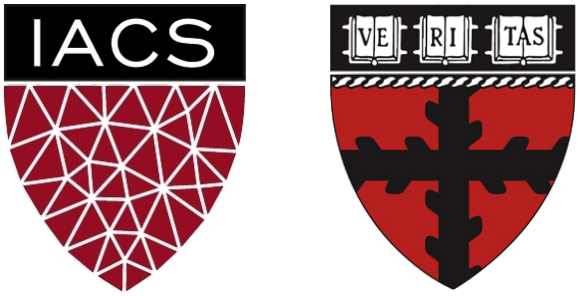
Lecture 14: Regularization

CS 109B, STAT 121B, AC 209B, CSE 109B

Mark Glickman and Pavlos Protopapas



Lecture 3
Regularization

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

Outline

- Norm Penalties
- Early Stopping
- Data Augmentation
- Bagging
- Dropout

Norm Penalties

- Optimize:

$$J(\theta;\, X, y) \;+\; \alpha\, \Omega(\theta)$$

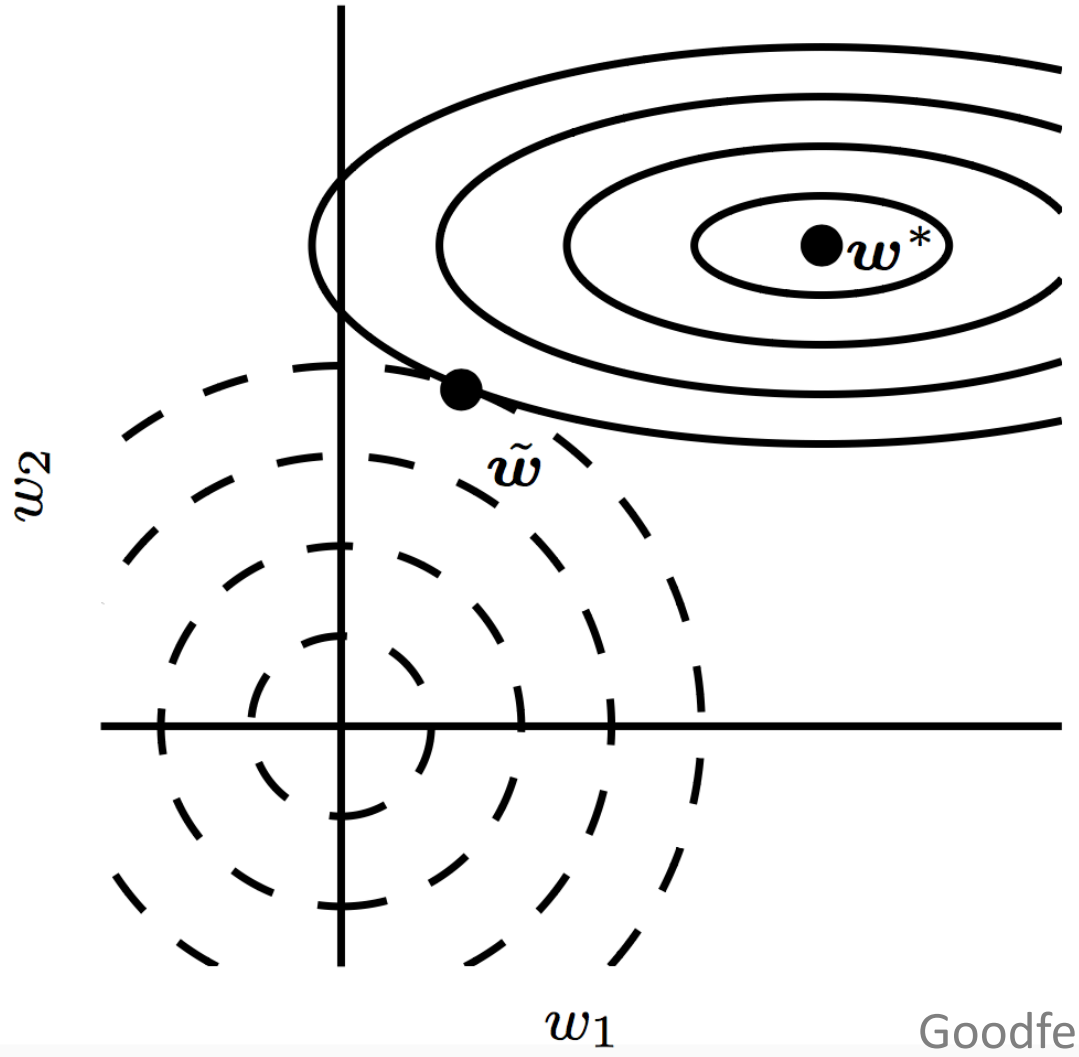
- L_2 regularization:
  - decays weights
  - MAP estimation with Gaussian prior

