Regularization

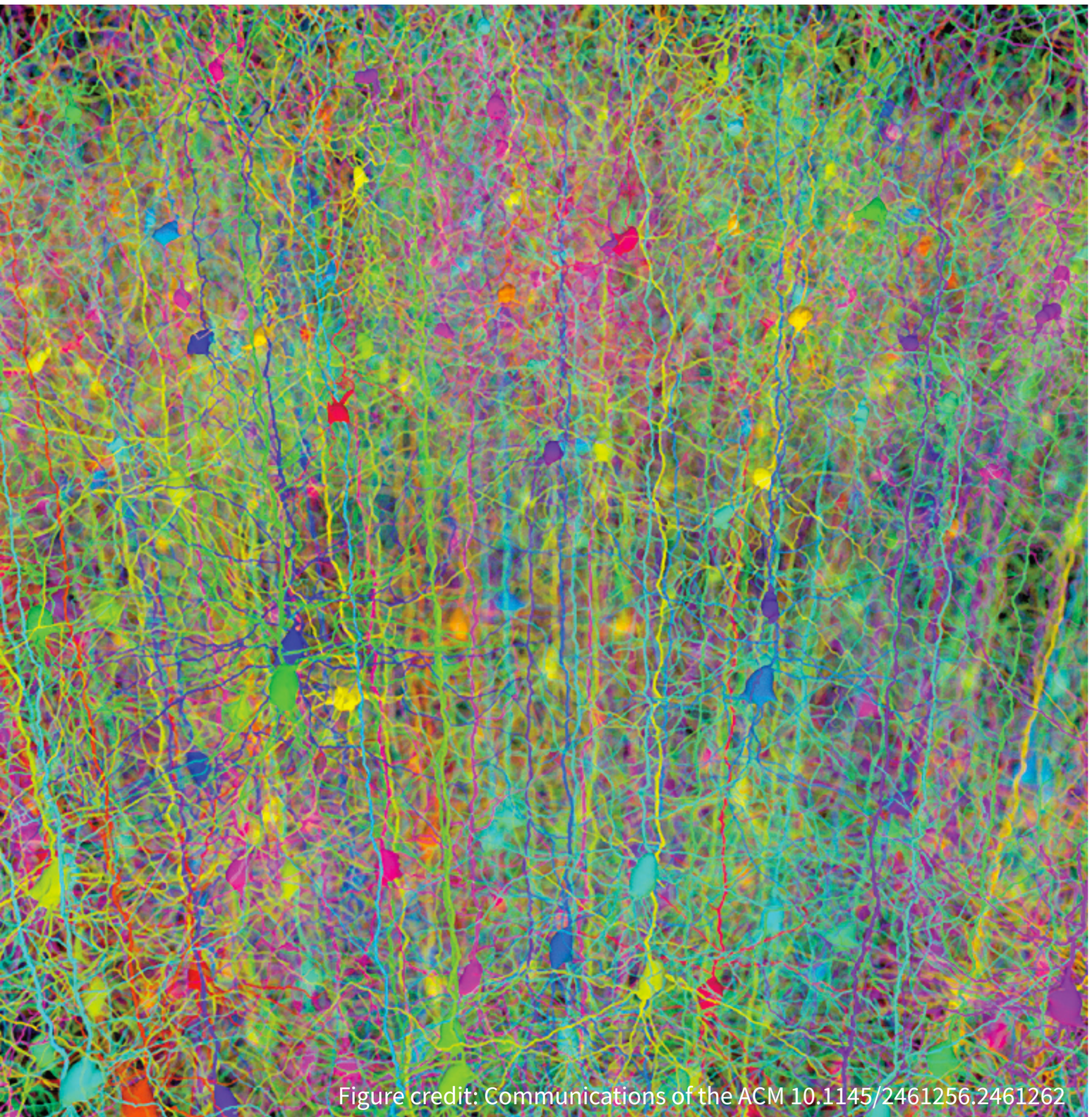


Figure credit: Communications of the ACM 10.1145/2461256.2461262

GRAHAM TAYLOR

VECTOR INSTITUTE

SCHOOL OF ENGINEERING
UNIVERSITY OF GUELPH

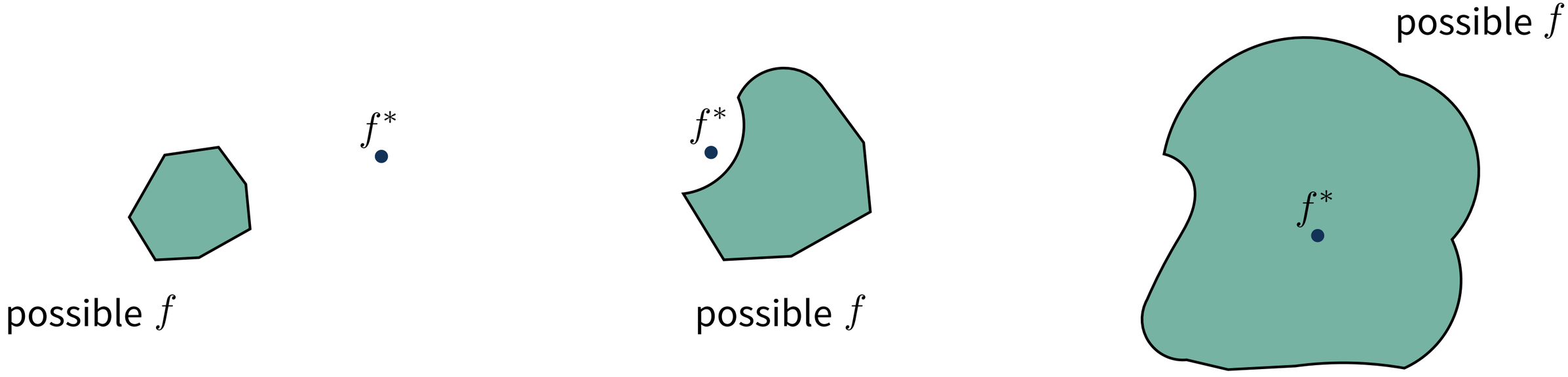
CANADIAN INSTITUTE
FOR ADVANCED RESEARCH

