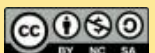
مرکز آموزش‌های الکترونیکی دانشگاه شهید بهشتی

شبکه‌های عصبی مصنوعی

Artificial Neural Networks

Ali Katanforoush*

* *Department of Data and Computer Sciences, Shahid Beheshti University*



Weight Regularization منظم سازی وزن

L2-norm

ضریب منظم سازی

$$Loss(w) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \hat{y}(x^{(i)};w)\right)^2 + \frac{\lambda}{2}\sum_{j=1}^{M} w_j^2$$

اضافه کردن یک عبارت منظم سازی در تعریف تابع زیان

Neural networks

Training neural networks - regularization

$$w^{new} = (1-\alpha)w^{old} - \eta \nabla_w f$$

Weight decay

REGULARIZATION

Topics: L2 regularization

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j \left(W_{i,j}^{(k)}\right)^2 = \sum_k ||\, \mathbf{W}^{(k)}||_F^2$$

- Gradient: $\nabla_{\mathbf{W}^{(k)}} \Omega(\boldsymbol{\theta}) = 2\, \mathbf{W}^{(k)}$

- Only applied on weights, not on biases (weight decay)
- Can be interpreted as having a Gaussian prior over the weights

REGULARIZATION

Topics: L1 regularization

$$\Omega(\boldsymbol{\theta}) = \sum_k \sum_i \sum_j |W_{i,j}^{(k)}|$$

- Gradient: $\nabla_{\mathbf{W}^{(k)}} \Omega(\boldsymbol{\theta}) = \text{sign}(\mathbf{W}^{(k)})$
  - where $\text{sign}(\mathbf{W}^{(k)})_{i,j} = 1_{\mathbf{W}_{i,j}^{(k)} > 0} - 1_{\mathbf{W}_{i,j}^{(k)} < 0}$
- Also only applied on weights
- Unlike L2, L1 will push certain weights to be exactly 0
- Can be interpreted as having a Laplacian prior over the weights