$$\Omega(\theta) = \frac{1}{2}\left\|\mathbf{w}\right\|_2^2$$

- L_1 regularization:
  - encourages sparsity
  - MAP estimation with Laplacian prior

$$\Omega(\theta) = \left\|\mathbf{w}\right\|_1$$

L_2 Regularization



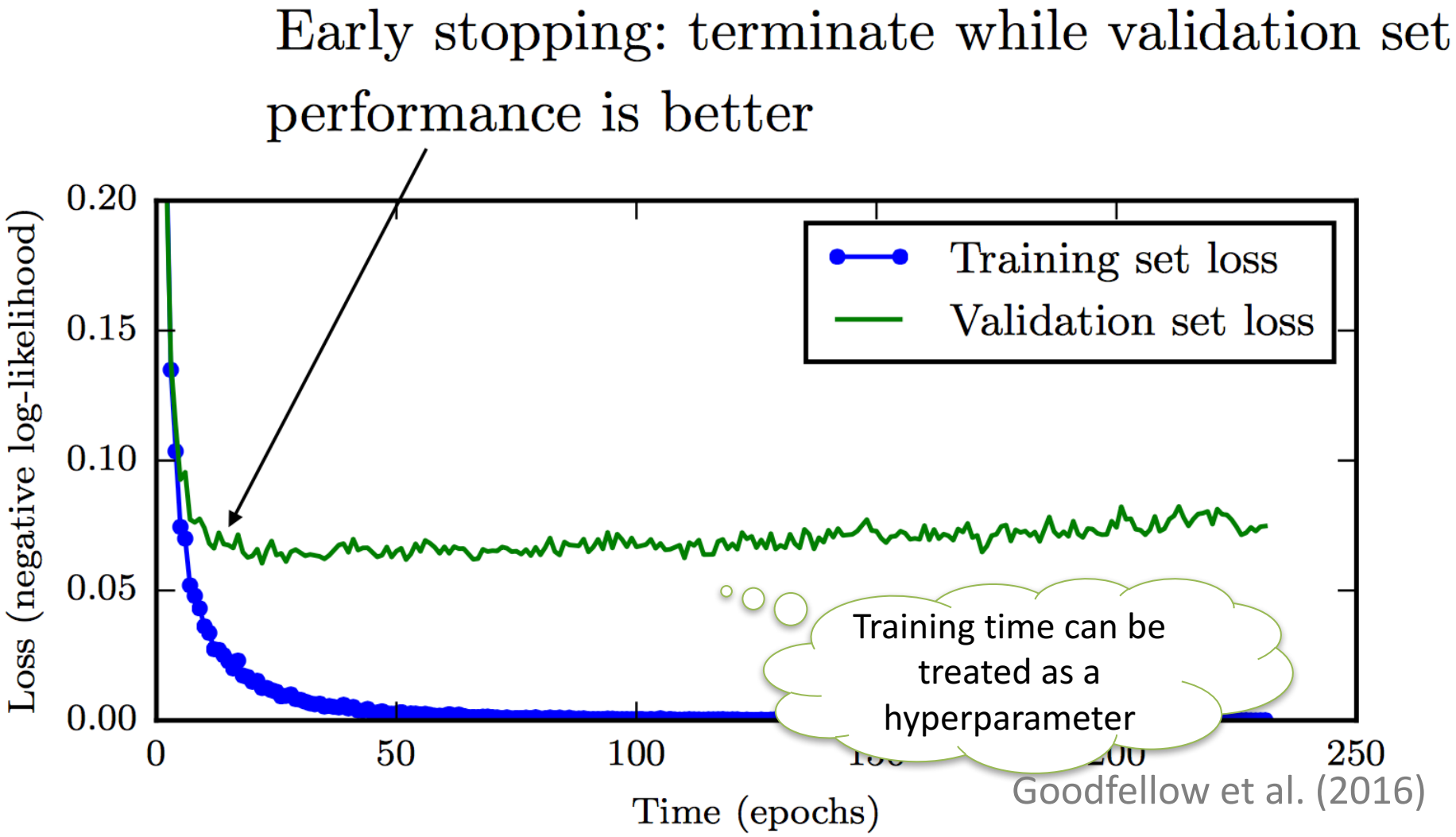
Goodfellow et al. (2016)

