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:

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

Biases not penalized

Don't penalize the bias:
1. won't increase too much model complexity
2. regularized bias tend to underfit

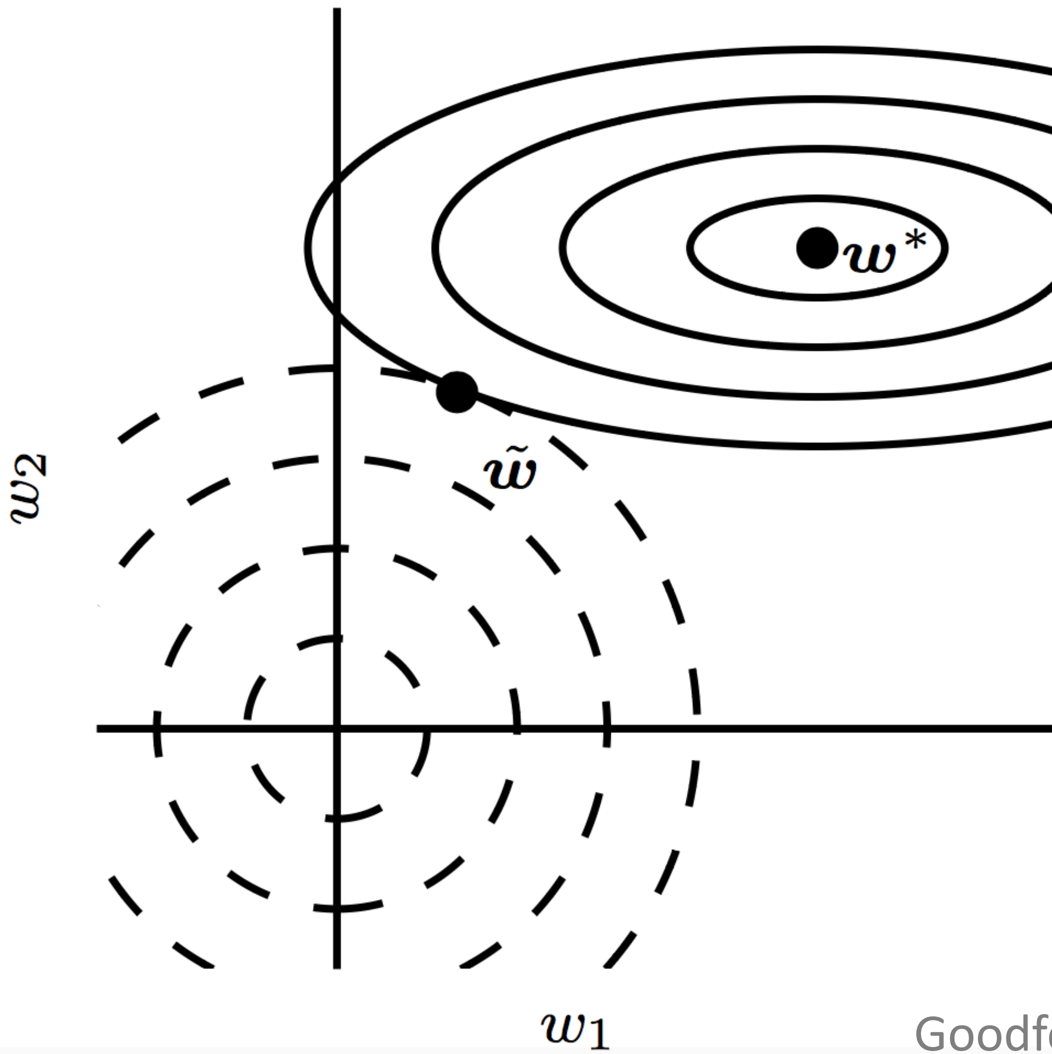
- L_2 regularization:
  - decays weights
  - MAP estimation with Gaussian prior

