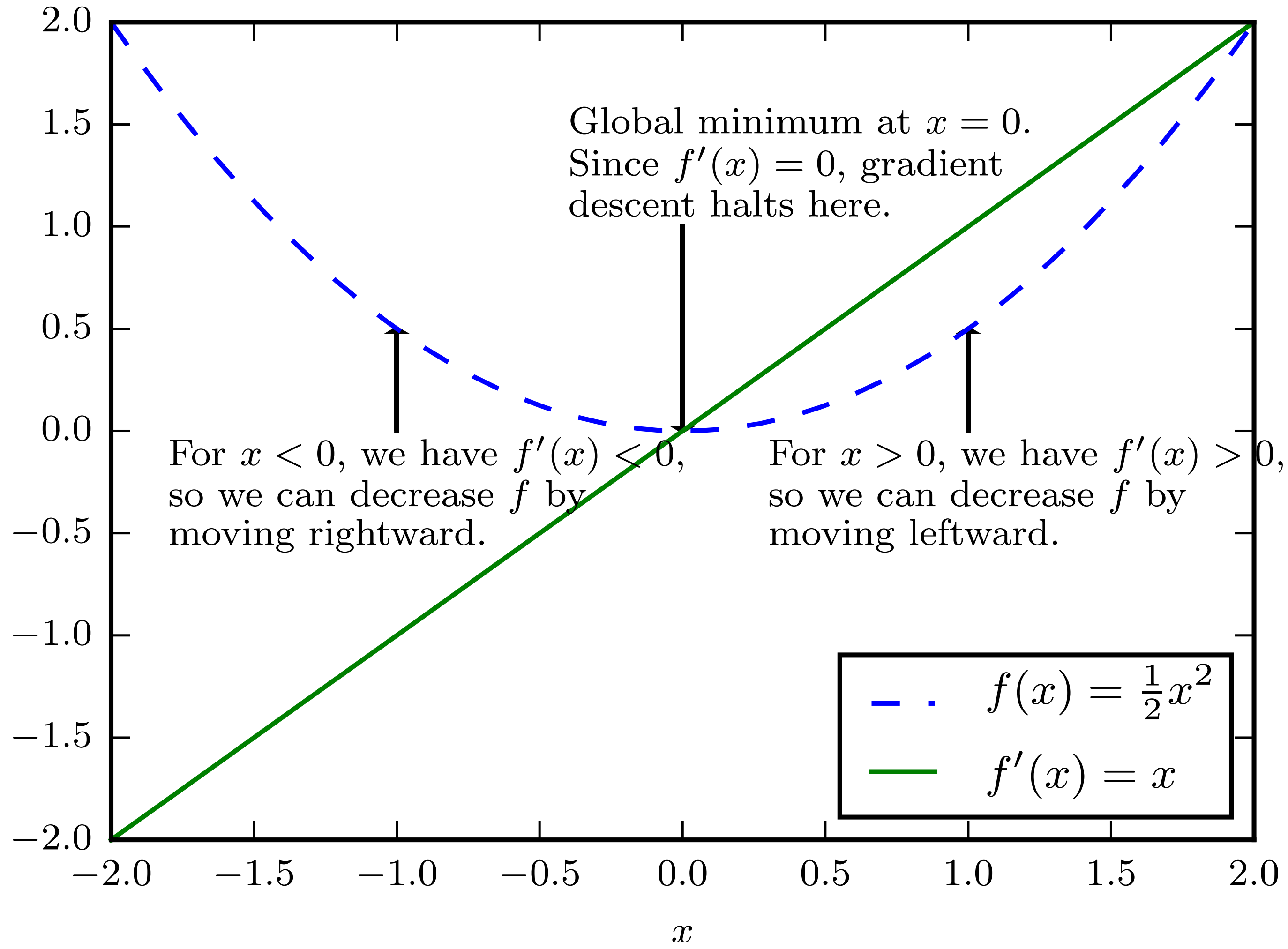
[Goodfellow, Bengio, Courville 2016]

Approximate Optimization



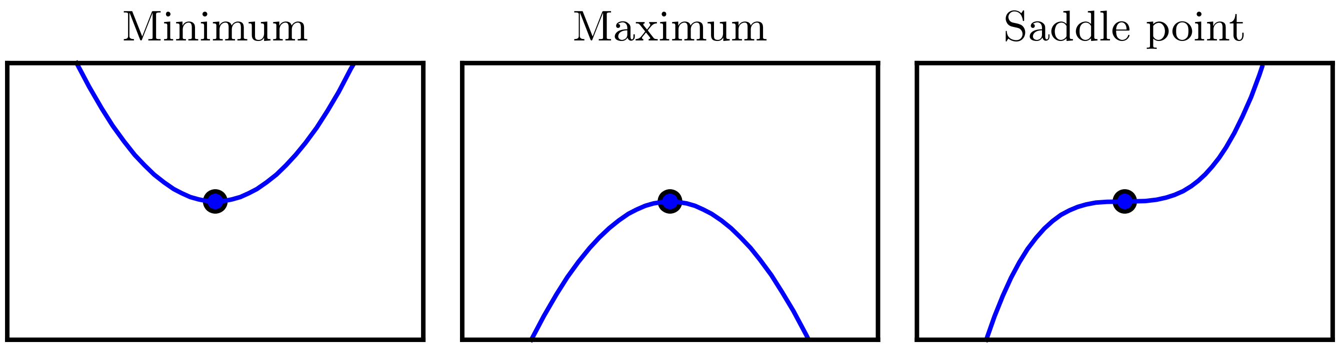
[Goodfellow, Bengio, Courville 2016]

Gradient descent



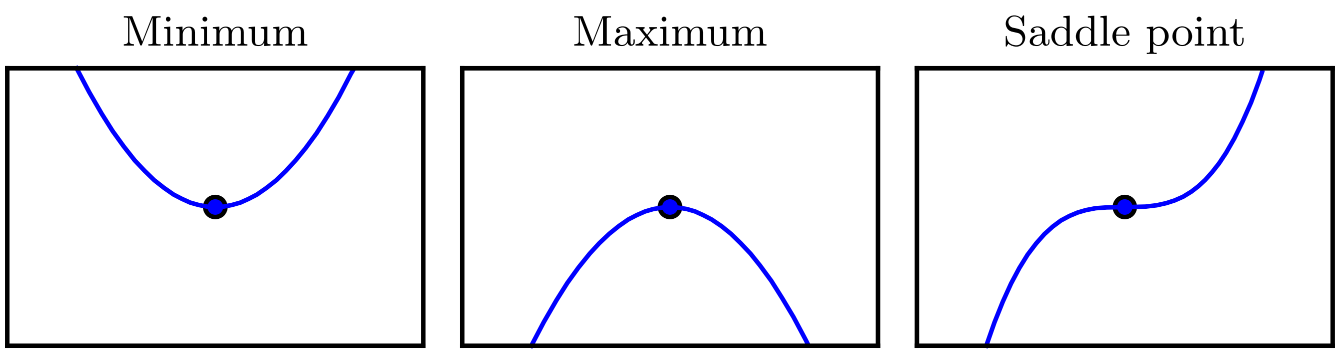
[Goodfellow, Bengio, Courville 2016]

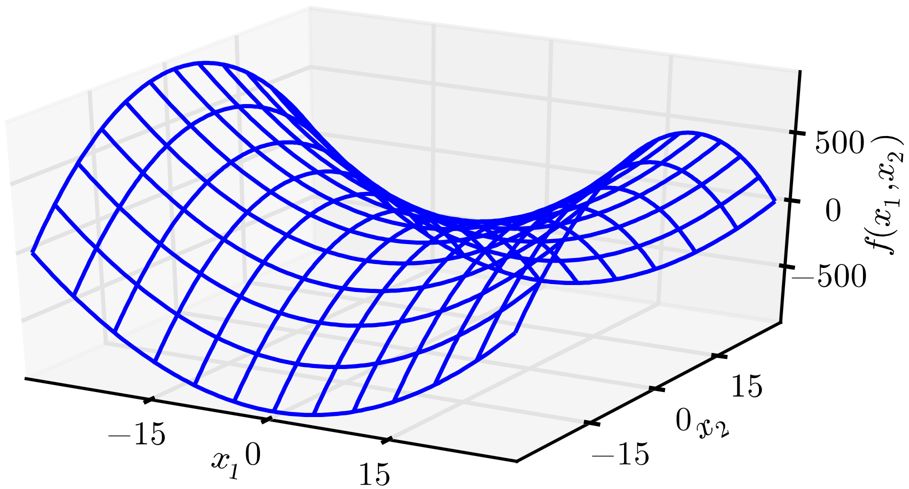
Critical points



[Goodfellow, Bengio, Courville 2016]