Norm Penalties as Constraints

$$\min_{\Omega(\theta) \leq K} J(\theta;\, X, y)$$

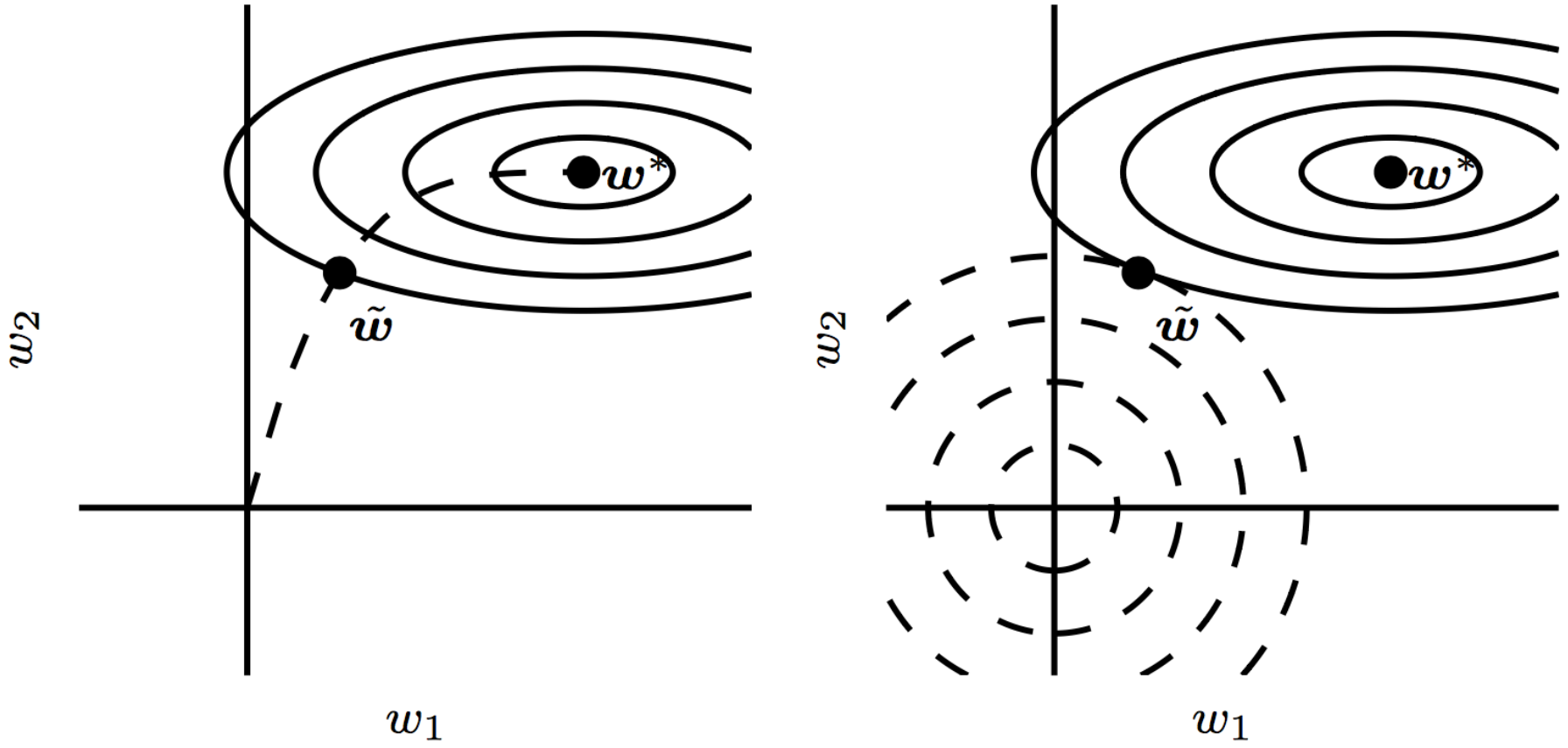
- Useful if *K* is known in advance
- Optimization:
  - Construct Lagrangian and apply gradient descent
  - Projected gradient descent