UNIVERSITY of GUELPH
CHANGING LIVES
IMPROVING LIFE

CIFAR
CANADIAN
INSTITUTE
FOR
ADVANCED
RESEARCH

Bias-Variance Trade-off

- **Variance:** does the model vary a lot if we change the training set?
- **Bias**: is the average model close to the true solution?
- **Generalization**: can be seen as the sum of the (squared) bias and the variance



reproduced from Larochelle

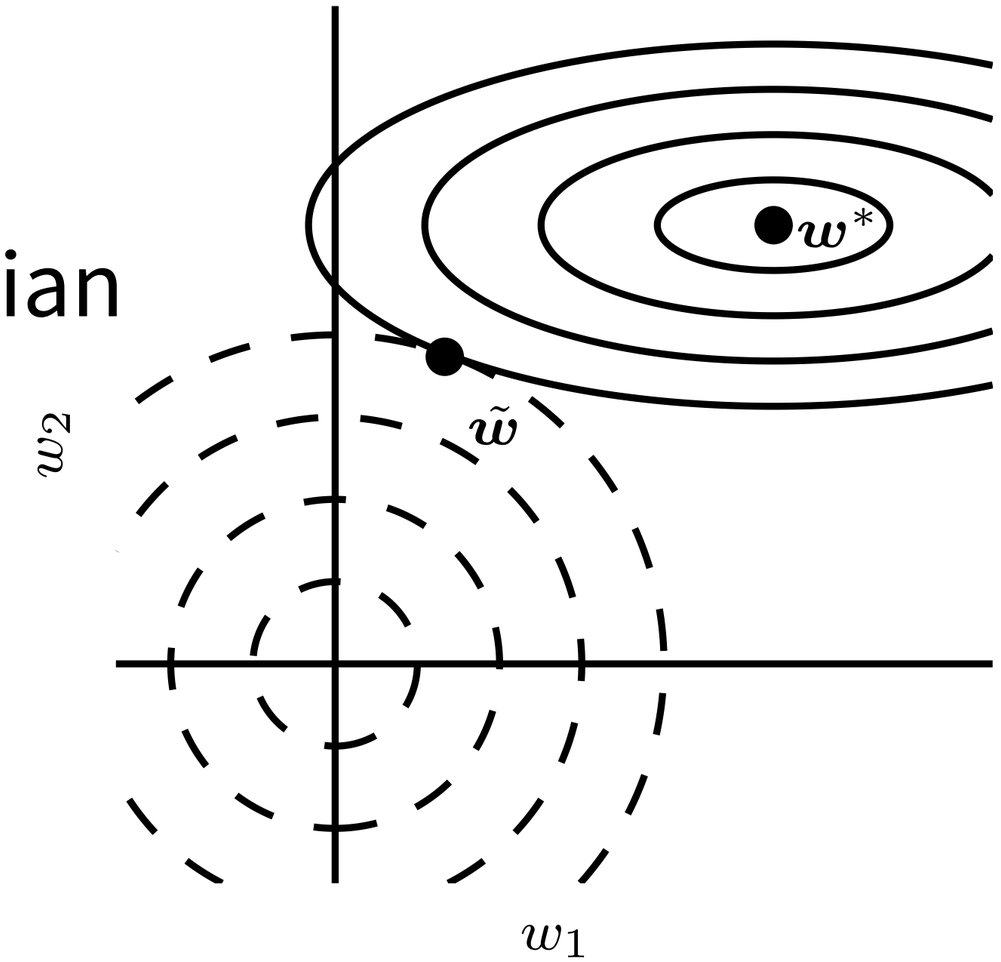
L2 Regularization

Commonly known as weight decay:

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

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

Can be interpreted as a Gaussian prior over weights



Reproduced from Goodfellow et al.

L1 Regularization

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

Gradient: $\nabla_{\boldsymbol{W}^{(k)}} \Omega(\boldsymbol{\theta}) = \text{sign}(\boldsymbol{W}^{(k)})$

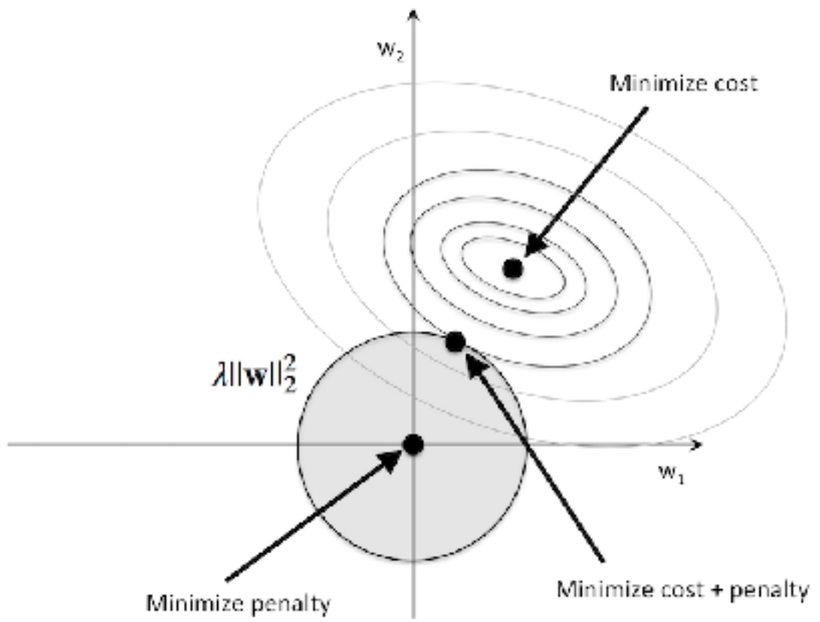
where $\text{sign}(\boldsymbol{W}^{(k)}) = \mathbf{1}_{\boldsymbol{W}_{i,j}^{(k)} > 0} - \mathbf{1}_{\boldsymbol{W}_{i,j}^{(k)} < 0}$