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

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

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

L_2 Regularization



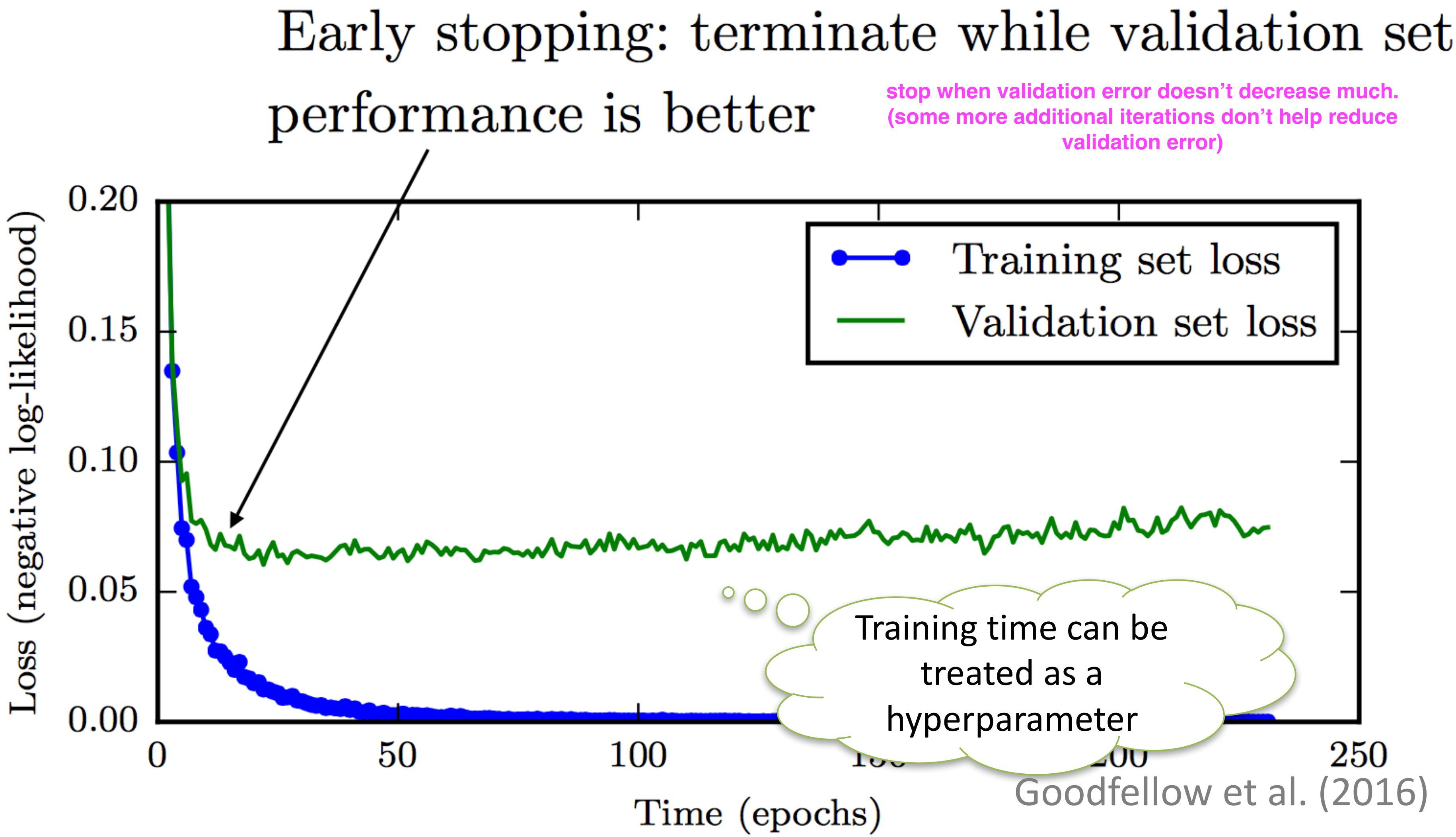
Goodfellow et al. (2016)