Neural networks

Deep learning - dropout

DEEP LEARNING

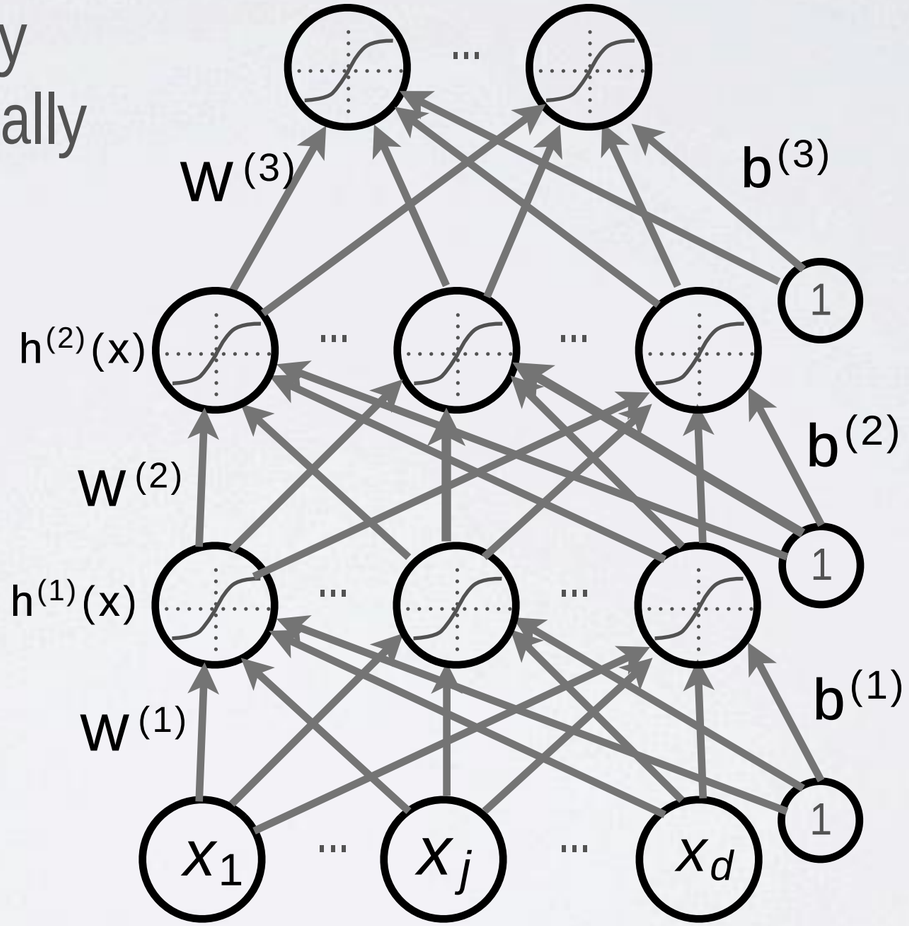
Topics: why training is hard

- Depending on the problem, one or the other situation will tend to prevail
- If first hypothesis (underfitting): use better optimization
  - this is an active area of research
- If second hypothesis (overfitting): use better regularization
  - unsupervised learning → (GAN)
  - stochastic «dropout» training

DROPOUT

Topics: dropout

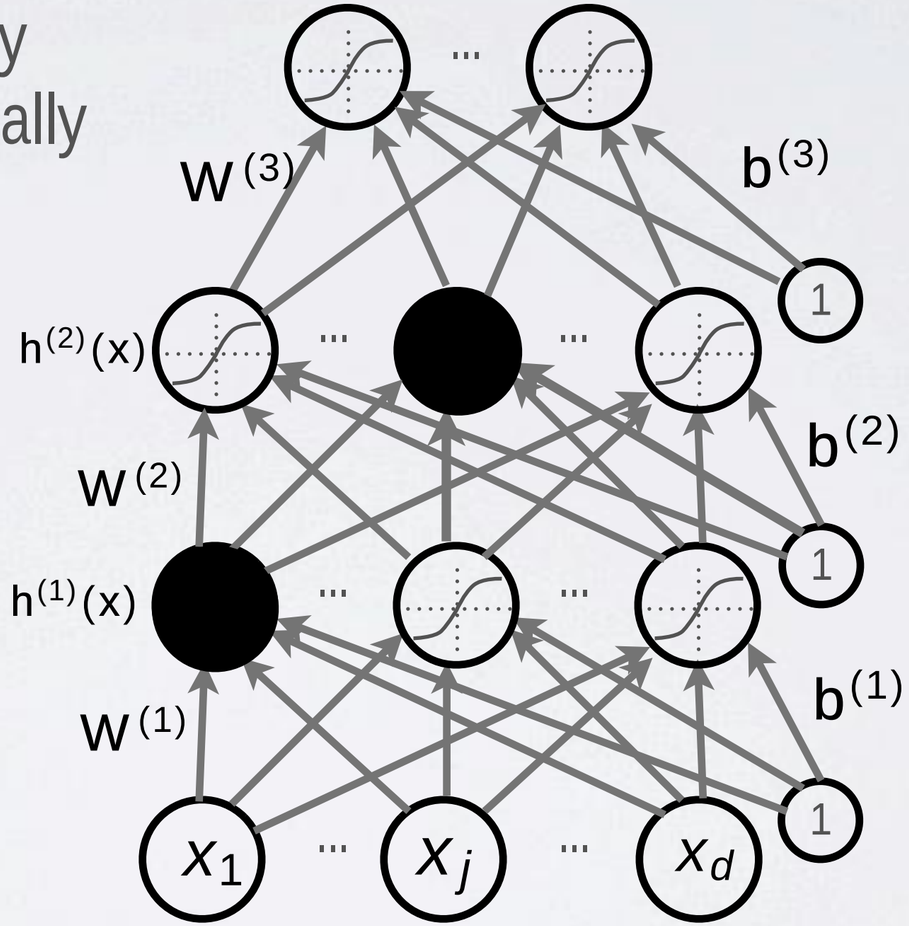
- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well