Results in a solution that is more sparse

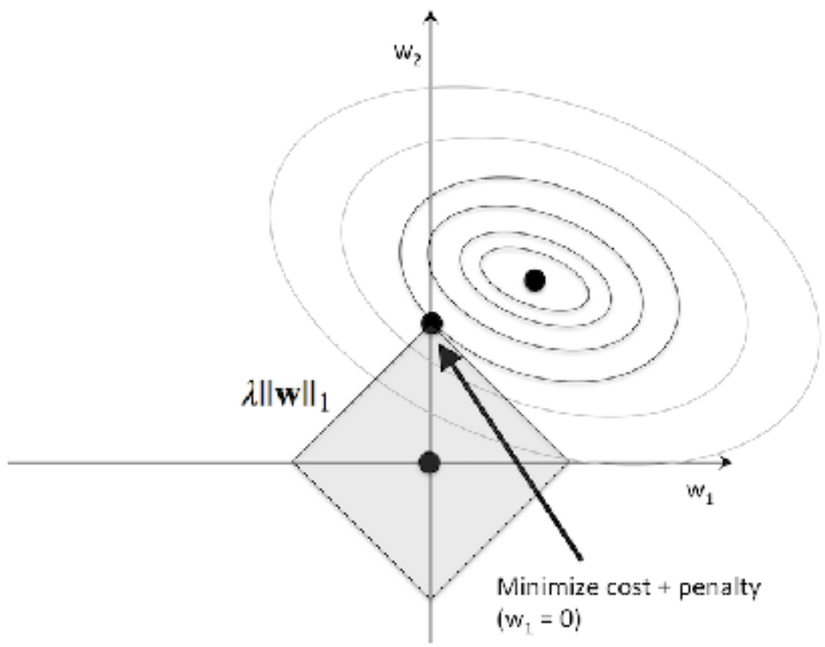
Will tend to push certain weights to be exactly zero

Can be interpreted as a Gaussian prior over weights

L2 regularization



L1 regularization



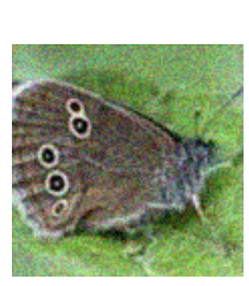
via mlxtend

Dataset Augmentation

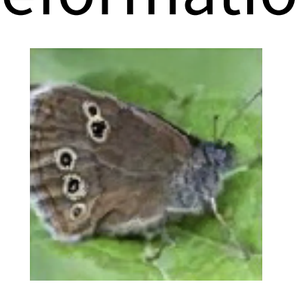
- Best way to make a ML model generalize better is collect more data
- One way to do so is synthesize new data and add it to the training set
- Easiest for classification, in domains where we know how examples should vary (e.g. images)

Affine Distortion

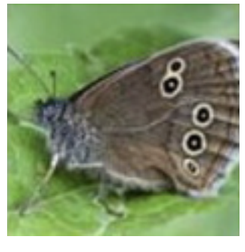
Noise



Elastic Deformation



Horizontal flip



Random Translation

Hue Shift

Reproduced from Goodfellow et al.