Norm Penalties as Constraints

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

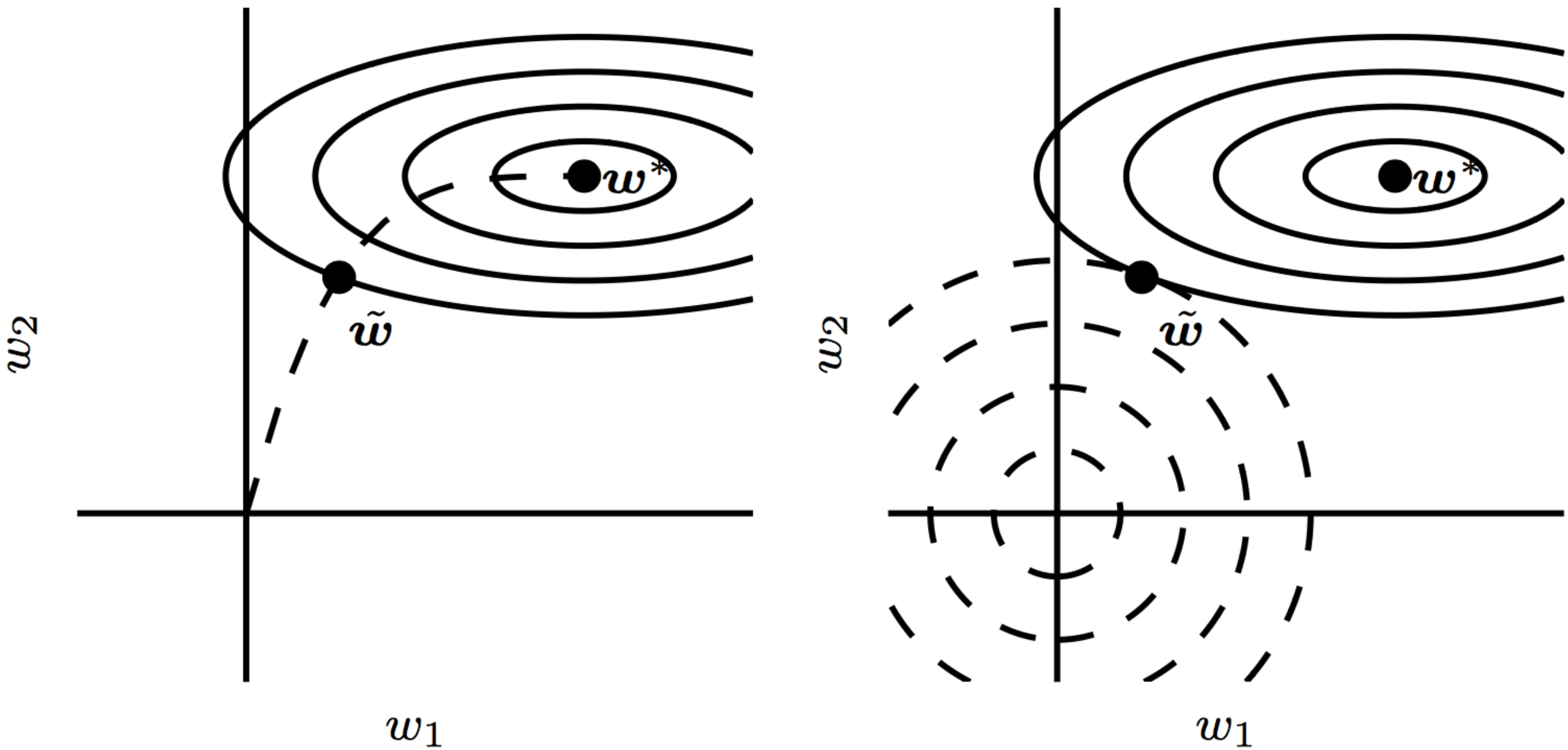
- Useful if *K* is known in advance
- Optimization:
  - Construct Lagrangian and apply gradient descent
  - Projected gradient descent

Early Stopping



Early Stopping ≈ Weight Decay

applies to:
1. logistic regression
2. boosting



Goodfellow et al. (2016)