DROPOUT

Topics: dropout

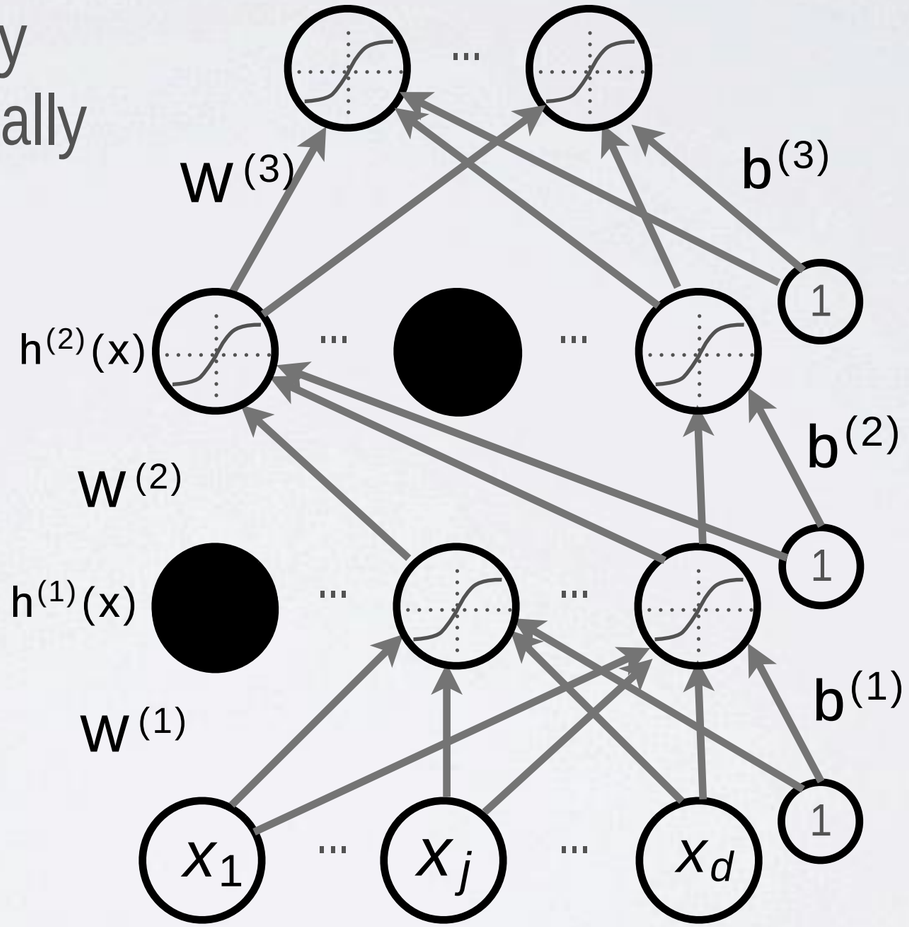
- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well



DROPOUT

Topics: dropout

- Idea: «cripple» neural network by removing hidden units stochastically
  - each hidden unit is set to 0 with probability 0.5
  - hidden units cannot co-adapt to other units
  - hidden units must be more generally useful
- Could use a different dropout probability, but 0.5 usually works well