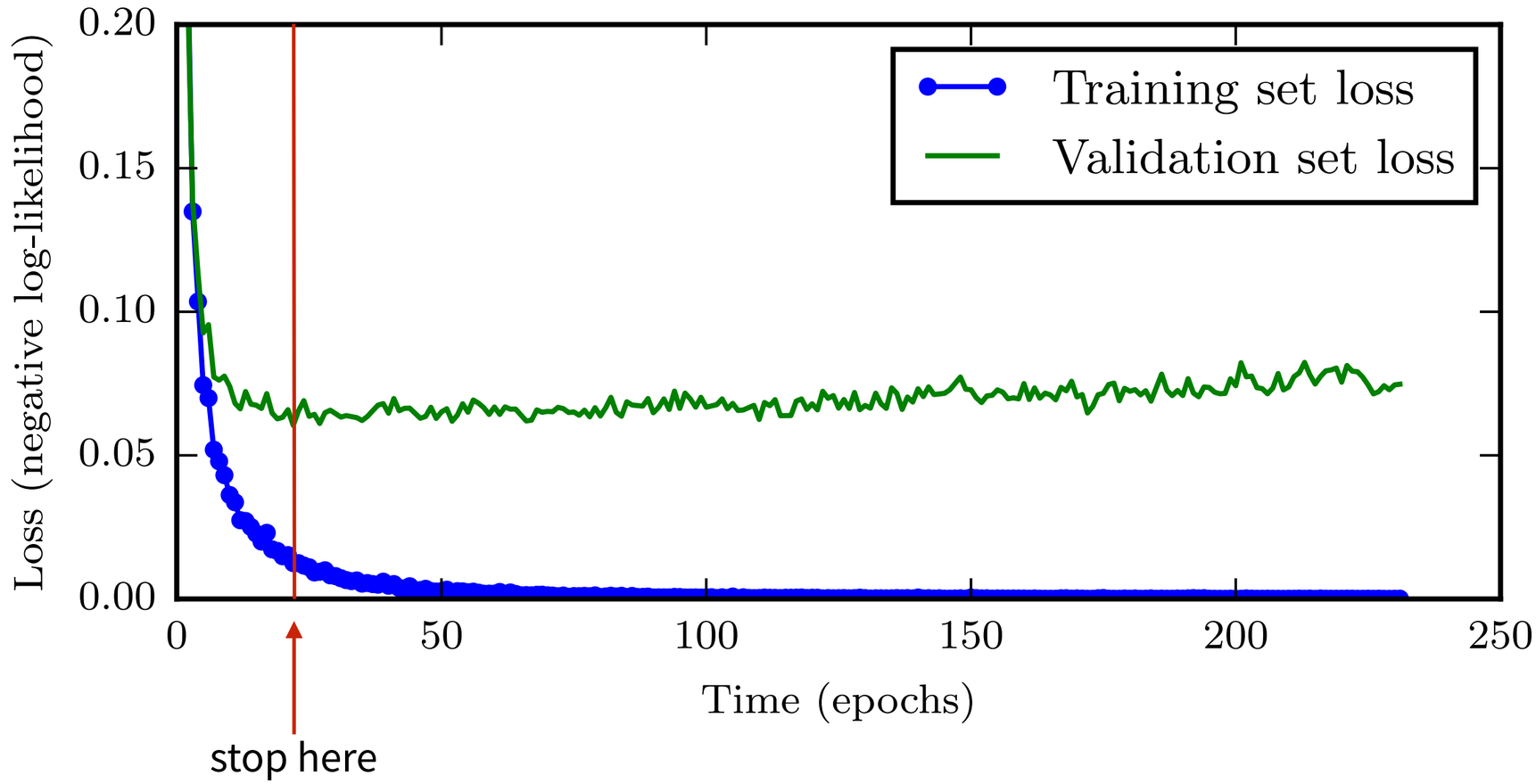
Noise Injection

Various kinds of noise added to networks can encourage robustness:

- Noise added to the inputs
  - in some models, equivalent to imposing a penalty on the norm of the weights
- Noise added to the weights
  - can be interpreted as stochastic Bayesian inference over the weights
- Noise added to the output targets
  - label smoothing is featured prominently in modern neural networks

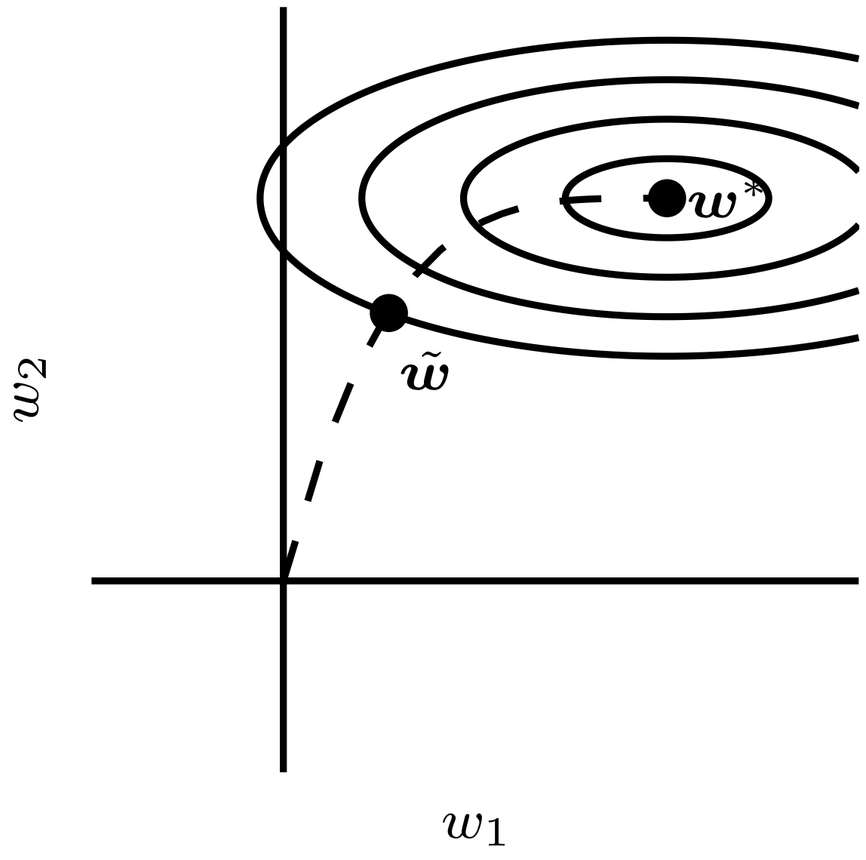
Early Stopping

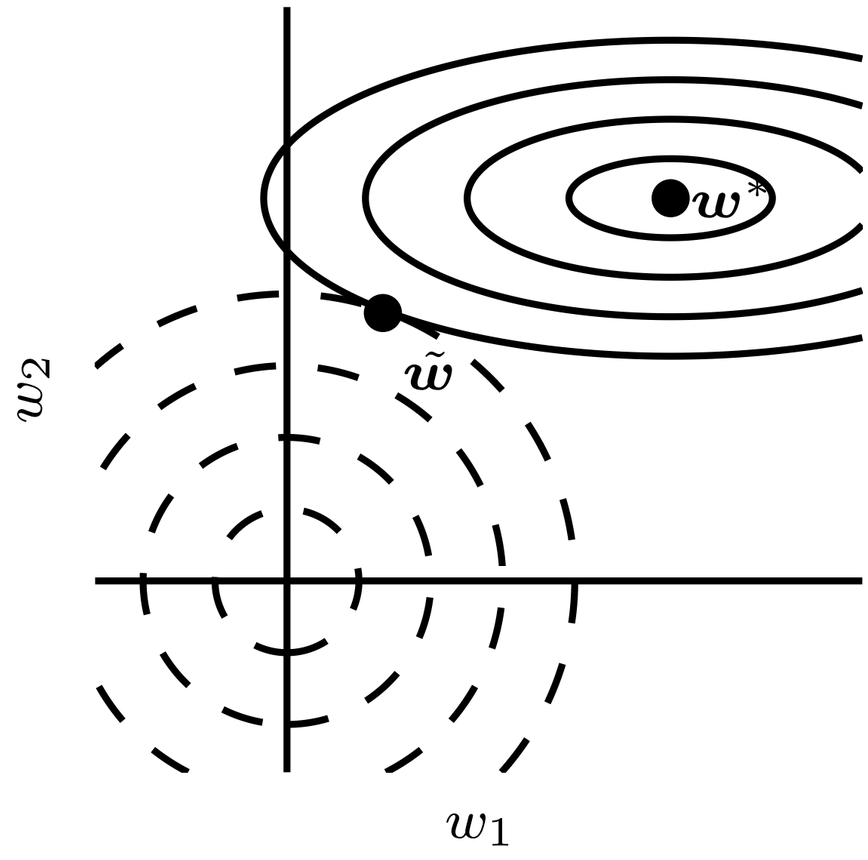
Terminate when validation set performance peaks



Reproduced from Goodfellow et al.