Critical points

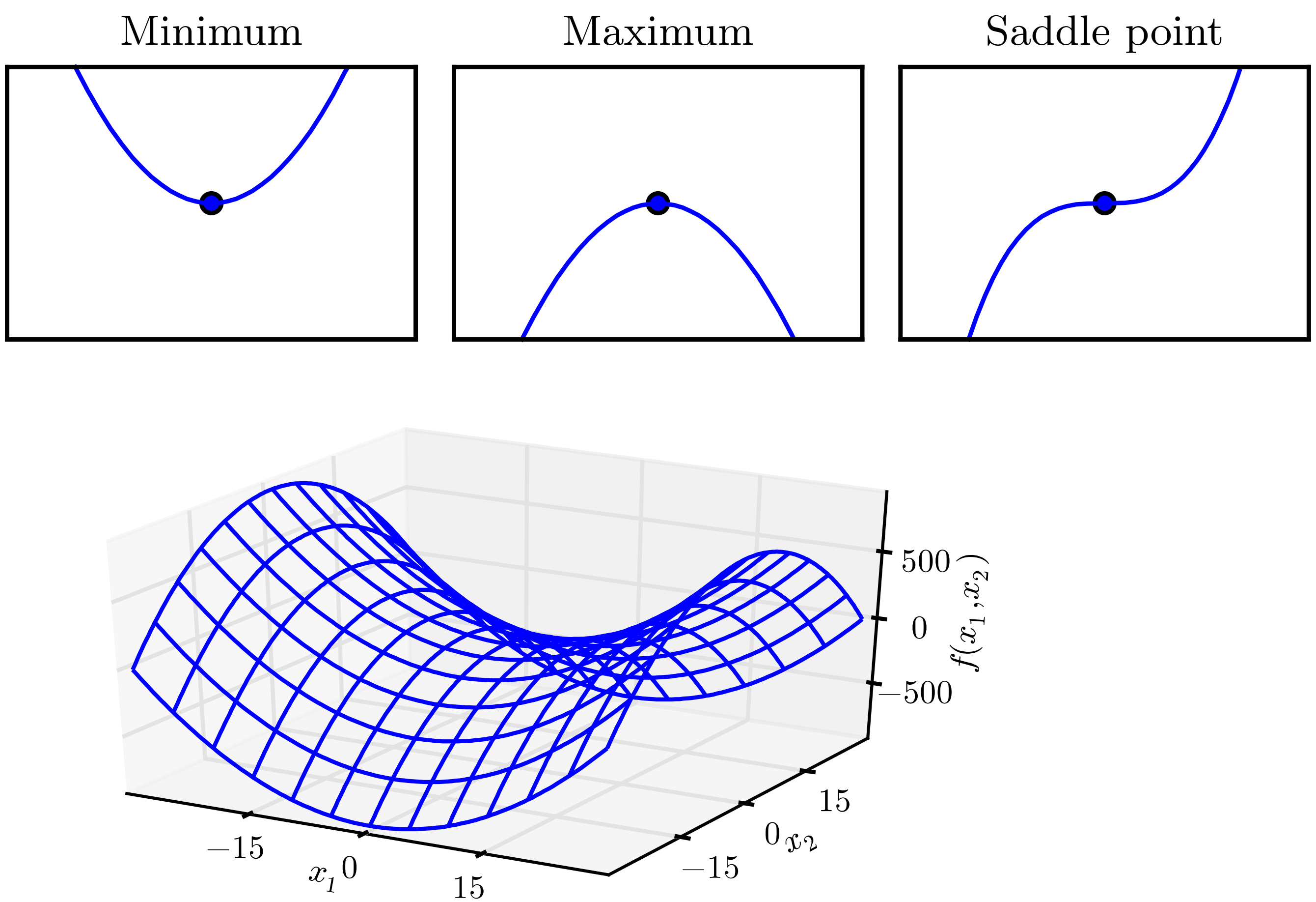




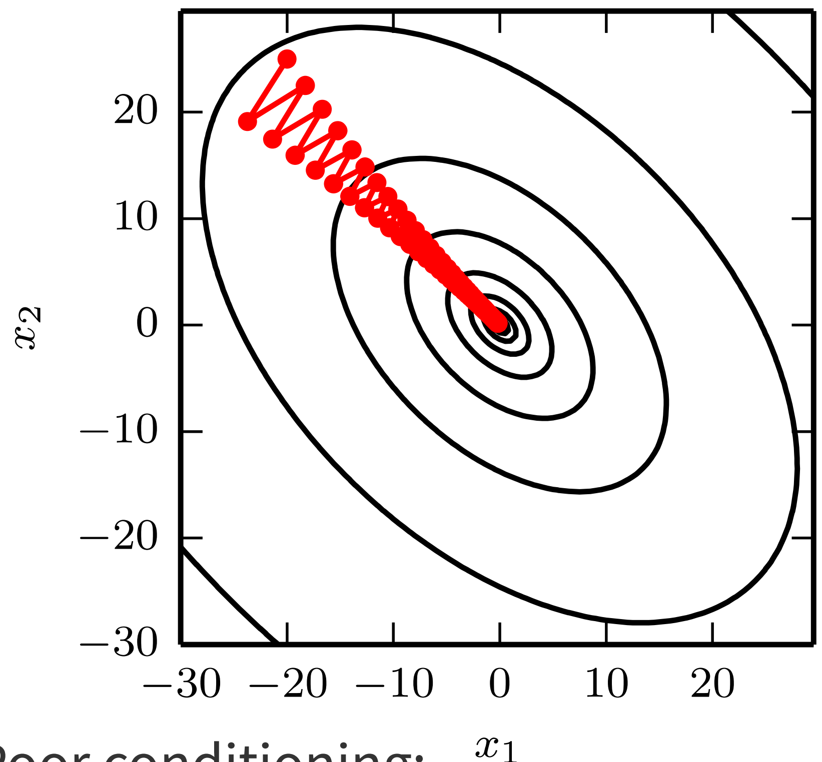
Saddle point – 1st and 2nd derivative vanish

[Goodfellow, Bengio, Courville 2016]

Critical points



Saddle point – 1st and 2nd derivative vanish



Poor conditioning:
1st deriv large in one and small in another direction

Tensorflow Playground

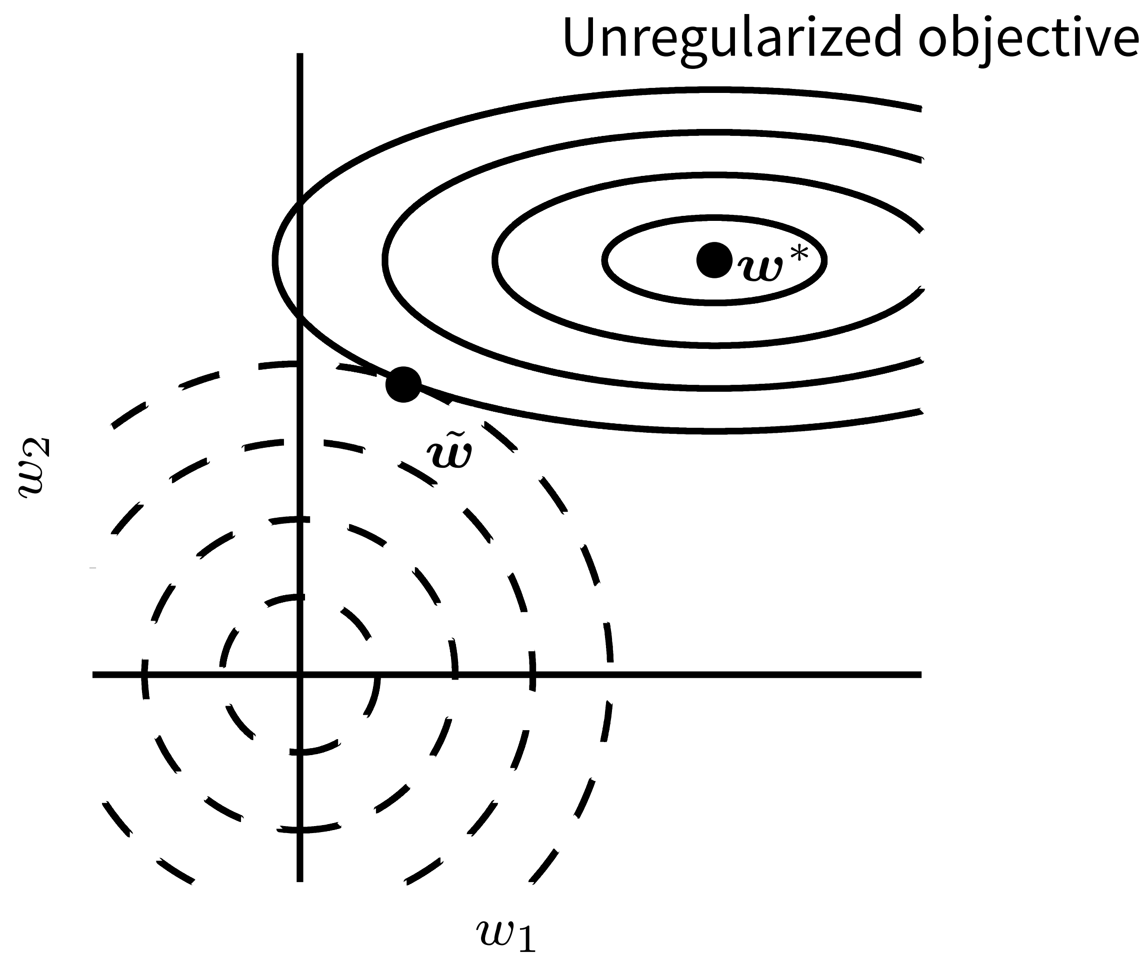
- http://playground.tensorflow.org/
  - Try out simple network configurations

- https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html
  - Visualize linear and non-linear mappings

Regularization

Reduced generalization error without impacting training error

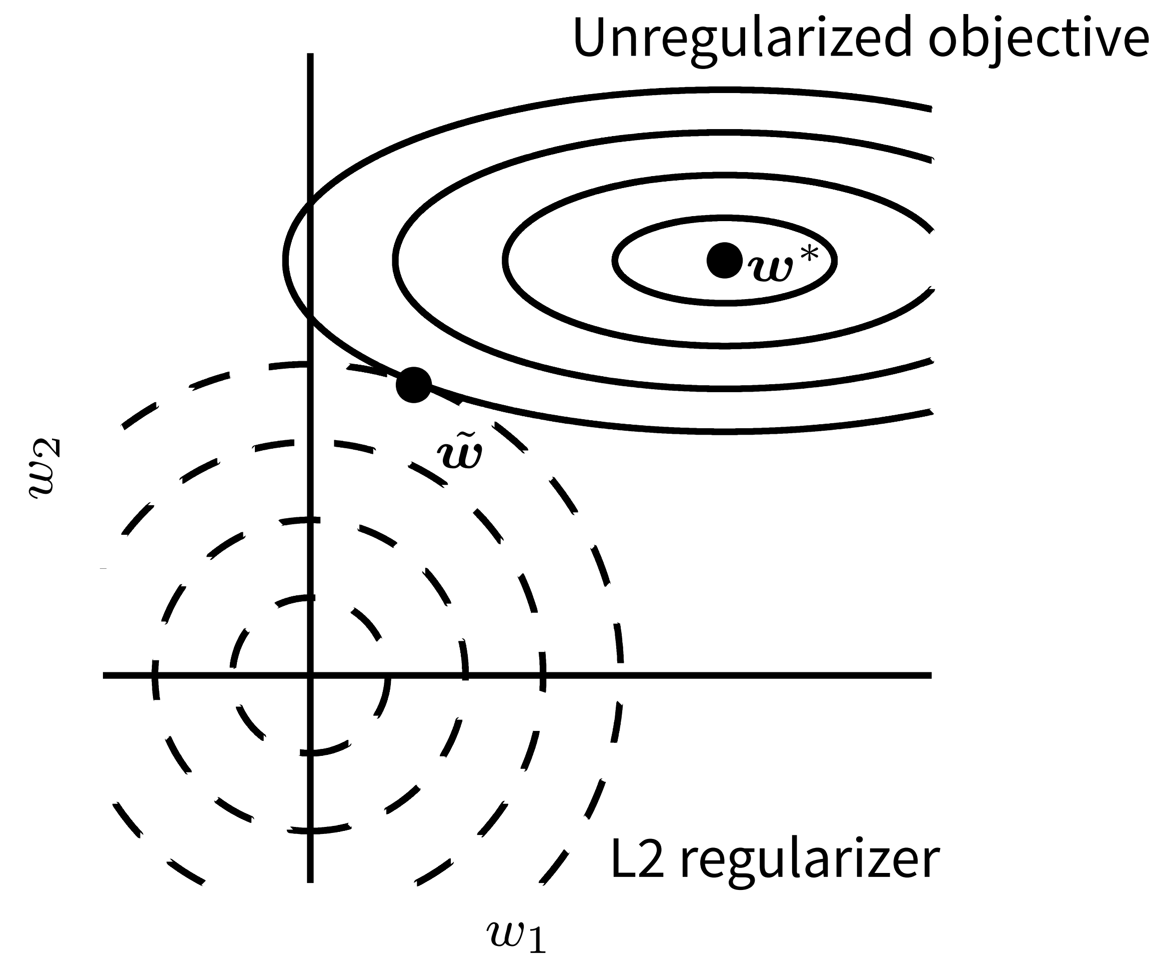
Constrained optimization



[Goodfellow, Bengio, Courville 2016]