Sparse Representations

- Weight decay *on activations* instead of parameters

Output of hidden layer

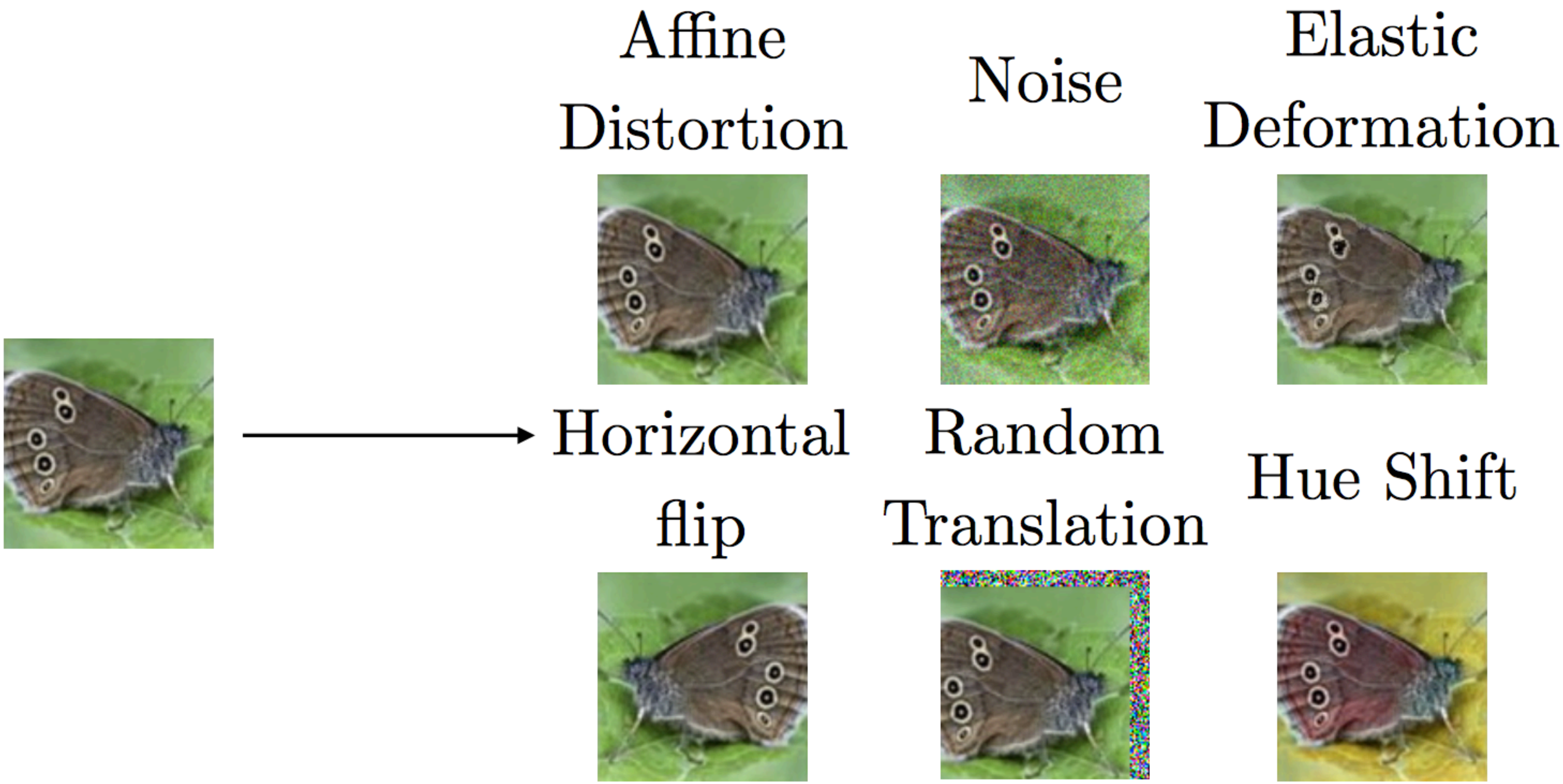
$$\begin{bmatrix} -14 \\ 1 \\ 19 \\ 2 \\ 23 \end{bmatrix} = \begin{bmatrix} 3 & -1 & 2 & -5 & 4 & 1 \\ 4 & 2 & -3 & -1 & 1 & 3 \\ -1 & 5 & 4 & 2 & -3 & -2 \\ 3 & 1 & 2 & -3 & 0 & -3 \\ -5 & 4 & -2 & 2 & -5 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \\ -3 \\ 0 \end{bmatrix}$$

$\boldsymbol{y} \in \mathbb{R}^m$ $\quad$ $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ $\quad$ $\boldsymbol{h} \in \mathbb{R}^n$