Early Stopping vs. Weight Decay



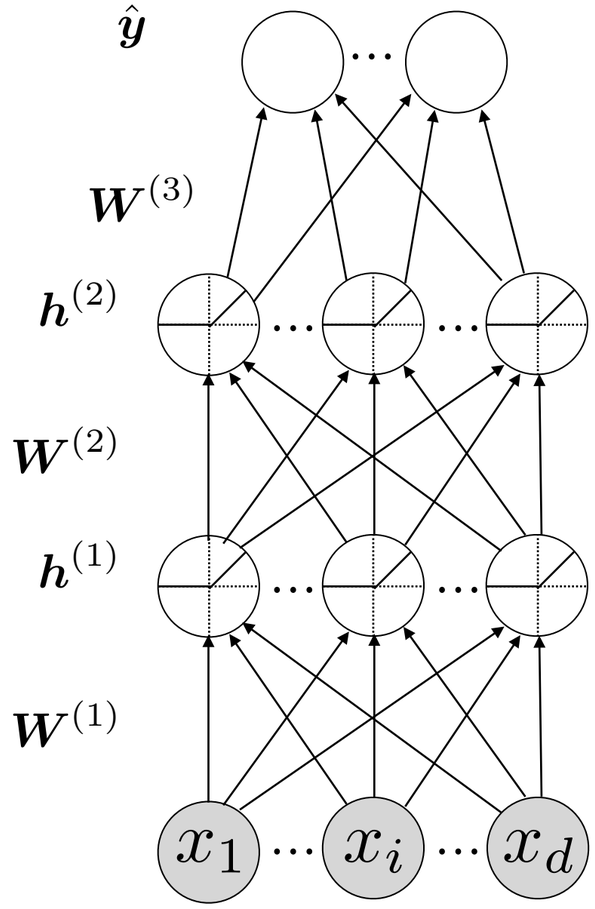


Reproduced from Goodfellow et al.

Dropout

Idea: “cripple” a neural network by removing hidden units stochastically

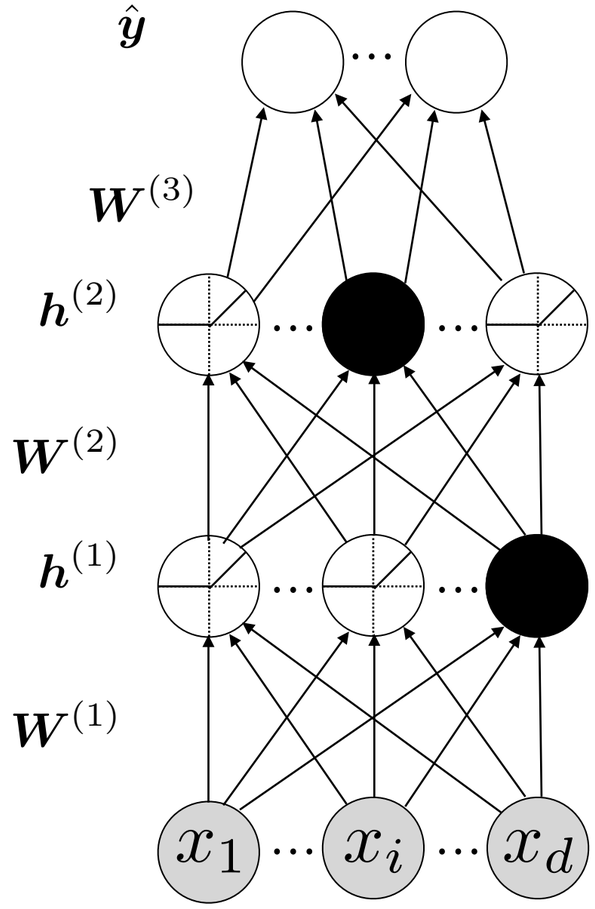
- Set each unit to 0 with probability 0.5 (could use a different probability)
- Hidden units cannot co-adapt to other units
- Causes hidden units to be more useful “on their own”



Dropout

Idea: “cripple” a neural network by removing hidden units stochastically

- Set each unit to 0 with probability 0.5 (could use a different probability)
- Hidden units cannot co-adapt to other units
- Causes hidden units to be more useful “on their own”



Dropout: Forward Pass

Implementation: use random binary masks $\boldsymbol{m}^{(k)}$ sampled at training time

- Layer pre-activation for $k > 0 \quad (\boldsymbol{h}^{(0)} = \boldsymbol{x})$

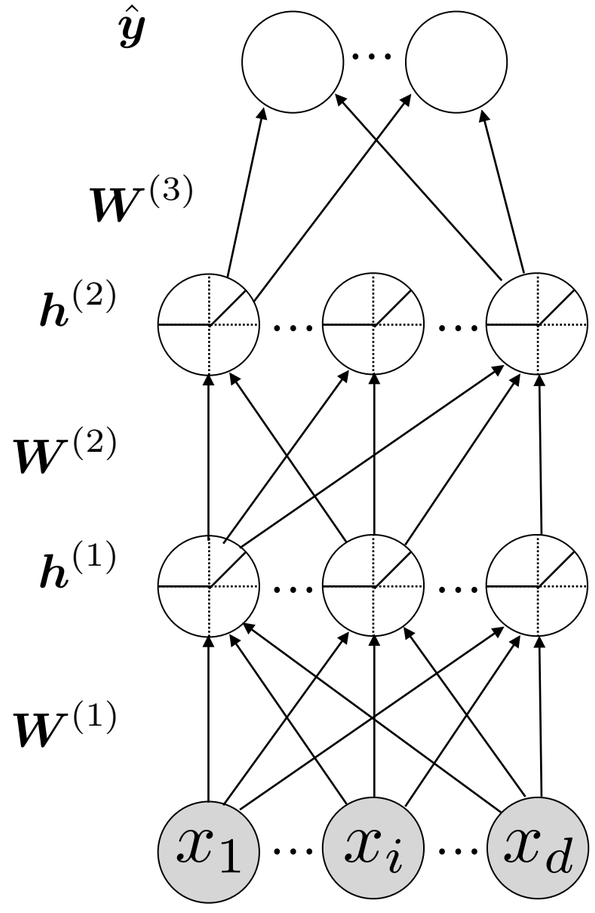
$$\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$$