Constrained optimization

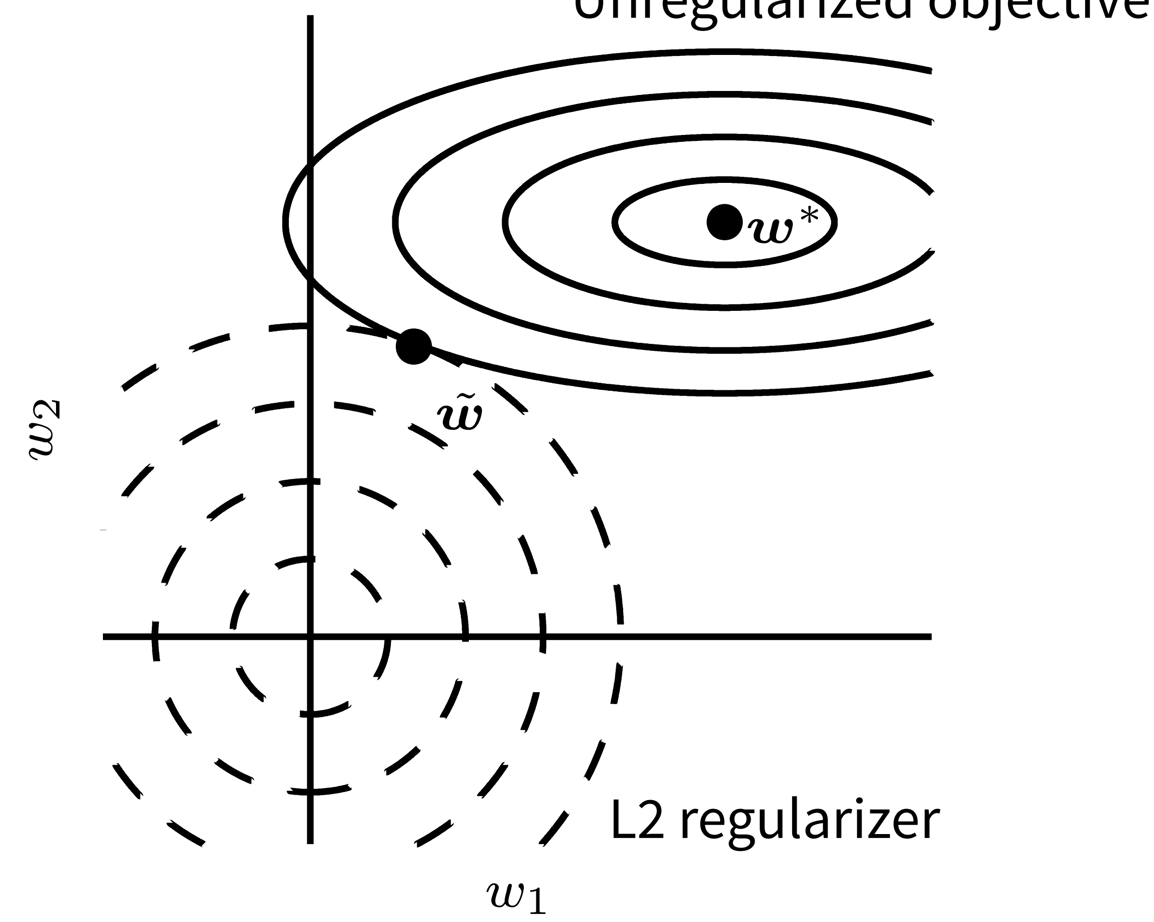
- Squared L2 encourages small weights



[Goodfellow, Bengio, Courville 2016]

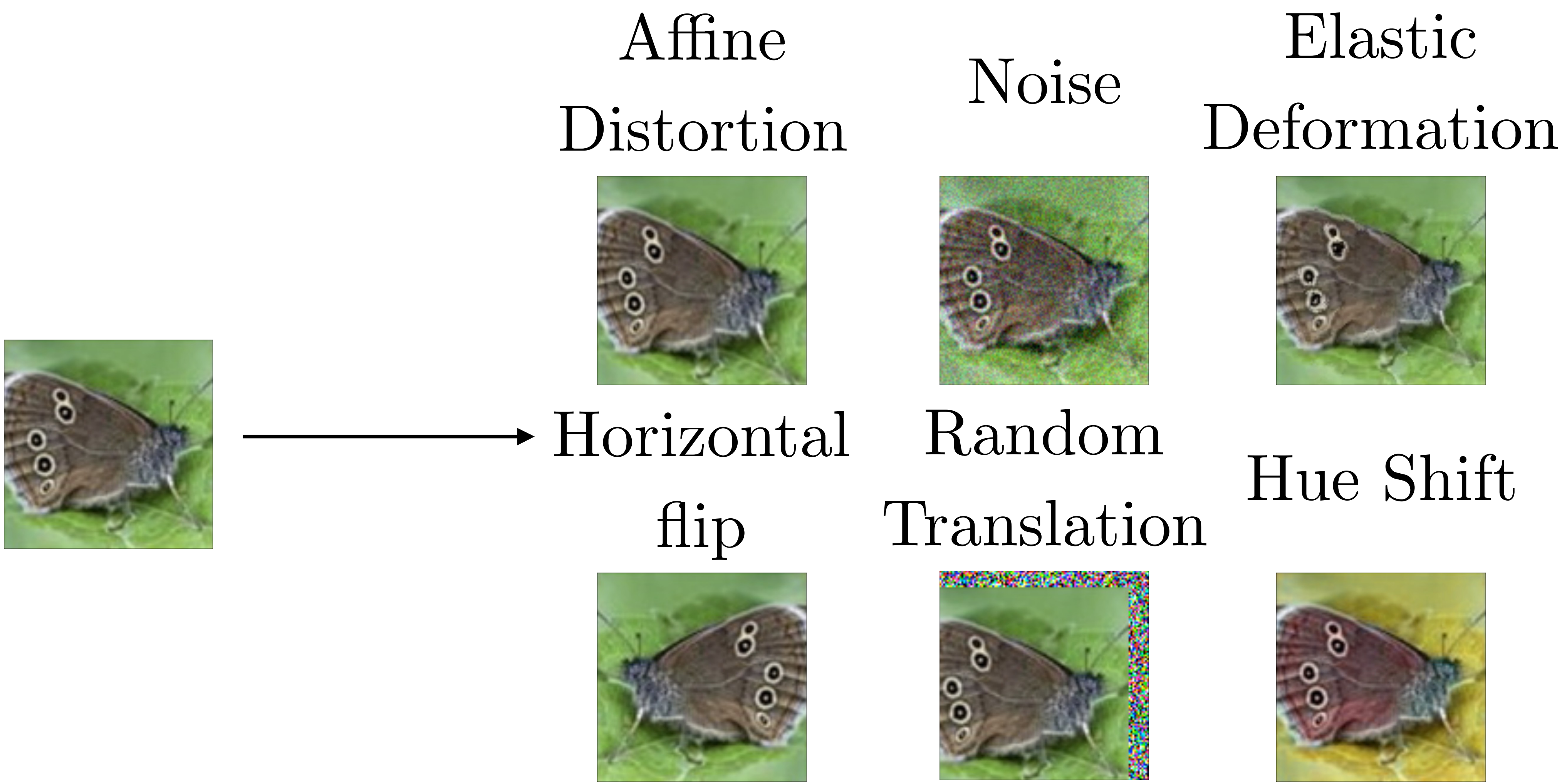
Constrained optimization

- Squared L2 encourages small weights
- L1 encourages sparsity of model parameters (weights)



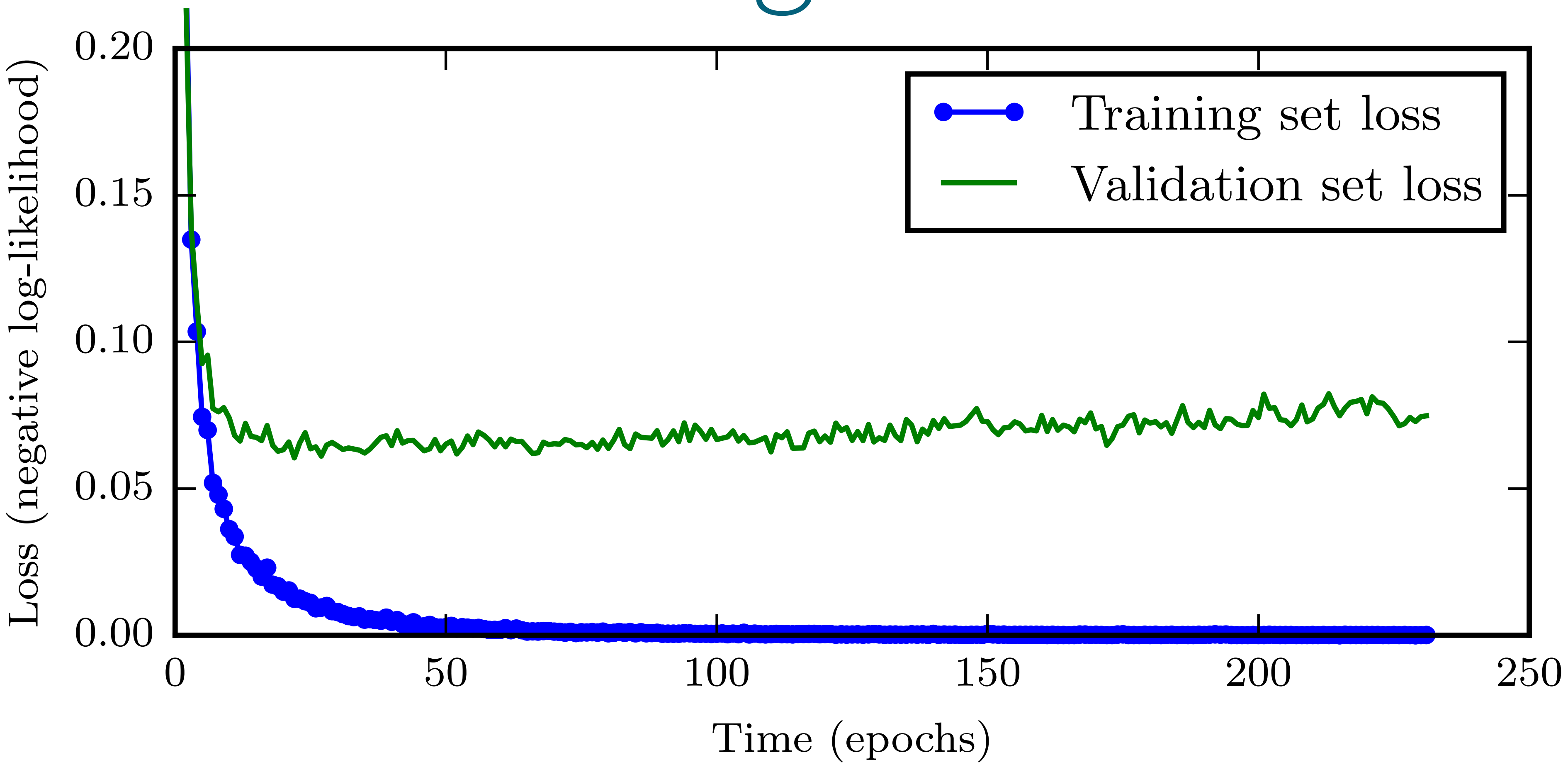
[Goodfellow, Bengio, Courville 2016]