Early Stopping



Goodfellow et al. (2016)

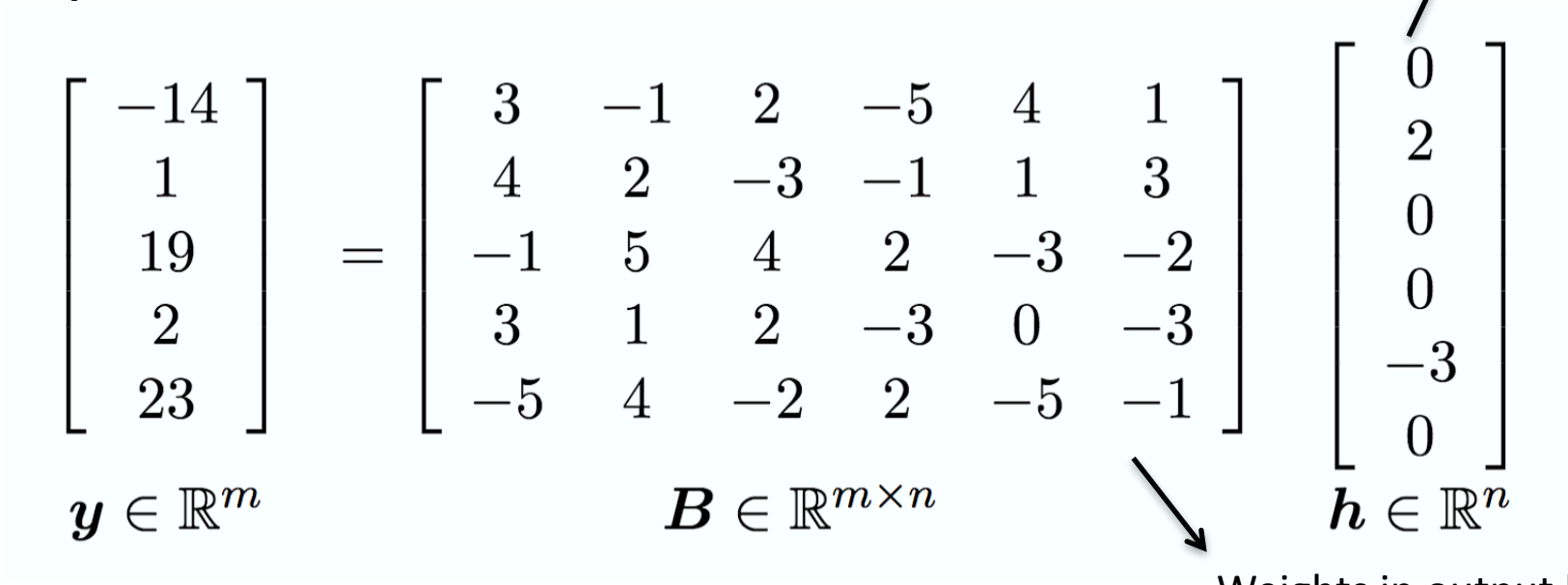
Early Stopping ≈ Weight Decay



Goodfellow et al. (2016)

Sparse Representations

- Weight decay *on activations* instead of parameters



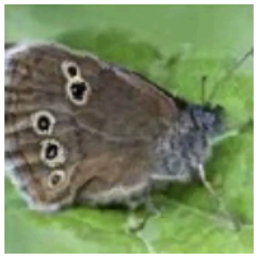
$$J(\theta;\, X, y) \;+\; \alpha\, \Omega(h)$$