- Hidden layer activation for $k = 1, \ldots, l-1$

$$\boldsymbol{h}^{(k)} = g\left(\boldsymbol{a}^{(k)}\right) \odot \boldsymbol{m}^{(k)}$$

- Output layer activation

$$\hat{\boldsymbol{y}} = o(\boldsymbol{a}^{(l)})$$



Dropout: Forward Pass

Implementation: use random binary masks $\boldsymbol{m}^{(k)}$ sampled at training time

- Layer pre-activation for $k > 0 \quad (\boldsymbol{h}^{(0)} = \boldsymbol{x})$

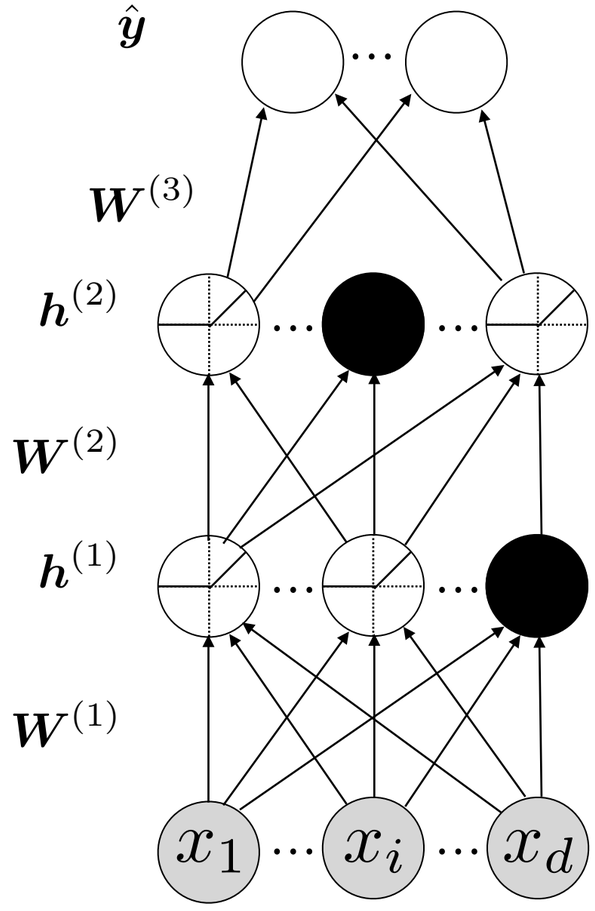
$$\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$$

- Hidden layer activation for $k = 1, \ldots, l-1$

$$\boldsymbol{h}^{(k)} = g\left(\boldsymbol{a}^{(k)}\right) \odot \boldsymbol{m}^{(k)}$$