Dataset augmentation

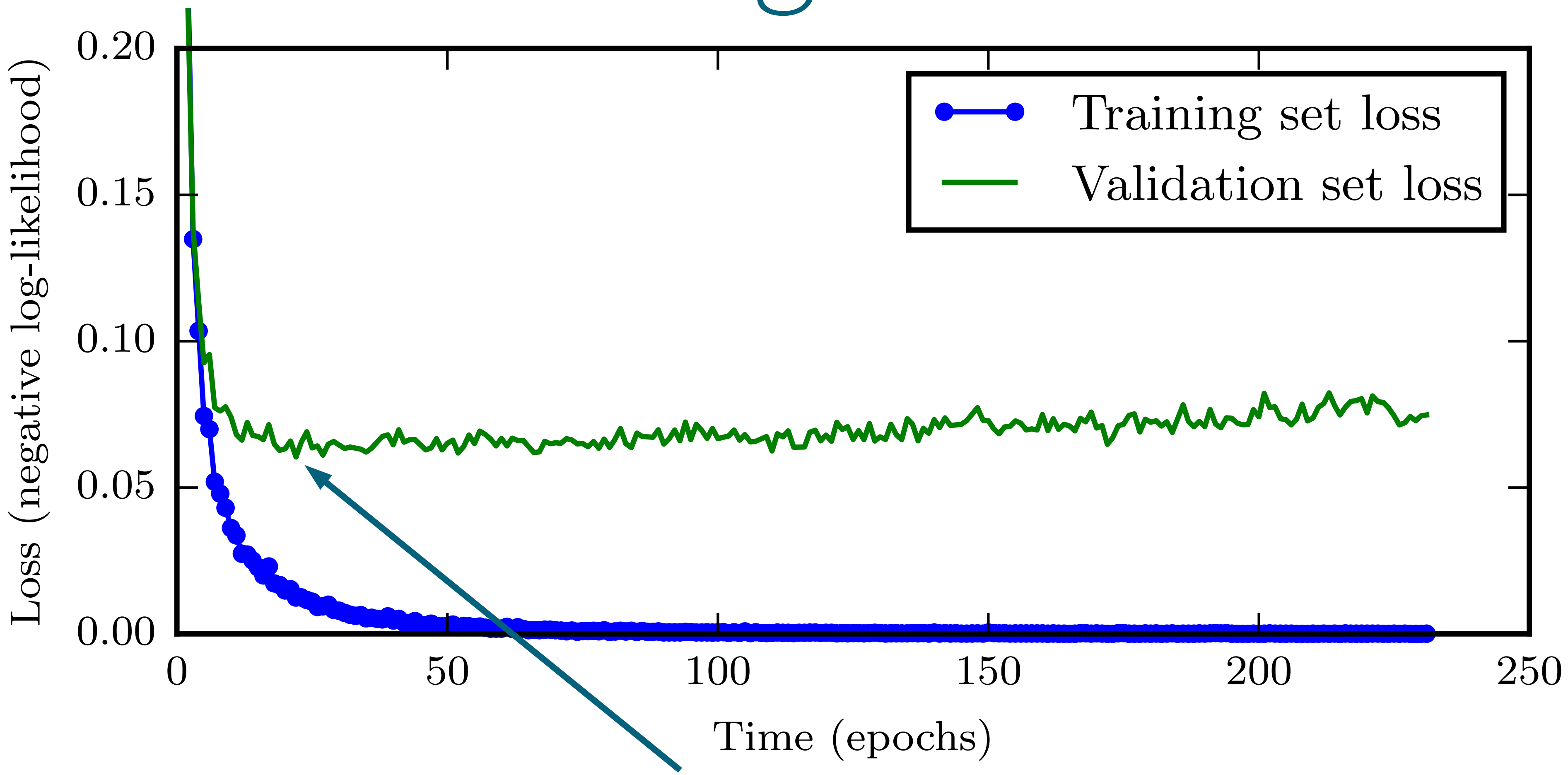


[Goodfellow, Bengio, Courville 2016]