DROPOUT

Topics: dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

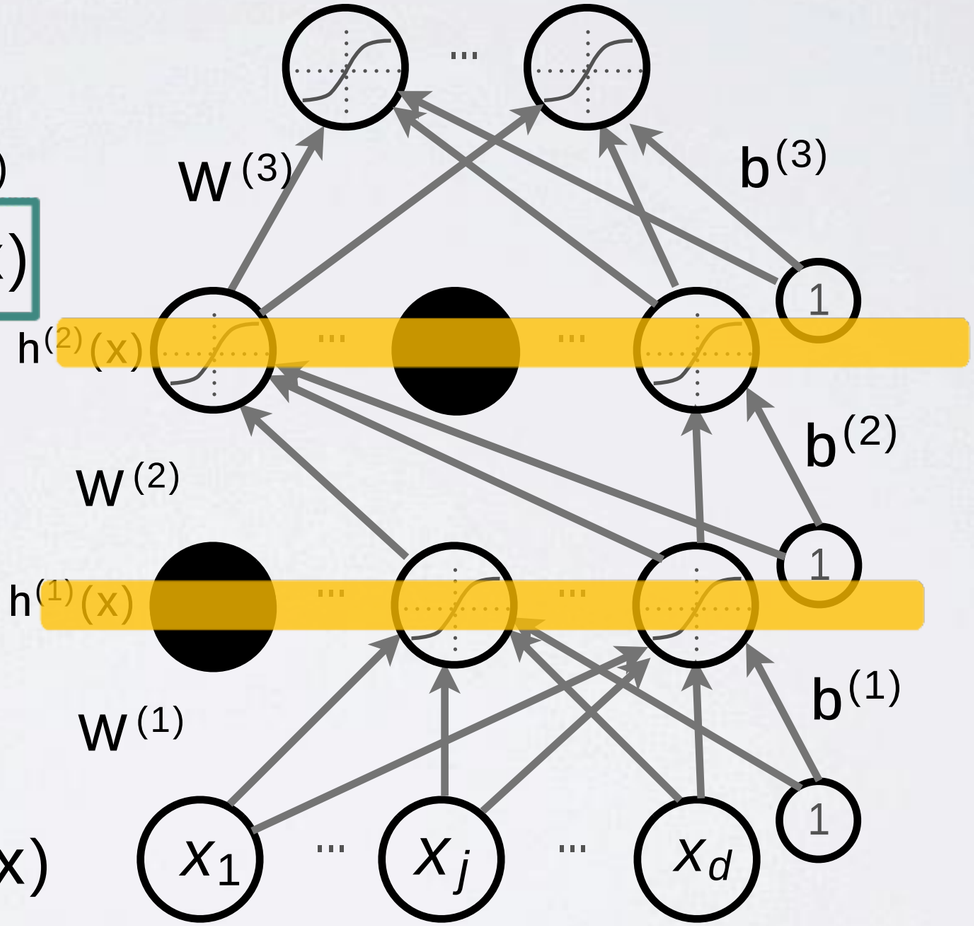
  - hidden layer activation (k from 1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = g(\mathbf{a}^{(k)}(\mathbf{x}))$$

Dropout بدون لایه

  - output layer activation ($k=L+1$):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = o(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



DROPOUT

Topics: dropout

- Use random binary masks $\mathbf{m}^{(k)}$
  - layer pre-activation for $k>0$ $(\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x})$

$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k)}\mathbf{h}^{(k-1)}(\mathbf{x})$$