Data Augmentation

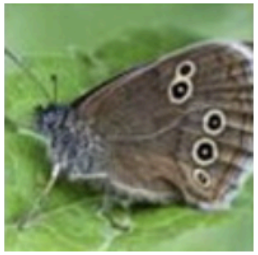
Affine Distortion

Noise

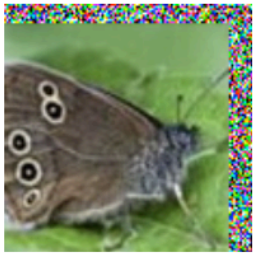
Elastic Deformation



Horizontal flip



Random Translation

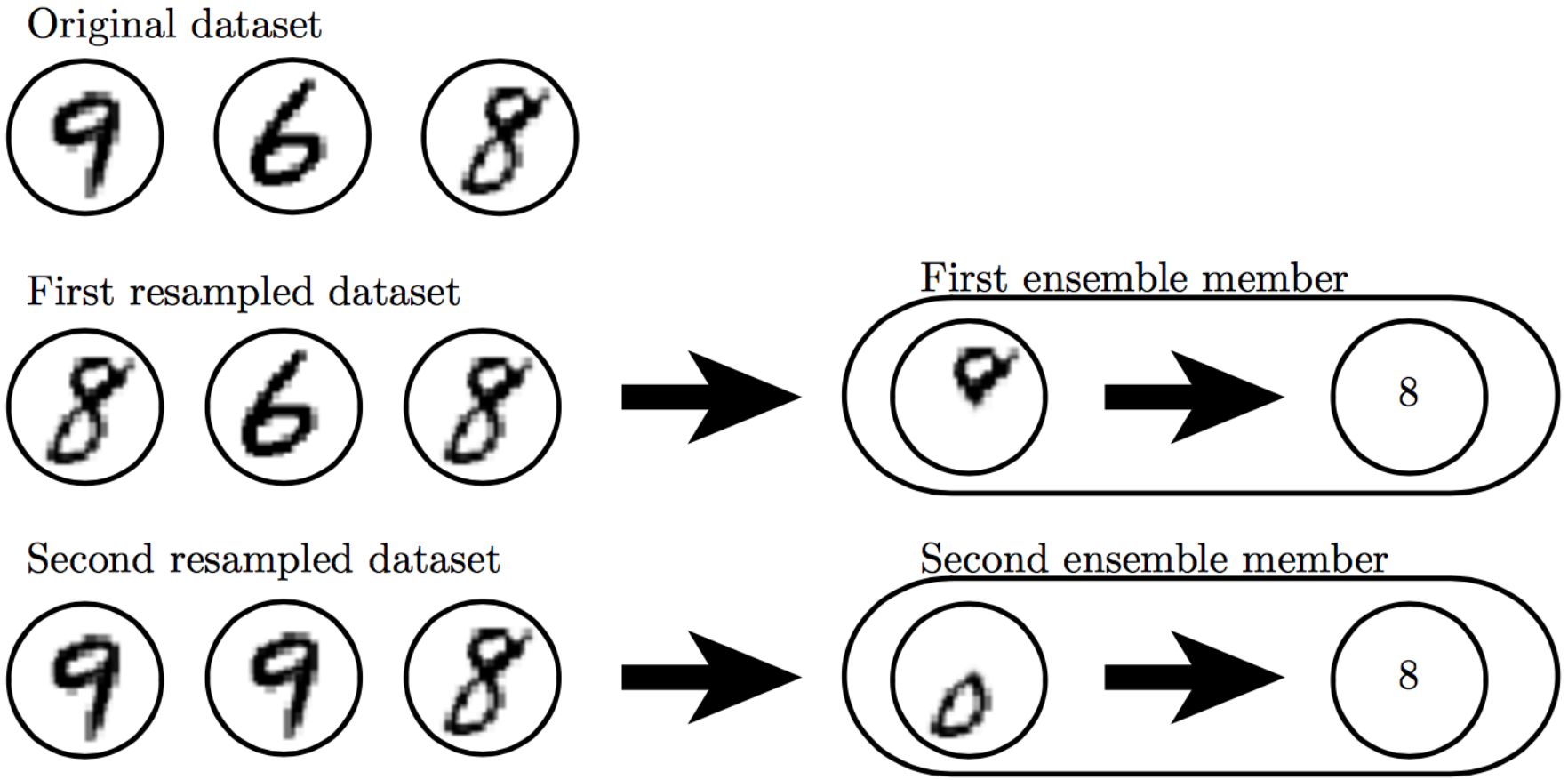


Hue Shift

Noise Robustness

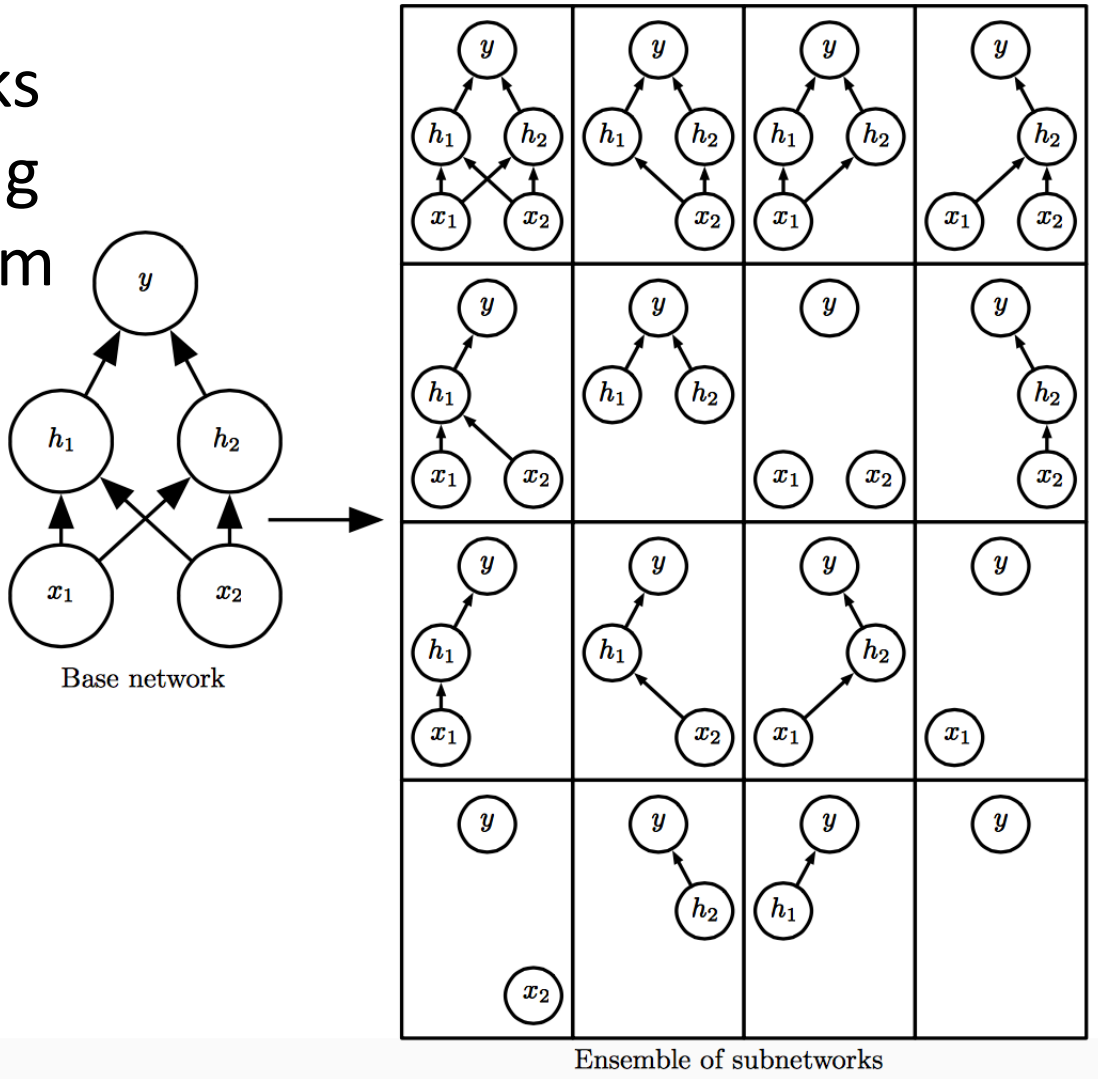
- Random perturbation of network weights
  - Gaussian noise: Equivalent to minimizing loss with regularization term $\mathbf{E}\left[\left\|\nabla_w y(x)\right\|\right]$
  - Encourages smooth function: small perturbation in weights leads to small changes in output
- Injecting noise in output labels
  - Better convergence: prevents pursuit of hard probabilities