Learning curves



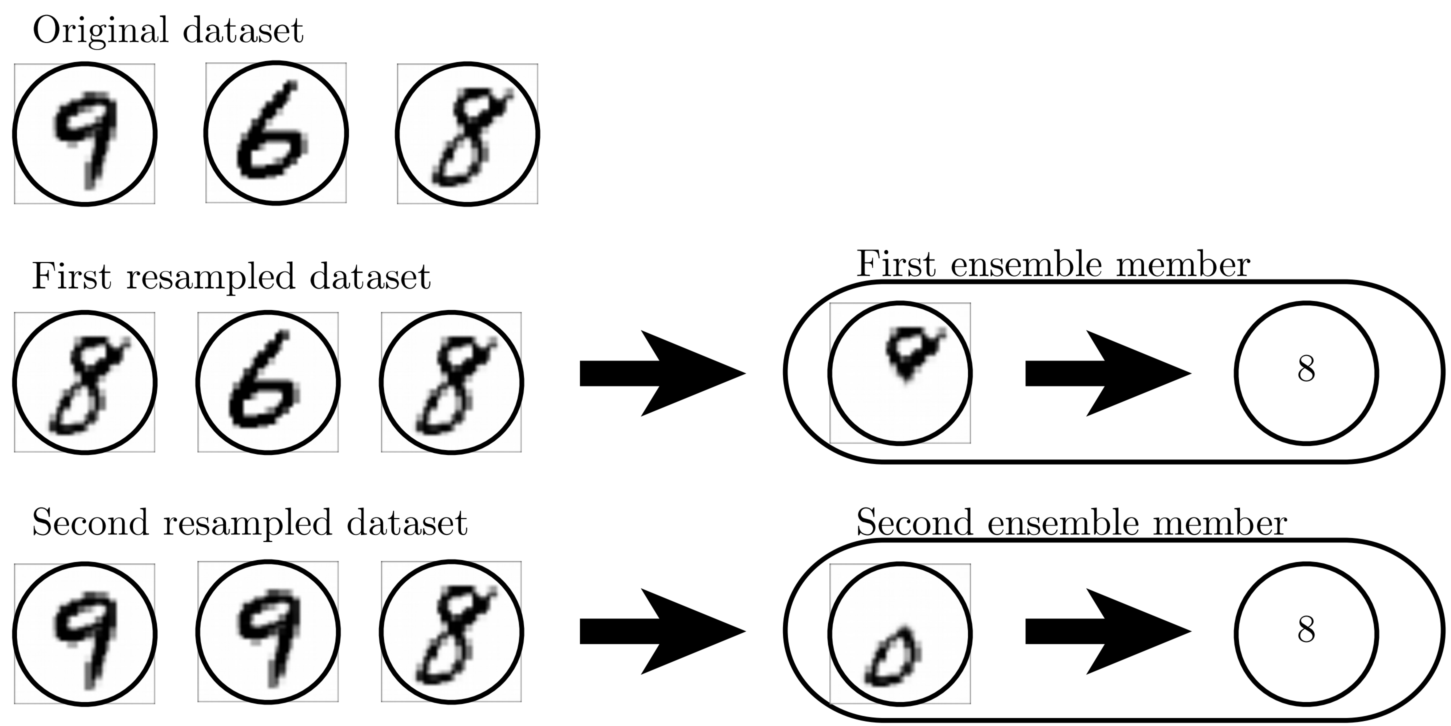
Learning curves



- Early stopping before validation error starts to increase

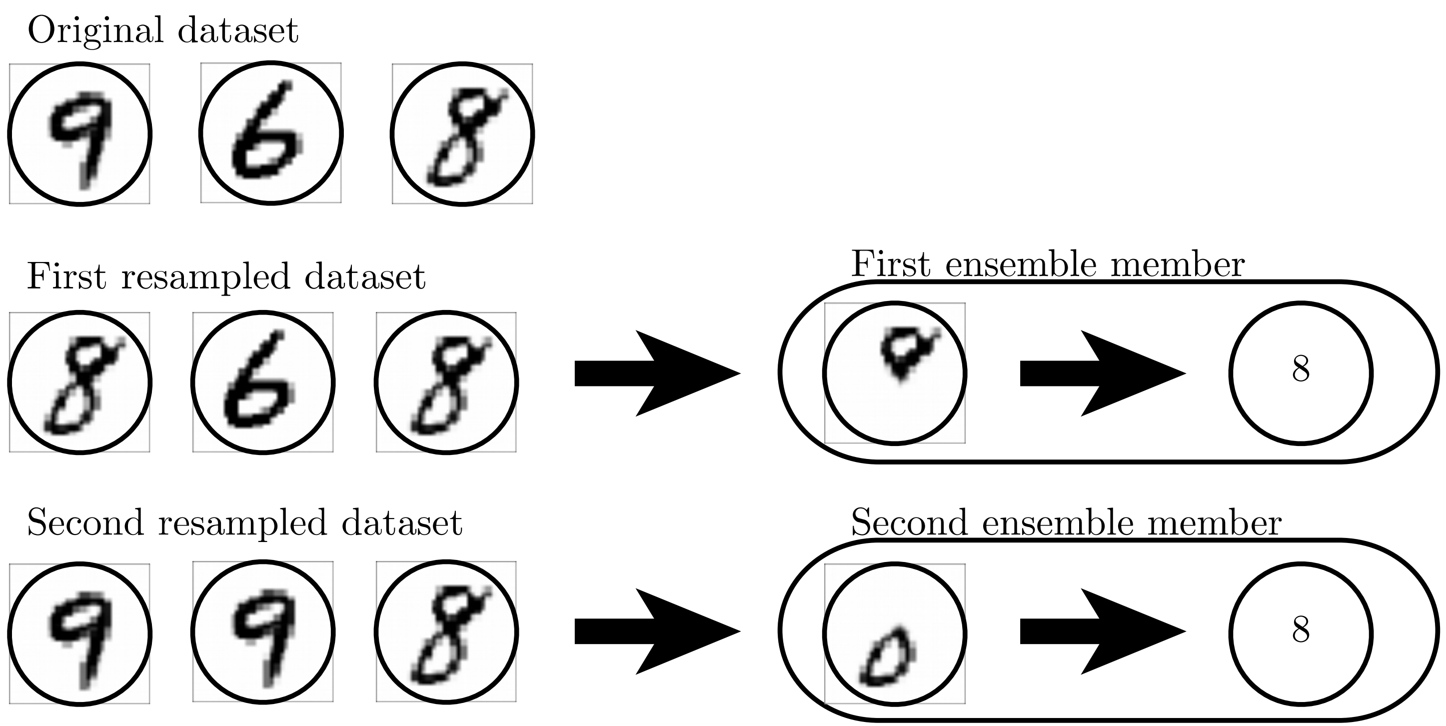
Bagging

- Average multiple models trained on subsets of the data



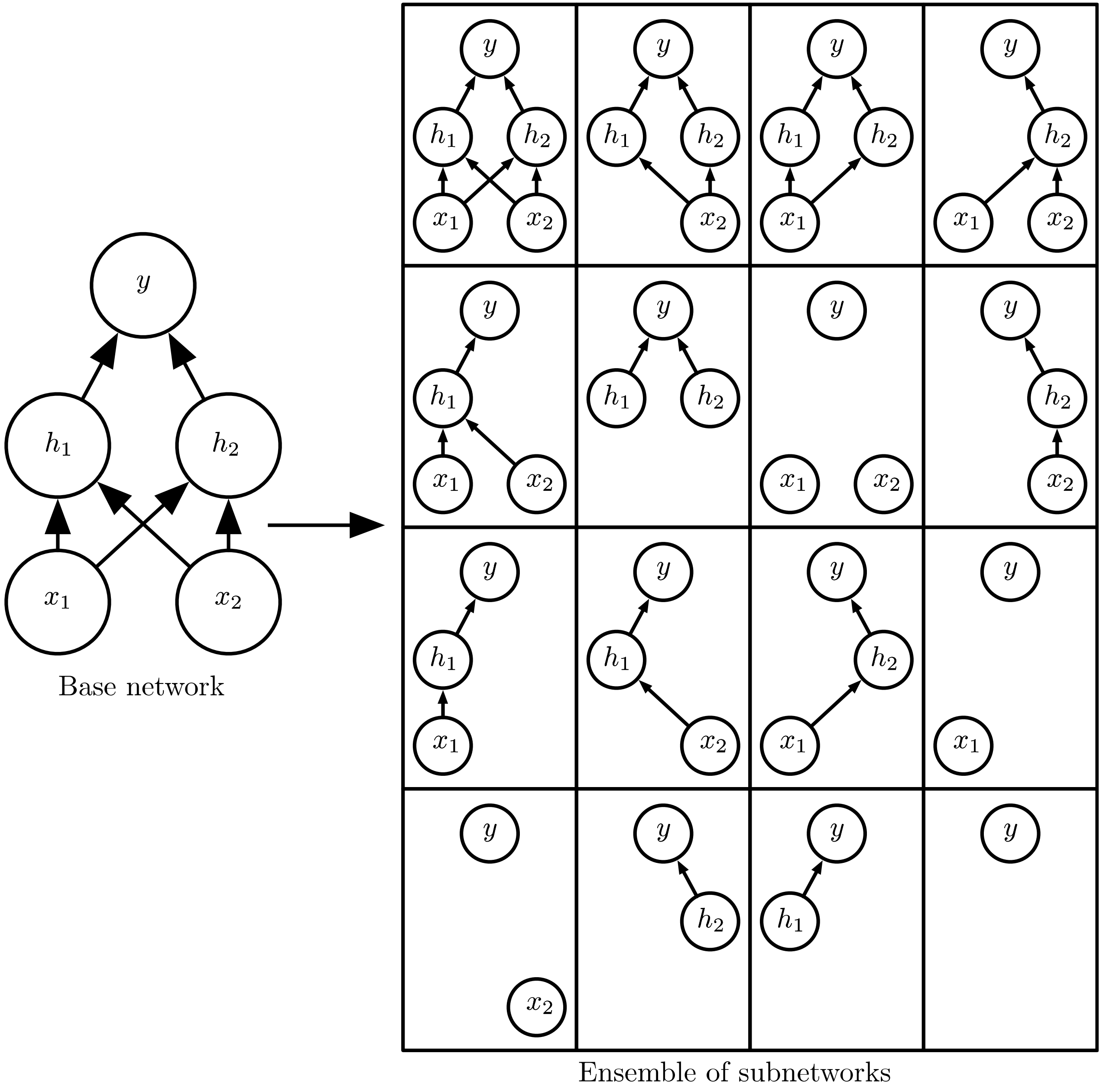
Bagging

- Average multiple models trained on subsets of the data
- First subset: learns top loop, Second subset: bottom loop



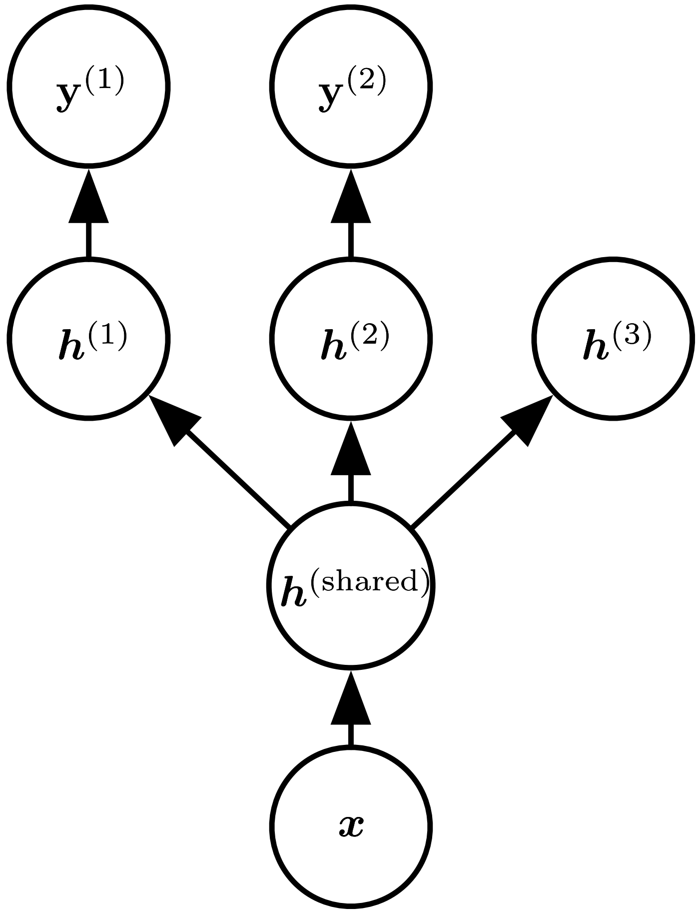
Dropout

- Random sample of connection weights is set to zero
- Train different network model each time
- Learn more robust, generalizable features



[Goodfellow, Bengio, Courville 2016]

Multitask learning

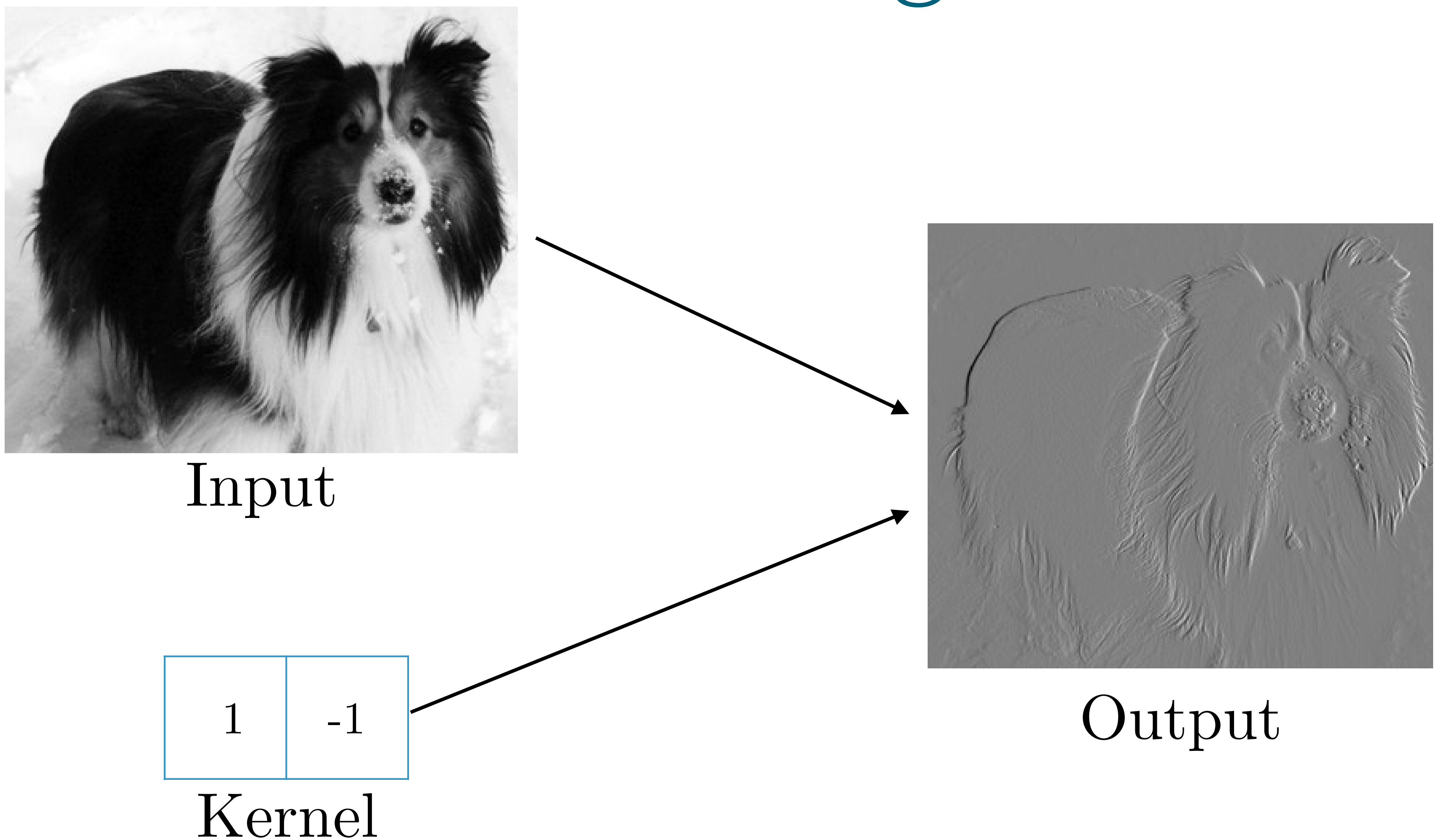


- Shared parameters are trained with more data
- Improved generalization error due to increased statistical strength

[Goodfellow, Bengio, Courville 2016]

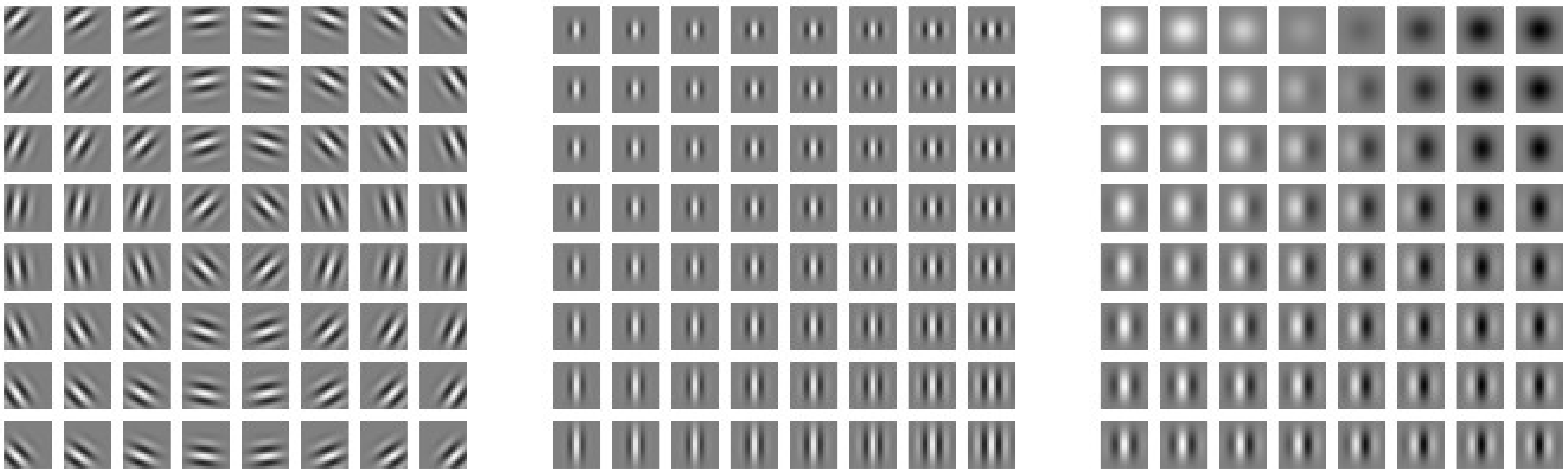
Components of popular architectures

Convolution as edge detector



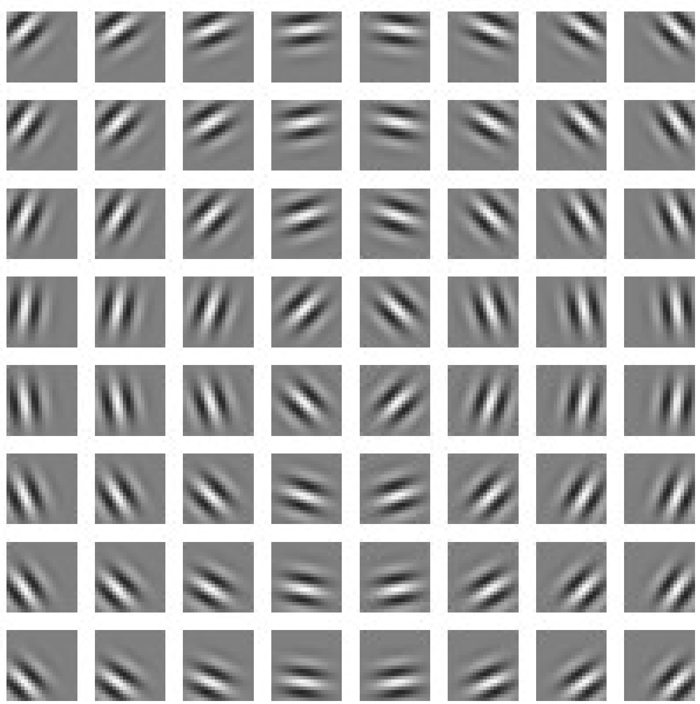
[Goodfellow, Bengio, Courville 2016]