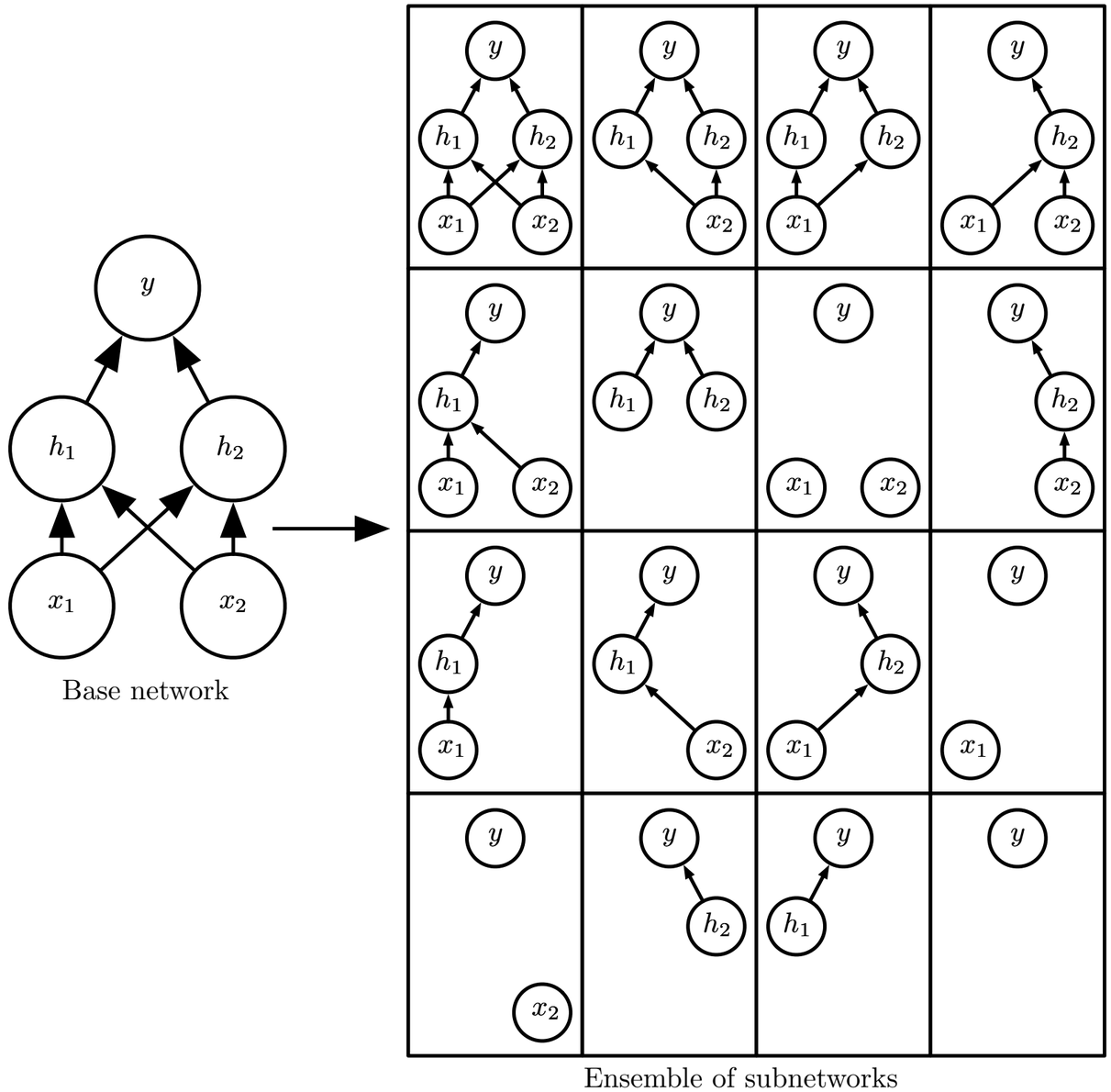
- Output layer activation

$$\hat{\boldsymbol{y}} = o(\boldsymbol{a}^{(l)})$$



Dropout: Test Time

- At test time, replace the masks by their expectation
- This is simply the constant vector 0.5 if dropout probability is 0.5
- For a single hidden layer, this is equivalent to taking the geometric mean of all neural networks, with all binary masks
- Extremely efficient ensembling



Reproduced from Goodfellow et al.

Stochastic regularization

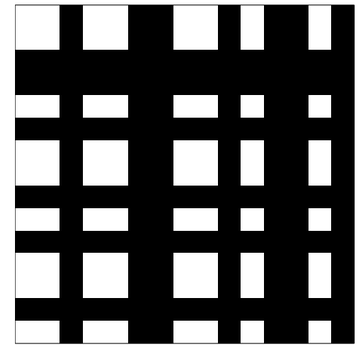
Can consider most methods as applying a mask to the weight matrix: $\boldsymbol{h}^{(k)} = f\left(\boldsymbol{W}'^{(k)}\boldsymbol{h}^{(k-1)}\right) = f\left(\left(\boldsymbol{M}^{(k)} \odot \boldsymbol{W}^{(k)}\right)\boldsymbol{h}^{(k-1)}\right)$

Stochastic regularization

Can consider most methods as applying a mask to the weight matrix: $\boldsymbol{h}^{(k)} = f\left(\boldsymbol{W}'^{(k)}\boldsymbol{h}^{(k-1)}\right) = f\left(\left(\boldsymbol{M}^{(k)} \odot \boldsymbol{W}^{(k)}\right)\boldsymbol{h}^{(k-1)}\right)$

Dropout
(Srivastava et al. 2012)

$$\boldsymbol{M}^{(k)} = \boldsymbol{m}^{(k)}\boldsymbol{m}^{(k-1)\top}$$

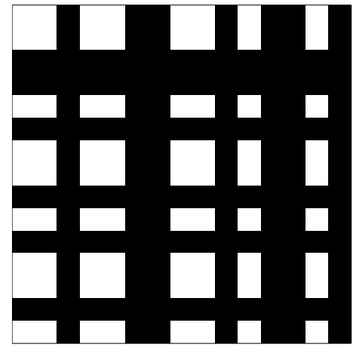


Stochastic regularization

Can consider most methods as applying a mask to the weight matrix: $\boldsymbol{h}^{(k)} = f\left(\boldsymbol{W}'^{(k)}\boldsymbol{h}^{(k-1)}\right) = f\left(\left(\boldsymbol{M}^{(k)} \odot \boldsymbol{W}^{(k)}\right)\boldsymbol{h}^{(k-1)}\right)$

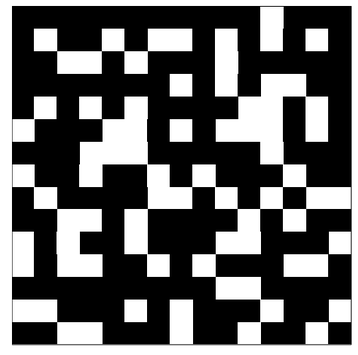
Dropout
(Srivastava et al. 2012)

$$\boldsymbol{M}^{(k)} = \boldsymbol{m}^{(k)}\boldsymbol{m}^{(k-1)\top}$$



DropConnect
(Wan et al. 2013)

$$\boldsymbol{M}^{(k)} = \left[\sim \text{Bern}(p^{(k)})\right]$$

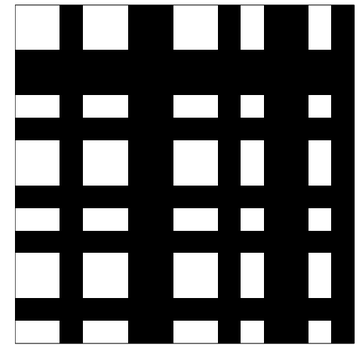


Stochastic regularization

Can consider most methods as applying a mask to the weight matrix: $\boldsymbol{h}^{(k)} = f\left(\boldsymbol{W}'^{(k)}\boldsymbol{h}^{(k-1)}\right) = f\left(\left(\boldsymbol{M}^{(k)} \odot \boldsymbol{W}^{(k)}\right)\boldsymbol{h}^{(k-1)}\right)$