Bagging


Original dataset
First resampled dataset
First ensemble member
8
Second resampled dataset
Second ensemble member
8

Dropout

Train all sub-networks obtained by removing non-output units from base network



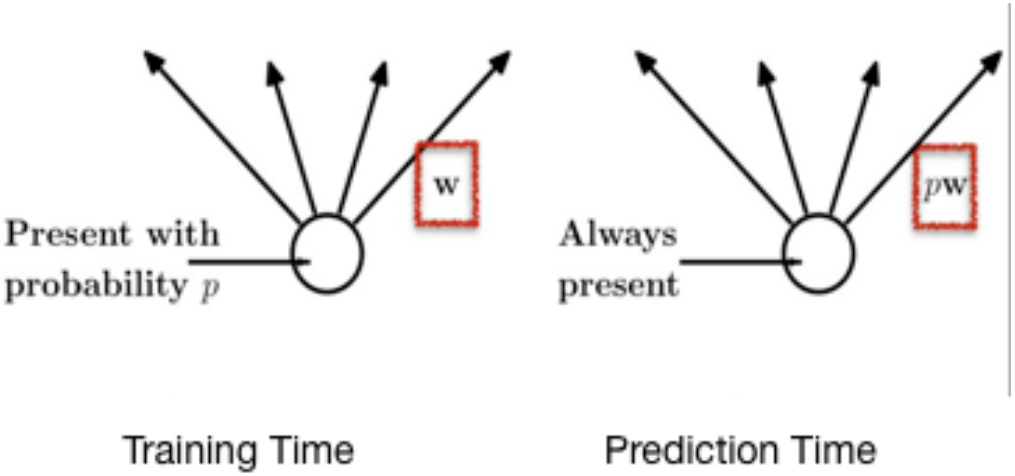
Goodfellow et al. (2016)