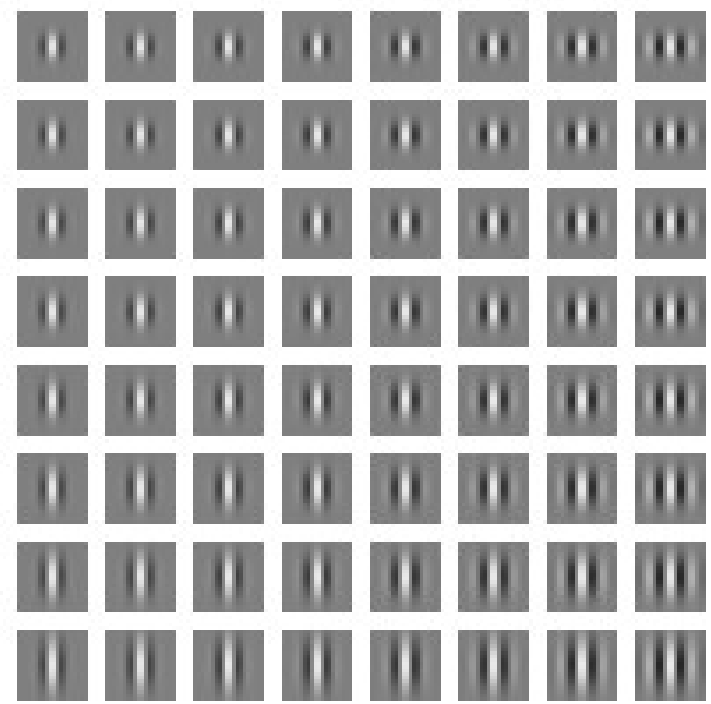
Gabor wavelets (kernels)

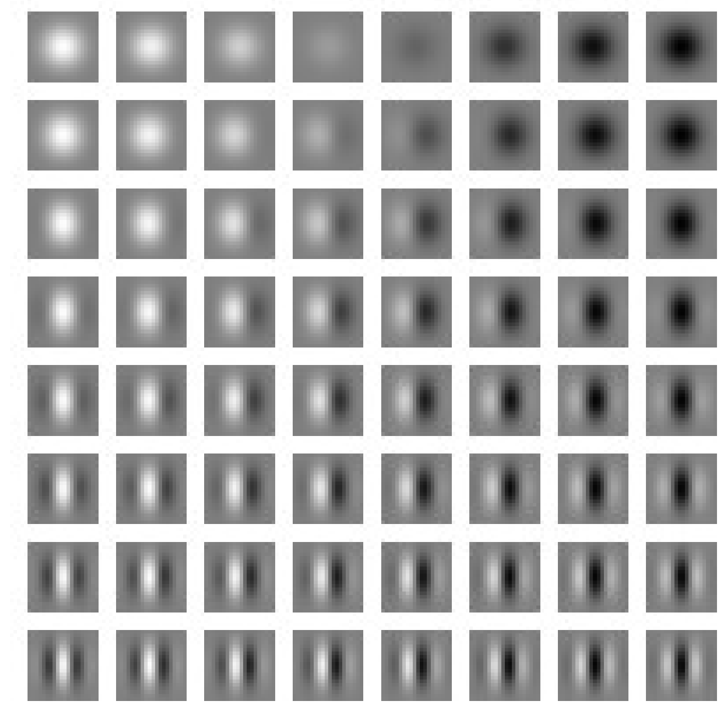


[Goodfellow, Bengio, Courville 2016]

Gabor wavelets (kernels)



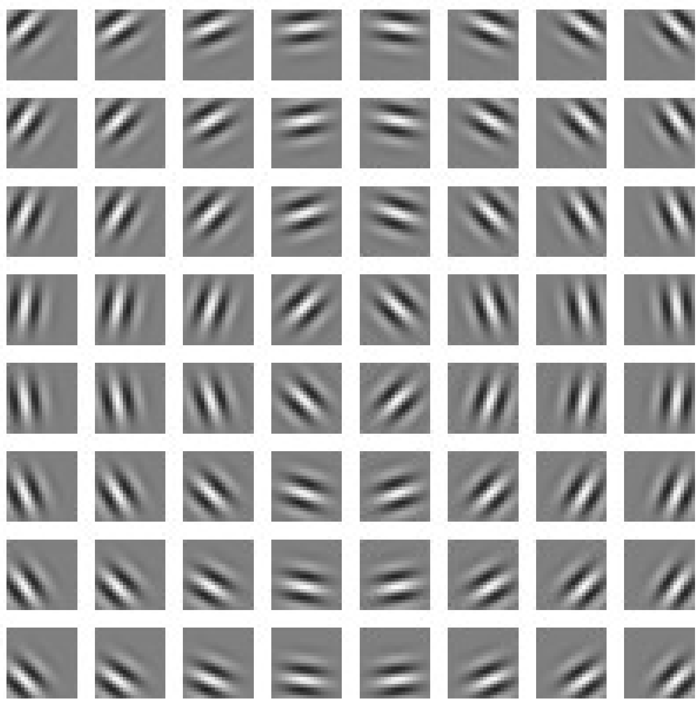


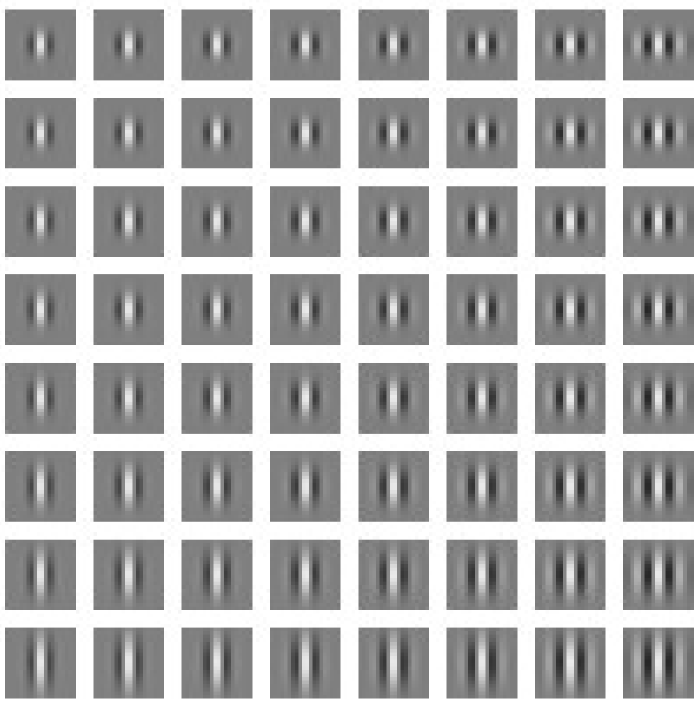


Local average, first derivative

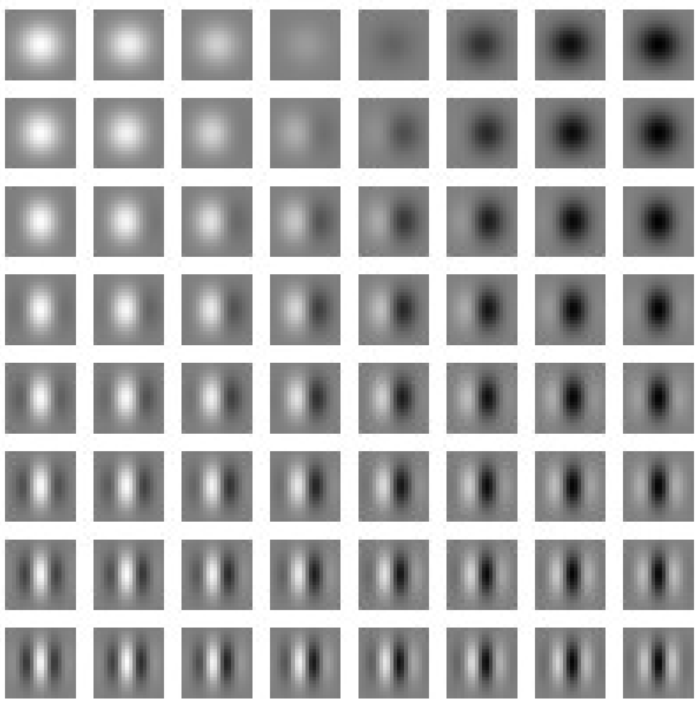
[Goodfellow, Bengio, Courville 2016]

Gabor wavelets (kernels)





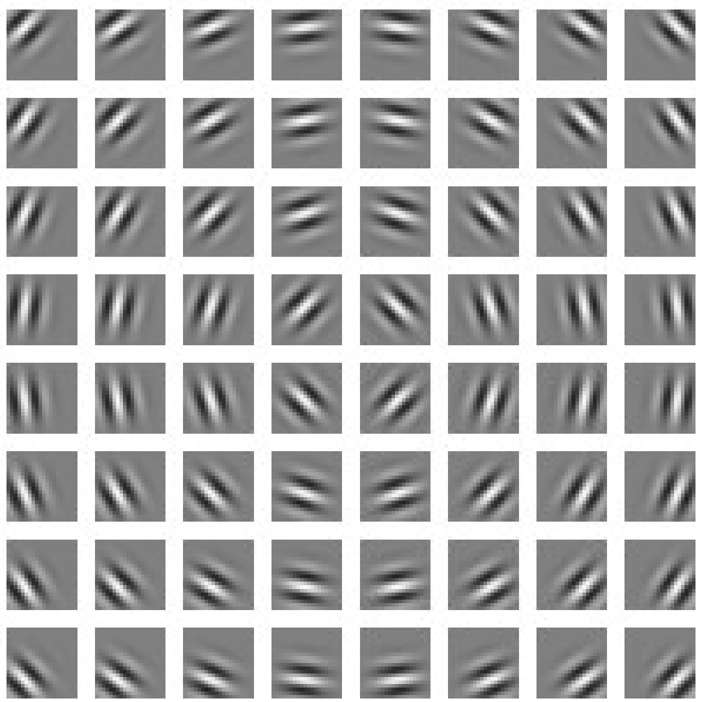
Second derivative (curvature)



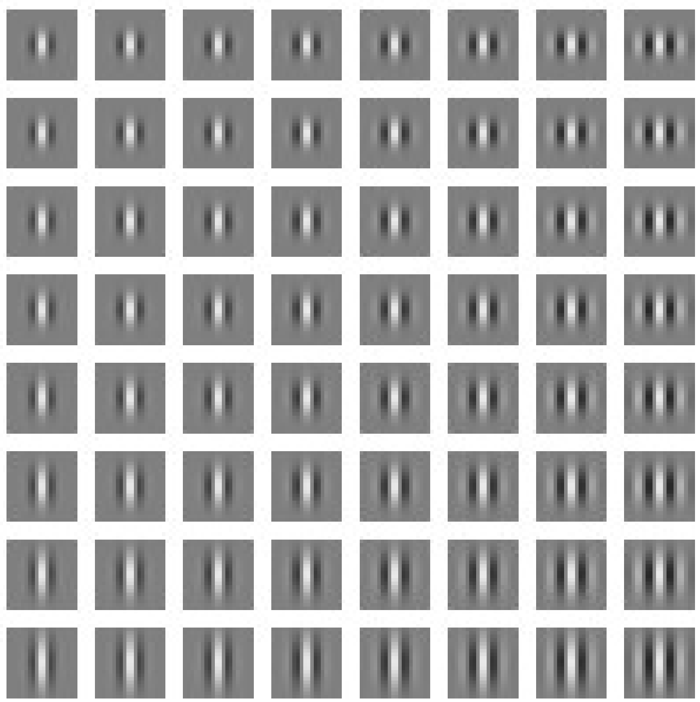
Local average, first derivative

[Goodfellow, Bengio, Courville 2016]