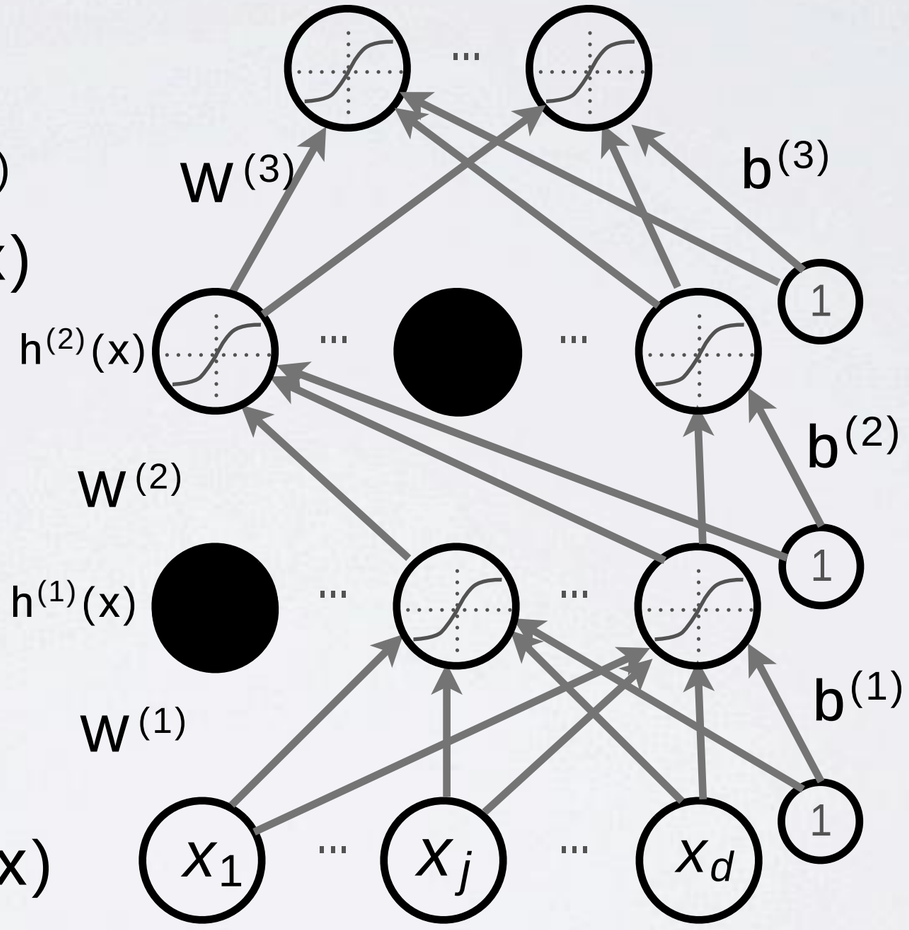
Dropout إلغاء

  - hidden layer activation (k from 1 to L):

$$\mathbf{h}^{(k)}(\mathbf{x}) = \mathbf{g}(\mathbf{a}^{(k)}(\mathbf{x})) \odot \mathbf{m}^{(k)}$$

  - output layer activation ($k=L+1$):

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{(L+1)}(\mathbf{x})) = \mathbf{f}(\mathbf{x})$$



$m = \langle 0, 1, 1, 0 \cdots, 1 \rangle$

0 ل 1

dropout في

$p = 0.2$

$\mathbb{E}[m_j] =$

$0 \times p + 1 \times (1-p)$

$= 1 - p = 0.8$

DROPOUT

Topics: dropout backpropagation

- This assumes a forward propagation has been made before
  - compute output gradient (before activation)

$$\nabla_{\mathbf{a}^{(L+1)}(\mathbf{x})} - \log f(\mathrm{x})_y \Leftarrow= -(\mathrm{e}(y) - \mathrm{f}(\mathrm{x}))$$

  - for k from $L+1$ to 1
    - compute gradients of hidden layer parameter

$$\nabla_{\mathbf{W}^{(k)}} - \log f(\mathrm{x})_y \Leftarrow= \left(\nabla_{\mathbf{a}^{(k)}(\mathbf{x})} - \log f(\mathrm{x})_y\right) \mathrm{h}^{(k-1)}(\mathrm{x})^\top$$

$$\nabla_{\mathbf{b}^{(k)}} - \log f(\mathrm{x})_y \Leftarrow= \nabla_{\mathbf{a}^{(k)}(\mathbf{x})} - \log f(\mathrm{x})_y$$

    - compute gradient of hidden layer below

$$\nabla_{\mathbf{h}^{(k-1)}(\mathbf{x})} - \log f(\mathrm{x})_y \Leftarrow= \mathrm{W}^{(k)\top} \left(\nabla_{\mathbf{a}^{(k)}(\mathbf{x})} - \log f(\mathrm{x})_y\right)$$