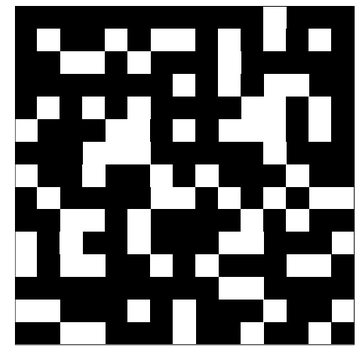
Dropout
(Srivastava et al. 2012)

$$\boldsymbol{M}^{(k)} = \boldsymbol{m}^{(k)}\boldsymbol{m}^{(k-1)\top}$$



DropConnect
(Wan et al. 2013)

$$\boldsymbol{M}^{(k)} = \left[\sim \text{Bern}(p^{(k)})\right]$$

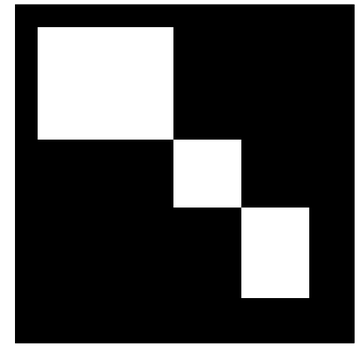


Blockout
(Murdoch et al. 2016)

$$\boldsymbol{M}^{(k)} = \frac{1}{c}\boldsymbol{C}^{(k)}\boldsymbol{C}^{(k-1)\top}$$

$$\boldsymbol{C}^{(k)} \sim \text{Bern}(\boldsymbol{P}^{(k)})$$

$$c - \#\text{ of clusters}$$

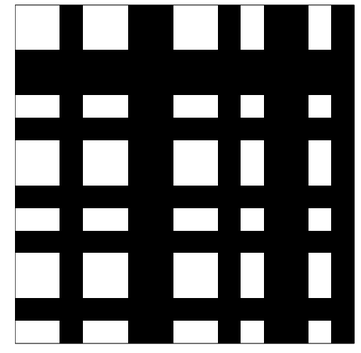


Stochastic regularization

Can consider most methods as applying a mask to the weight matrix: $\boldsymbol{h}^{(k)} = f\left(\boldsymbol{W}'^{(k)}\boldsymbol{h}^{(k-1)}\right) = f\left(\left(\boldsymbol{M}^{(k)} \odot \boldsymbol{W}^{(k)}\right)\boldsymbol{h}^{(k-1)}\right)$

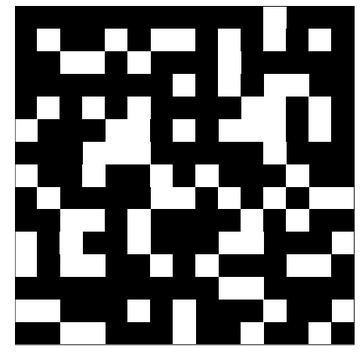
Dropout
(Srivastava et al. 2012)

$$\boldsymbol{M}^{(k)} = \boldsymbol{m}^{(k)}\boldsymbol{m}^{(k-1)\top}$$



DropConnect
(Wan et al. 2013)

$$\boldsymbol{M}^{(k)} = \left[\sim \text{Bern}(p^{(k)})\right]$$

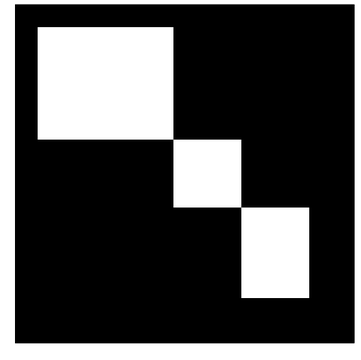


Blockout
(Murdoch et al. 2016)

$$\boldsymbol{M}^{(k)} = \frac{1}{c}\boldsymbol{C}^{(k)}\boldsymbol{C}^{(k-1)\top}$$

$$\boldsymbol{C}^{(k)} \sim \text{Bern}(\boldsymbol{P}^{(k)})$$

$$c - \#\text{ of clusters}$$



Blockout: Architecture by Masking

- By stochastically assigning units to clusters, Blockout (Murdoch et al. 2016) permit a “library” of architectural components:

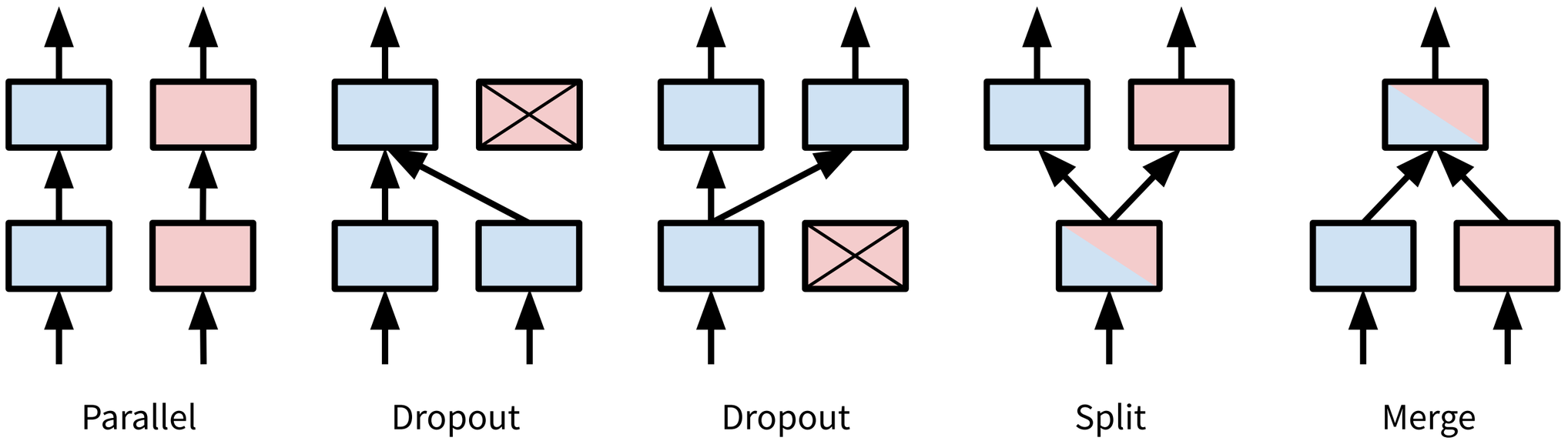


Image Credit: Murdoch et al. (2016)