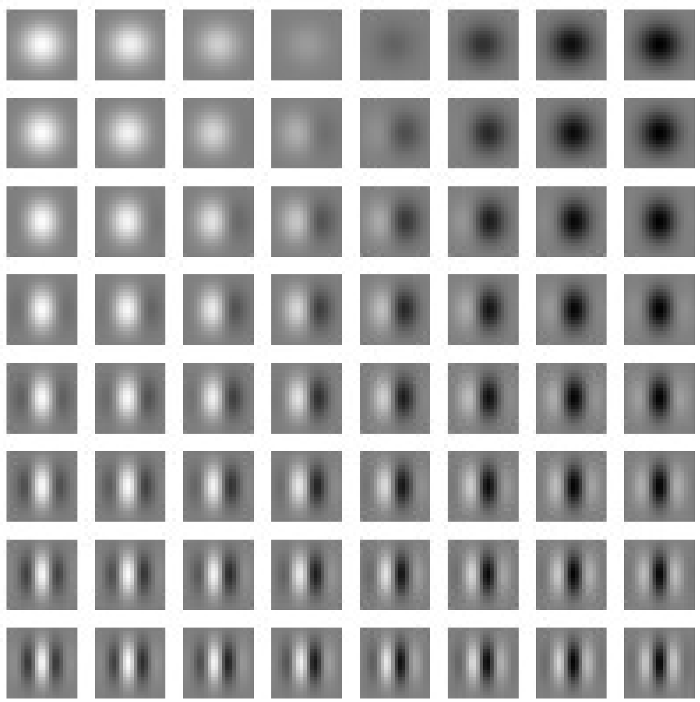
Gabor wavelets (kernels)



Directional second derivative



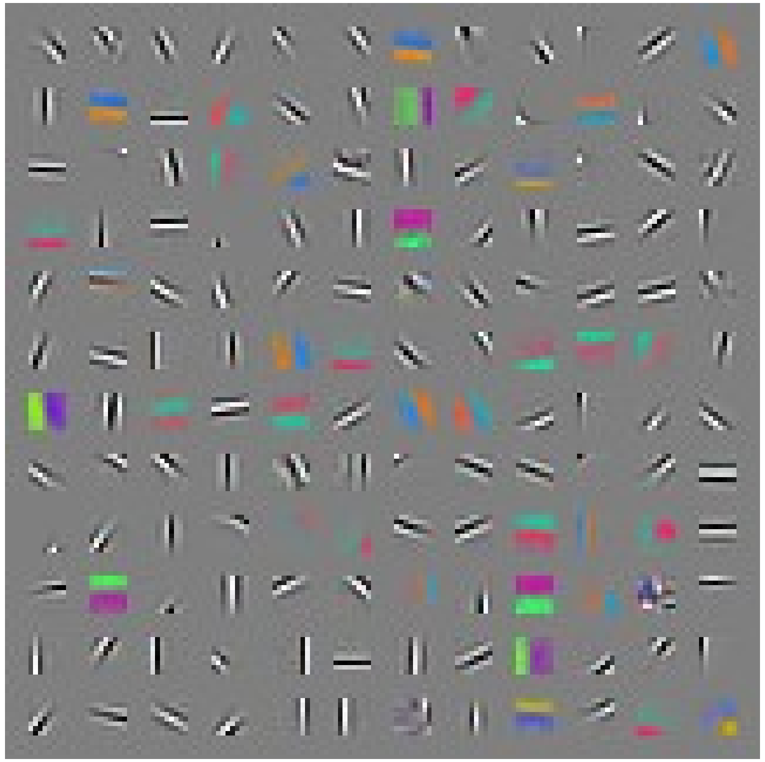
Second derivative (curvature)

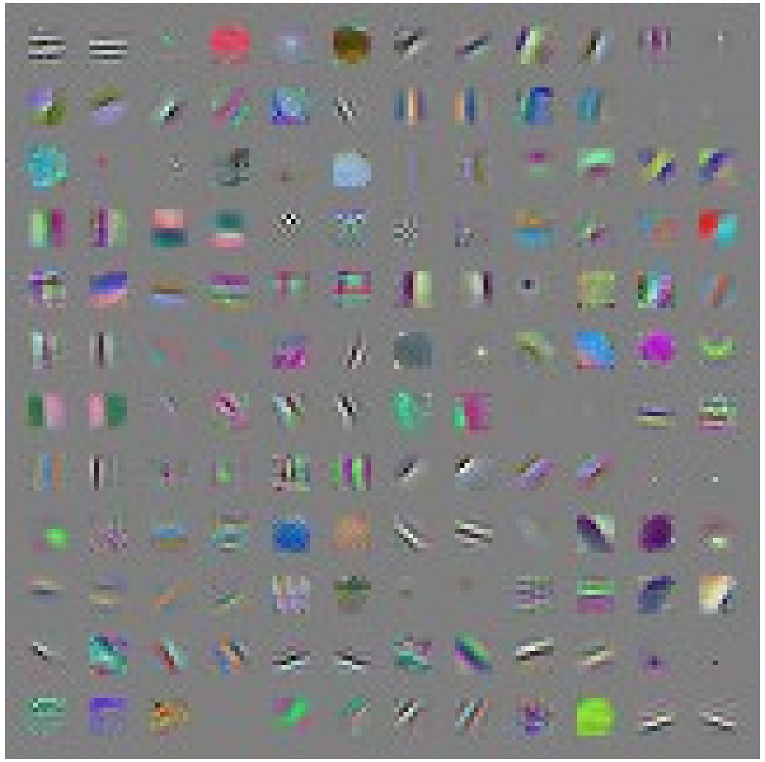


Local average, first derivative

[Goodfellow, Bengio, Courville 2016]

Gabor-like learned kernels

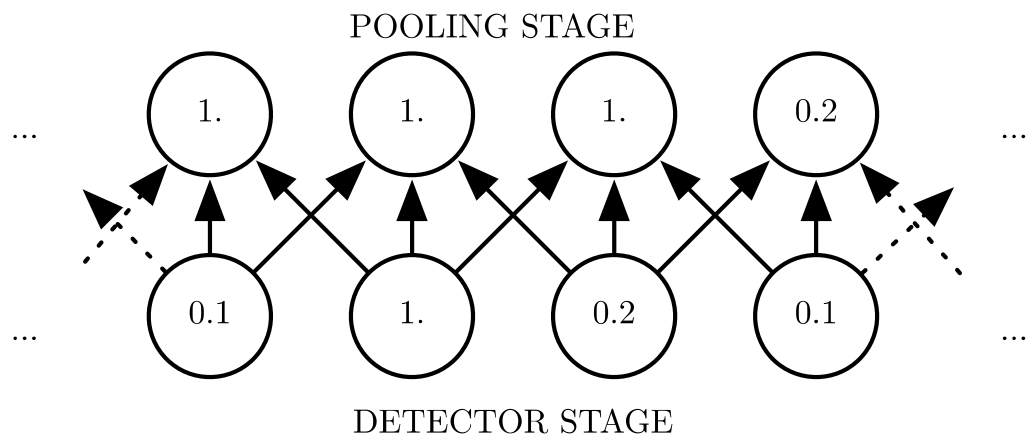


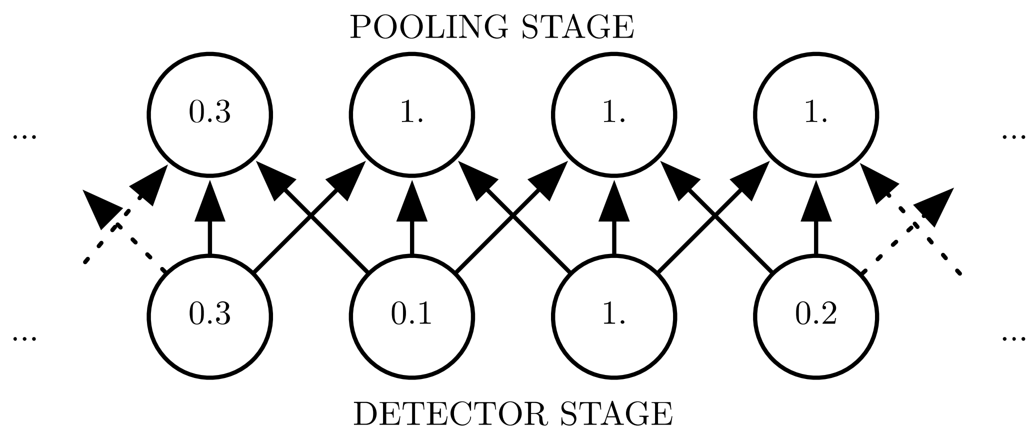


- Features extractors provided by pretrained networks

[Goodfellow, Bengio, Courville 2016]

Max pooling translation invariance

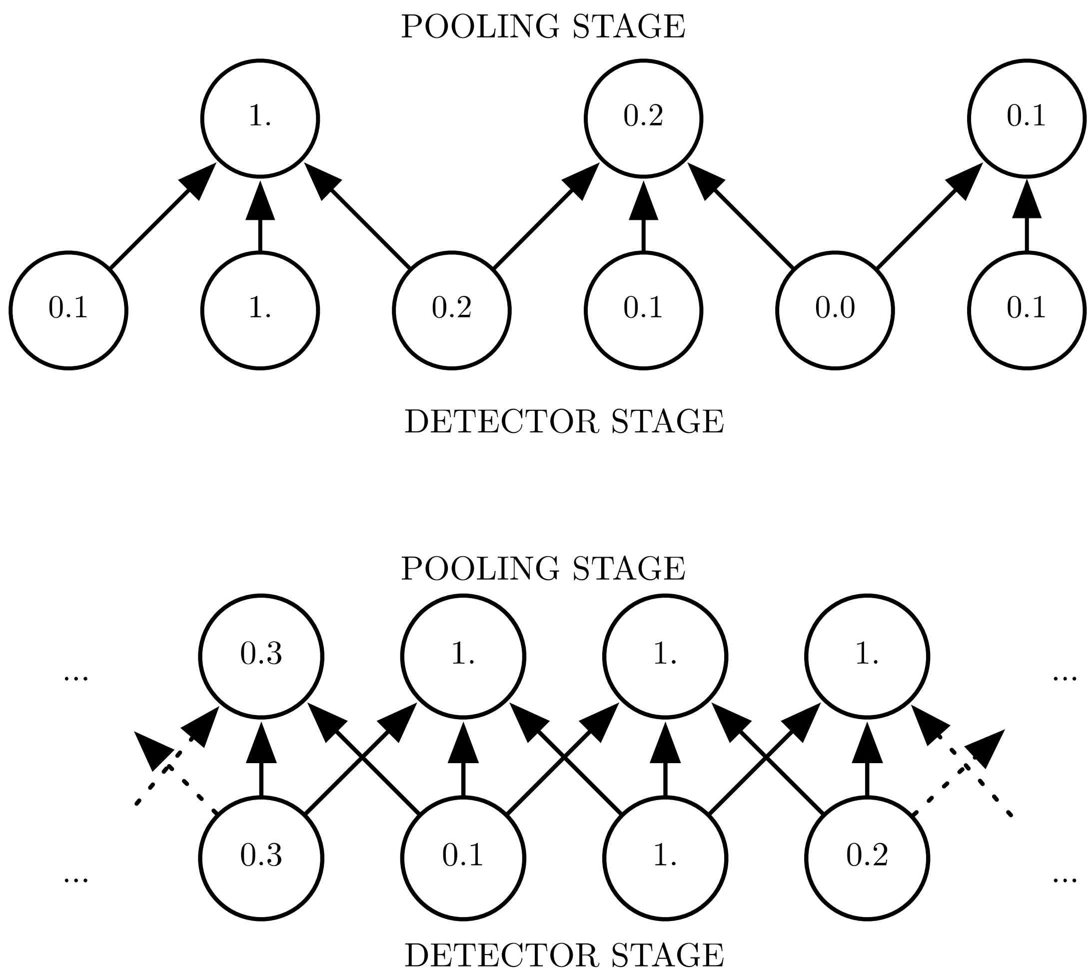




- Take max of certain neighbourhood

[Goodfellow, Bengio, Courville 2016]