    - compute gradient of hidden layer below (before activation)

$$\nabla_{\mathbf{a}^{(k-1)}(\mathbf{x})} - \log f(\mathrm{x})_y \Leftarrow= \left(\nabla_{\mathbf{h}^{(k-1)}(\mathbf{x})} - \log f(\mathrm{x})_y\right) \odot [\ldots, g'(a^{(k-1)}(\mathrm{x})_j), \ldots]$$

بدون لایه Dropout

DROPOUT

Topics: dropout backpropagation

- This assumes a forward propagation has been made before
  - compute output gradient (before activation)

$$\nabla_{\mathbf{a}^{(L+1)}(\mathbf{x})} - \log f(\mathrm{x})_y \Leftarrow= -(\mathrm{e}(y) - \mathrm{f}(\mathrm{x}))$$

  - for k from $L+1$ to 1
    - compute gradients of hidden layer parameter

$$\nabla_{\mathbf{W}^{(k)}} - \log f(\mathrm{x})_y \Leftarrow= \left(\nabla_{\mathbf{a}^{(k)}(\mathbf{x})} - \log f(\mathrm{x})_y\right) \mathrm{h}^{(k-1)}(\mathrm{x})^\top$$

$$\nabla_{\mathbf{b}^{(k)}} - \log f(\mathrm{x})_y \Leftarrow= \nabla_{\mathbf{a}^{(k)}(\mathbf{x})} - \log f(\mathrm{x})_y$$

includes the mask $\mathrm{m}^{(k-1)}$

    - compute gradient of hidden layer below

$$\nabla_{\mathbf{h}^{(k-1)}(\mathbf{x})} - \log f(\mathrm{x})_y \Leftarrow= \mathrm{W}^{(k)\top} \left(\nabla_{\mathbf{a}^{(k)}(\mathbf{x})} - \log f(\mathrm{x})_y\right)$$

    - compute gradient of hidden layer below (before activation)

$$\nabla_{\mathbf{a}^{(k-1)}(\mathbf{x})} - \log f(\mathrm{x})_y \Leftarrow= \left(\nabla_{\mathbf{h}^{(k-1)}(\mathbf{x})} - \log f(\mathrm{x})_y\right) \odot [\ldots, g'(a^{(k-1)}(\mathrm{x})_j), \ldots] \odot \mathrm{m}^{(k-1)}$$

Dropout لا!

DROPOUT

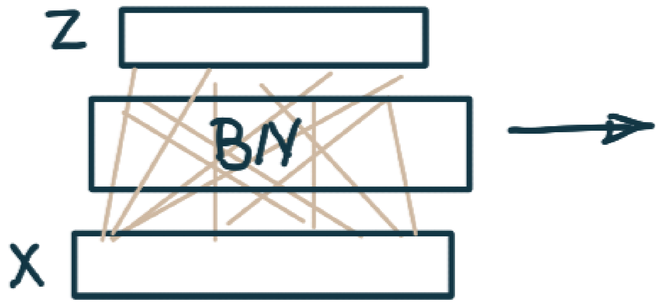
Topics: test time classification

- At test time, we replace the masks by their expectation
  - this is simply the constant vector 0.5 if dropout probability is 0.5
  - for single hidden layer, can show this is equivalent to taking the geometric average of all neural networks, with all possible binary masks
- Can be combined with unsupervised pre-training
- Beats regular backpropagation on many datasets
  - Improving neural networks by preventing co-adaptation of feature detectors. Hinton, Srivastava, Krizhevsky, Sutskever and Salakhutdinov, 2012.

Batch Normalization

مثال:

لایه تمام متصل
با تابع فعال سازی
ReLU



mini-batch با اندازه : m

$$Z = X W$$

$$\tilde{Z} = Z - \frac{1}{m}\sum_{i=1}^{m} Z_{i,:}$$

$$\hat{Z} = \frac{\tilde{Z}}{\sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m} \tilde{Z}_{i,:}^{2}}}$$

$$H = \max\{0, \gamma\hat{Z} + \beta\}$$

"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Ioffe and Szegedy 2015