Weights in output layer

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

Data Augmentation

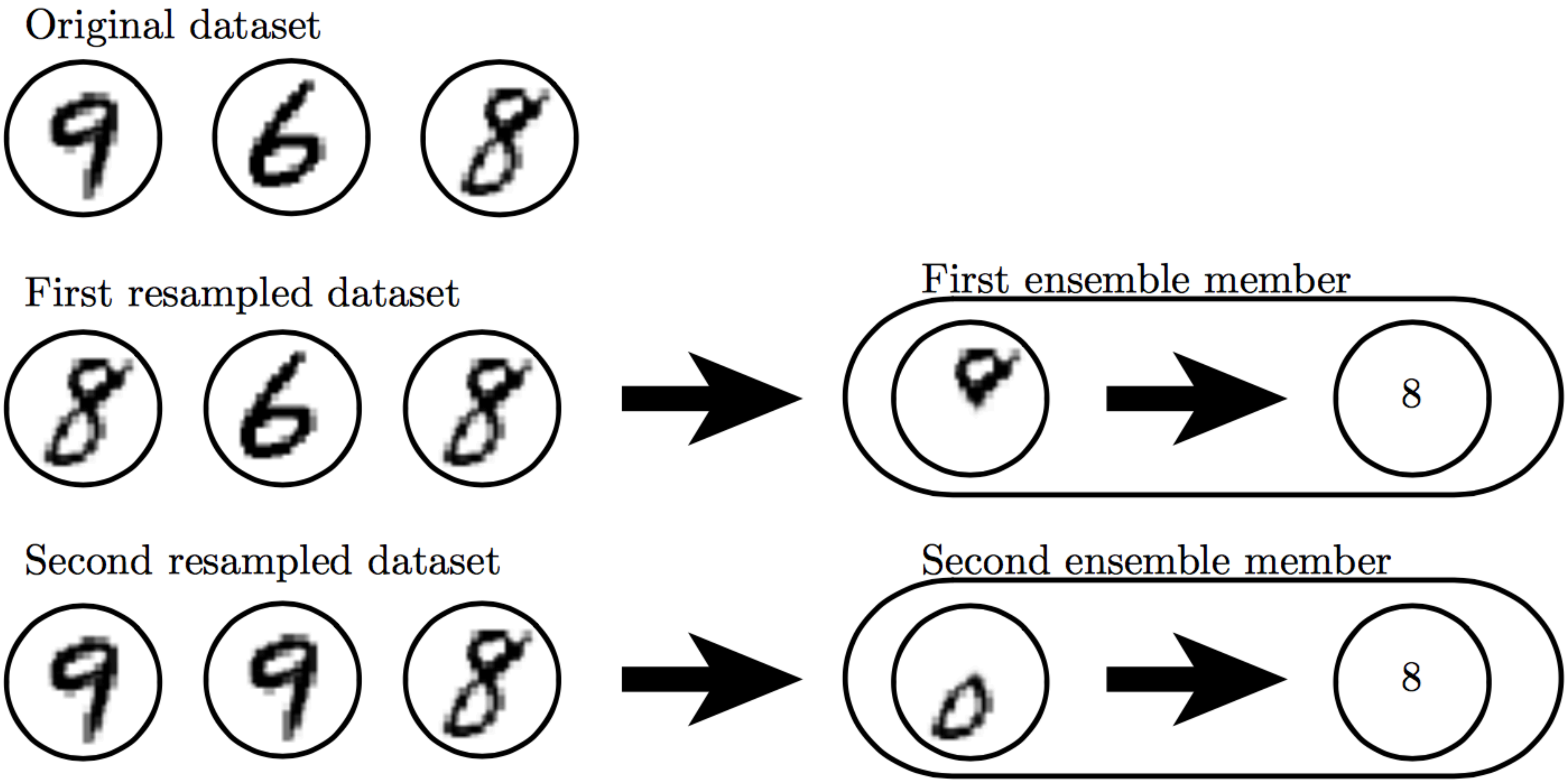


deeplearningbook.org

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