Dropout: Stochastic GD

- For each new example/mini-batch:
  - Randomly sample a binary mask μ independently, where μ_i indicates if input/hidden node i is included
  - Multiply output of node i with μ_i, and perform gradient update
- Typically, an input node is included with prob.0.8, hidden node with prob. 0.5

Dropout: Weight Scaling

- During prediction time use all units, but scale weights with probability of inclusion



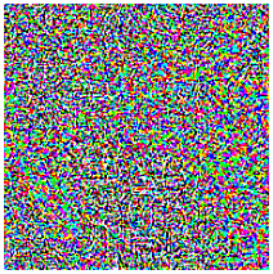
- Approximates the following inference rule:

$$\tilde{p}_{\text{ensemble}}(y \mid \boldsymbol{x}) = \sqrt[2^d]{\prod_{\boldsymbol{\mu}} p(y \mid \boldsymbol{x}, \boldsymbol{\mu})}$$

Cristina Scheau (2016)

Adversarial Examples

$+ .007 \times$



$=$

$\boldsymbol{x}$

$y =$"panda"
w/ 57.7%
confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

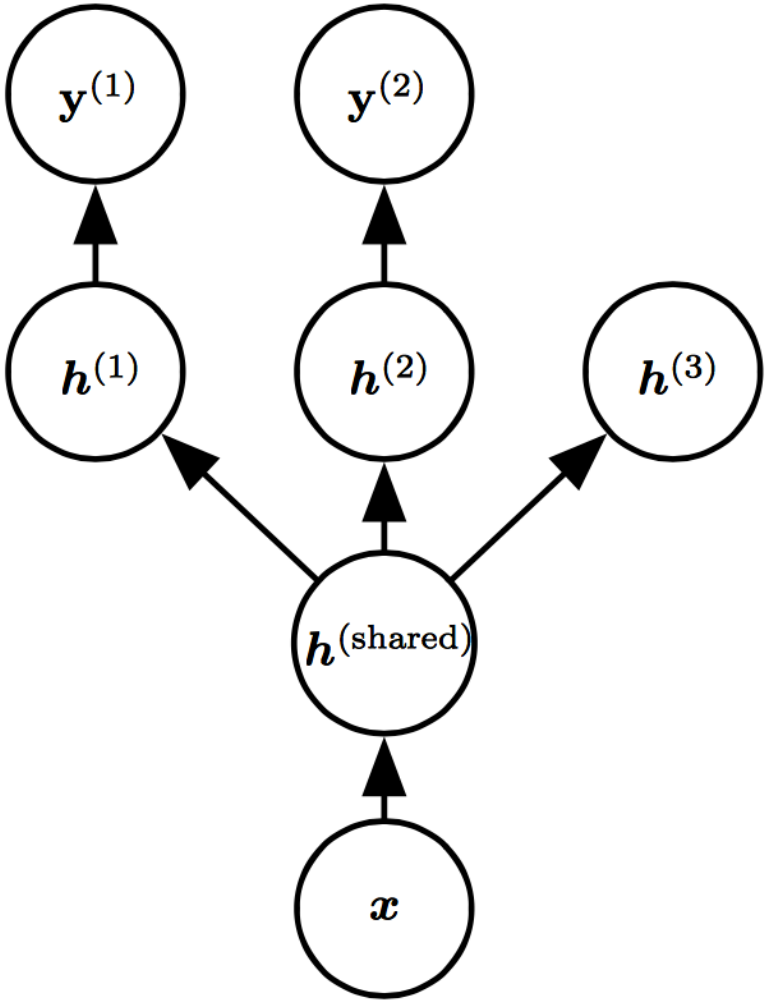
"nematode"
w/ 8.2%
confidence

$\boldsymbol{x} +$
$\epsilon \, \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$

"gibbon"
w/ 99.3 %
confidence

Training on adversarial examples is mostly intended to improve security, but can sometimes provide generic regularization.

Multi-task Learning



Goodfellow et al. (2016)