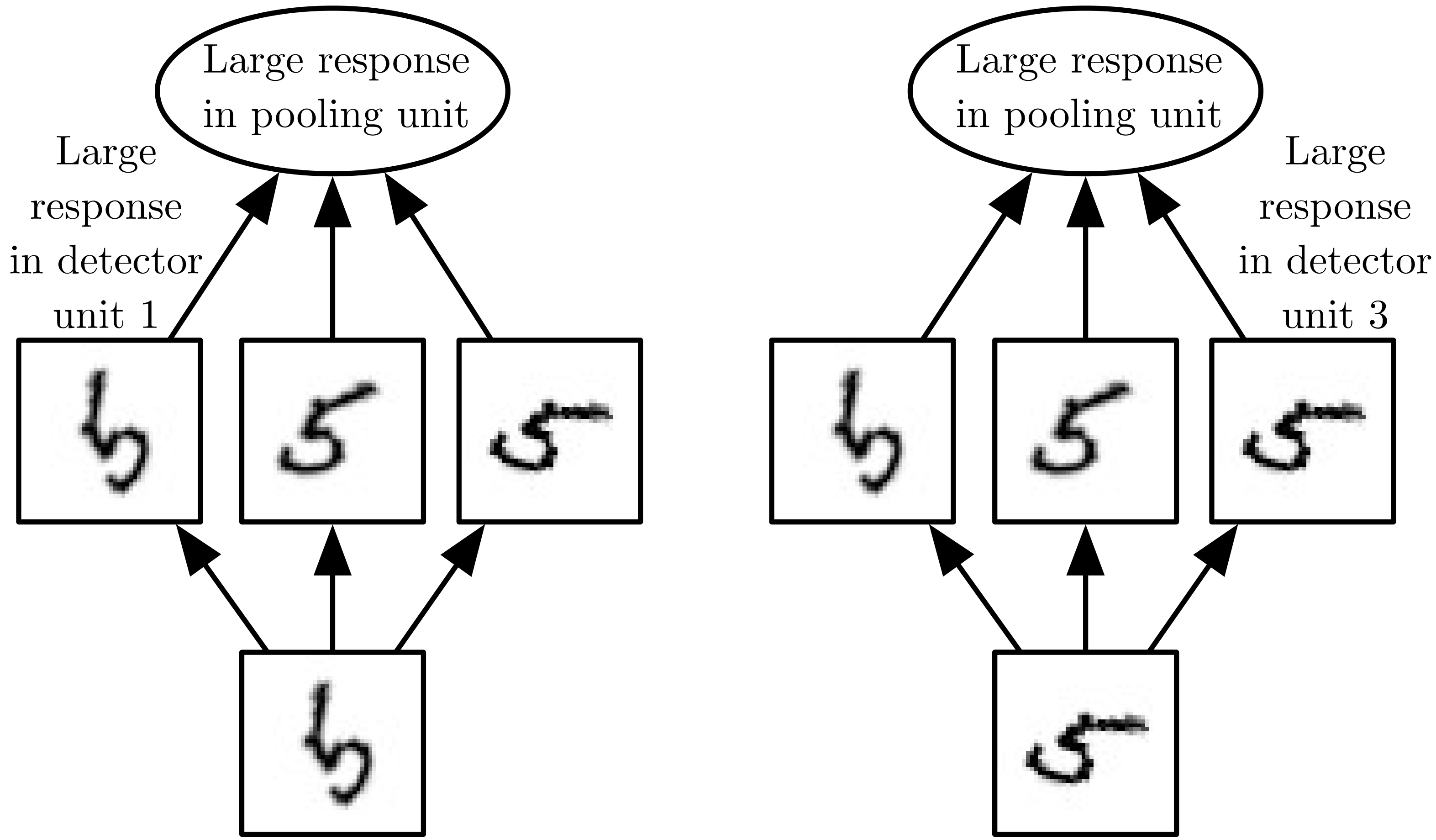
Max pooling translation invariance



- Take max of certain neighbourhood
- Often combined followed by downsampling

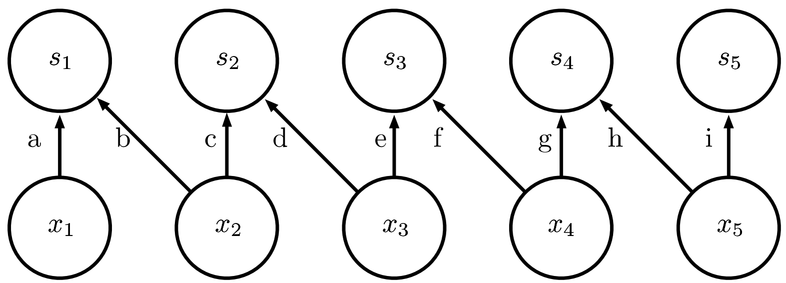
[Goodfellow, Bengio, Courville 2016]

Max pooling transform invariance



[Goodfellow, Bengio, Courville 2016]

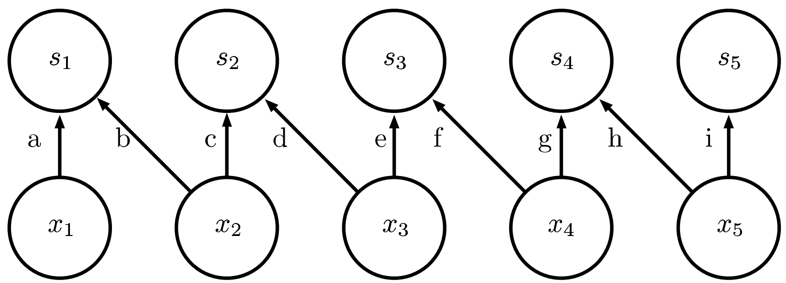
Types of connectivity



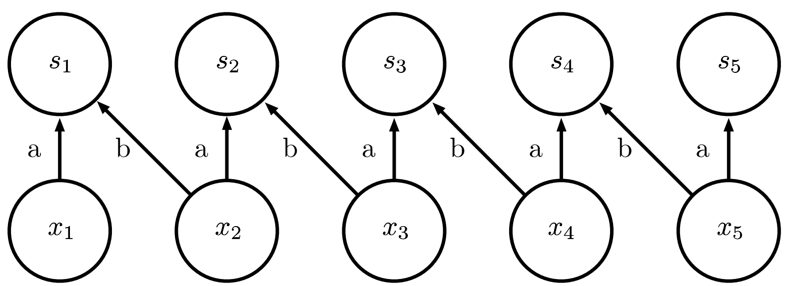
Local connection:
like convolution,
but no sharing

[Goodfellow, Bengio, Courville 2016]

Types of connectivity



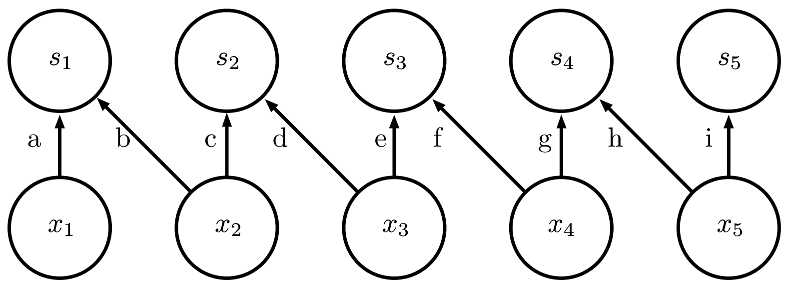
Local connection: like convolution, but no sharing



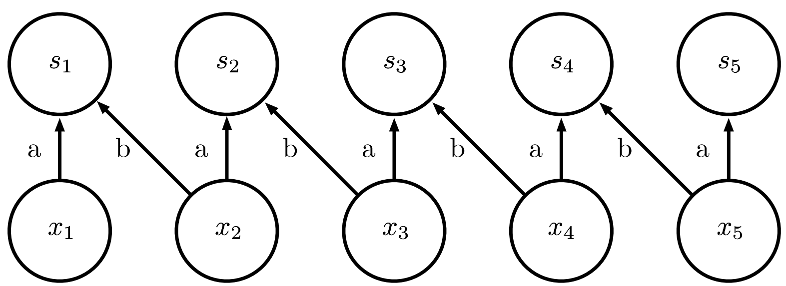
Convolution

[Goodfellow, Bengio, Courville 2016]

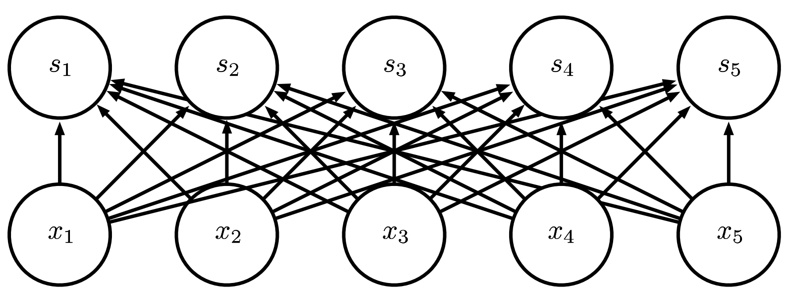
Types of connectivity



Local connection:
like convolution,
but no sharing



Convolution



Fully connected

[Goodfellow, Bengio, Courville 2016]

Choosing architecture family

Choosing architecture family

- No structure $\rightarrow$ fully connected

Choosing architecture family

- No structure $\rightarrow$ fully connected
- Spatial structure $\rightarrow$ convolutional

Choosing architecture family

- No structure $\rightarrow$ fully connected
- Spatial structure $\rightarrow$ convolutional
- Sequential structure $\rightarrow$ recurrent

Optimization Algorithm

- Lots of variants address choice of learning rate
- See Visualization of Algorithms
- AdaDelta and RMSprop often work well

Software for Deep Learning

Current Frameworks

- Tensorflow / Keras
- Pytorch
- DL4J
- Caffe
- And many more
- Most have CPU-only mode but much faster on NVIDIA GPU

Development strategy

- Identify needs: High accuracy or low accuracy?
- Choose metric
  - Accuracy (% of examples correct), Coverage (% examples processed)
  - Precision TP/(TP+FP), Recall TP/(TP+FN)
  - Amount of error in case of regression
- Build end-to-end system
  - Start from baseline, e.g. initialize with pre-trained network
- Refine driven by data

Sources

- I. Goodfellow, Y. Bengio, A. Courville “Deep Learning” MIT Press 2016 [link]