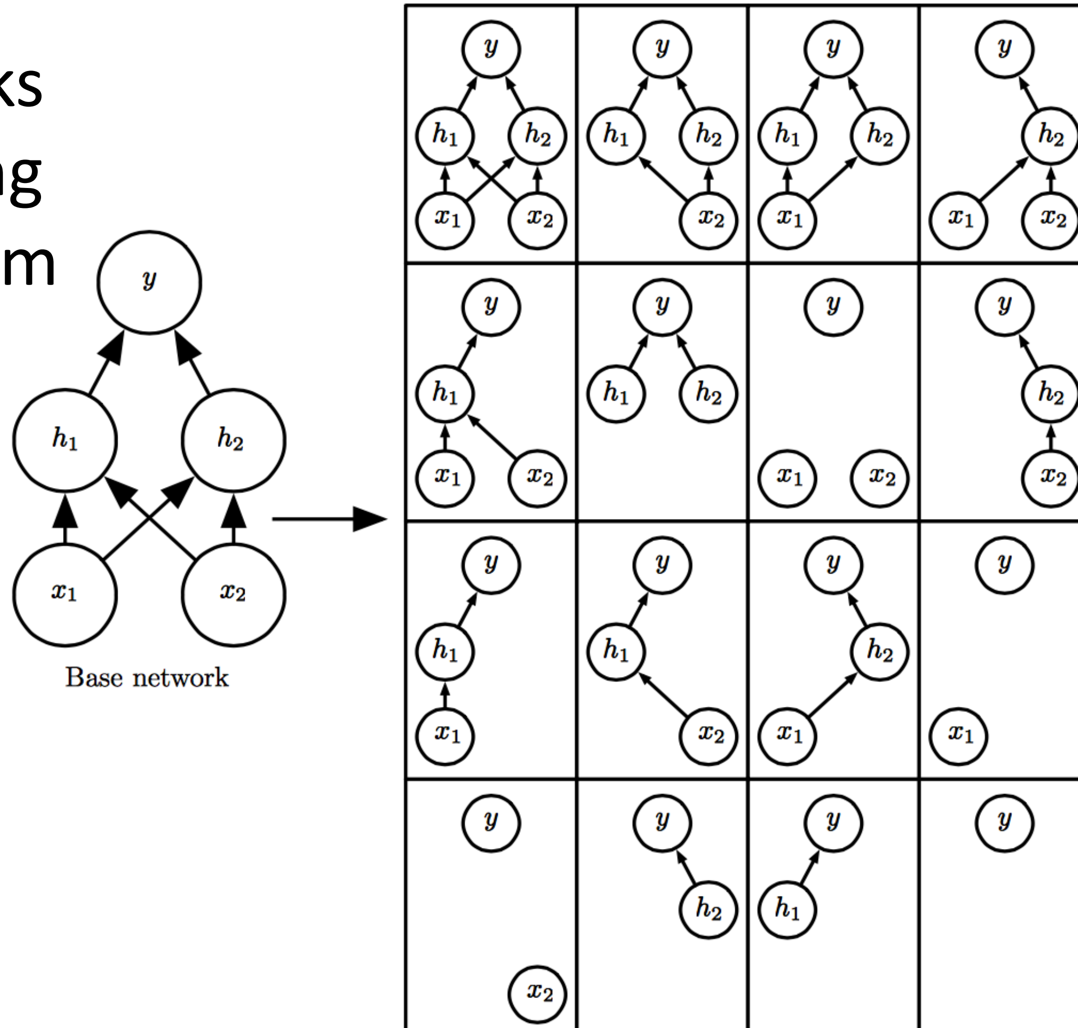
Bagging



deeplearningbook.org

Dropout

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



Ensemble of subnetworks

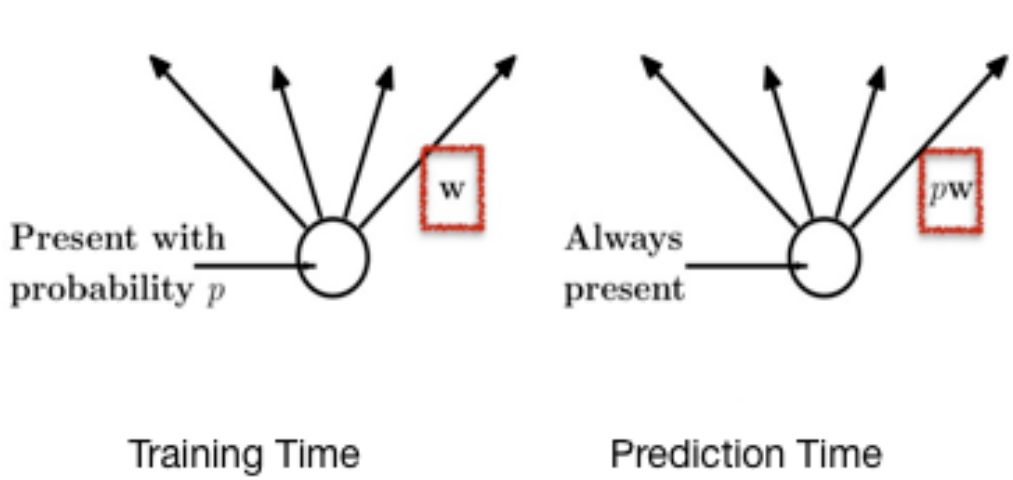
Goodfellow et al. (2016)

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

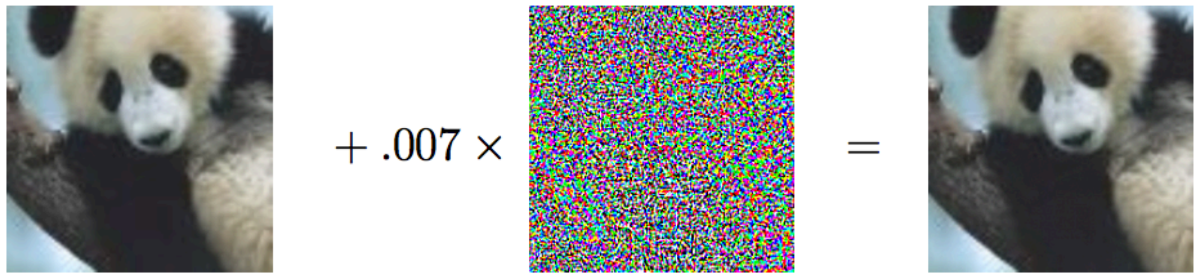


- Approximates the following inference rule:

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

Cristina Scheau (2016)

Adversarial Examples



$$\boldsymbol{x} + .007 \times \mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)) = \boldsymbol{x} + \epsilon\,\mathrm{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$$

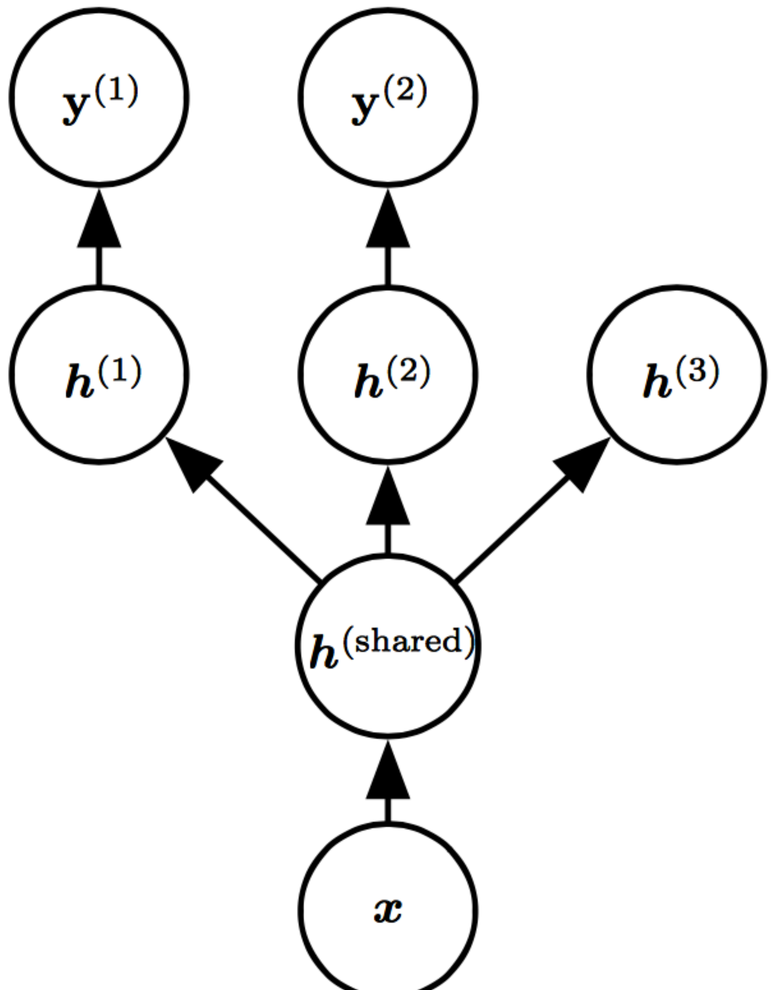
$y =$"panda" w/ 57.7% confidence

"nematode" w/ 8.2% confidence

"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)