Deep Learning Basics
Lecture 4: Regularization II

Princeton University COS 495

Instructor: Yingyu Liang

Review

Regularization as hard constraint

- Constrained optimization

$$\min_{\theta} \hat{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i)$$

$$\text{subject to: } R(\theta) \leq r$$

Regularization as soft constraint

- Unconstrained optimization

$$\min_{\theta} \hat{L}_R(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(\theta, x_i, y_i) + \lambda R(\theta)$$

for some regularization parameter $\lambda > 0$

Regularization as Bayesian prior

- Bayesian rule:

$$p(\theta \mid \{x_i, y_i\}) = \frac{p(\theta)p(\{x_i, y_i\}|\theta)}{p(\{x_i, y_i\})}$$

- Maximum A Posteriori (MAP):

$$\max_{\theta} \log p(\theta \mid \{x_i, y_i\}) = \max_{\theta} \underbrace{\log p(\theta)}_{\text{Regularization}} + \underbrace{\log p(\{x_i, y_i\} \mid \theta)}_{\text{MLE loss}}$$

Classical regularizations

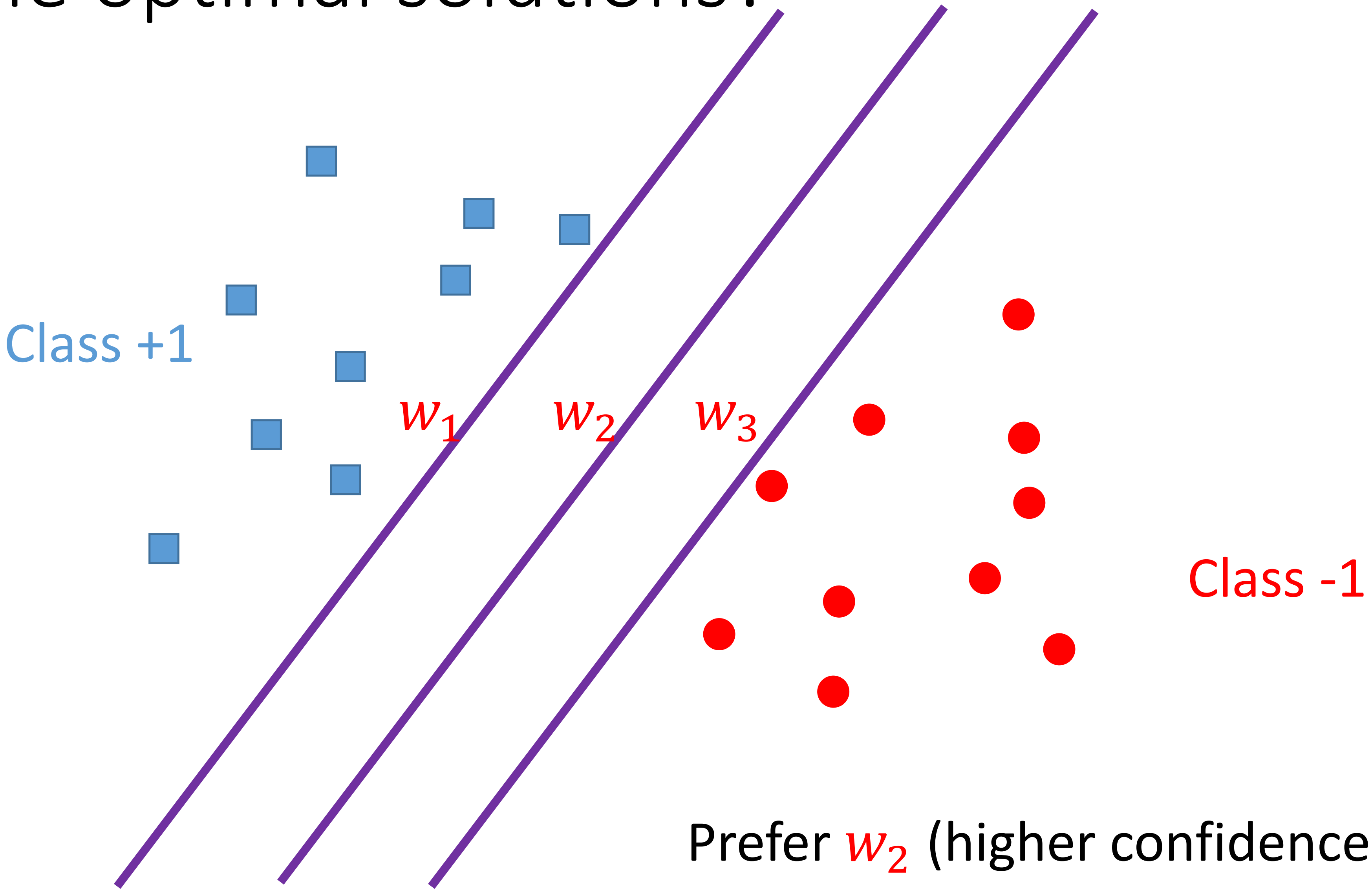
- Norm penalty
  - l_2 regularization
  - l_1 regularization

More examples

Other types of regularizations

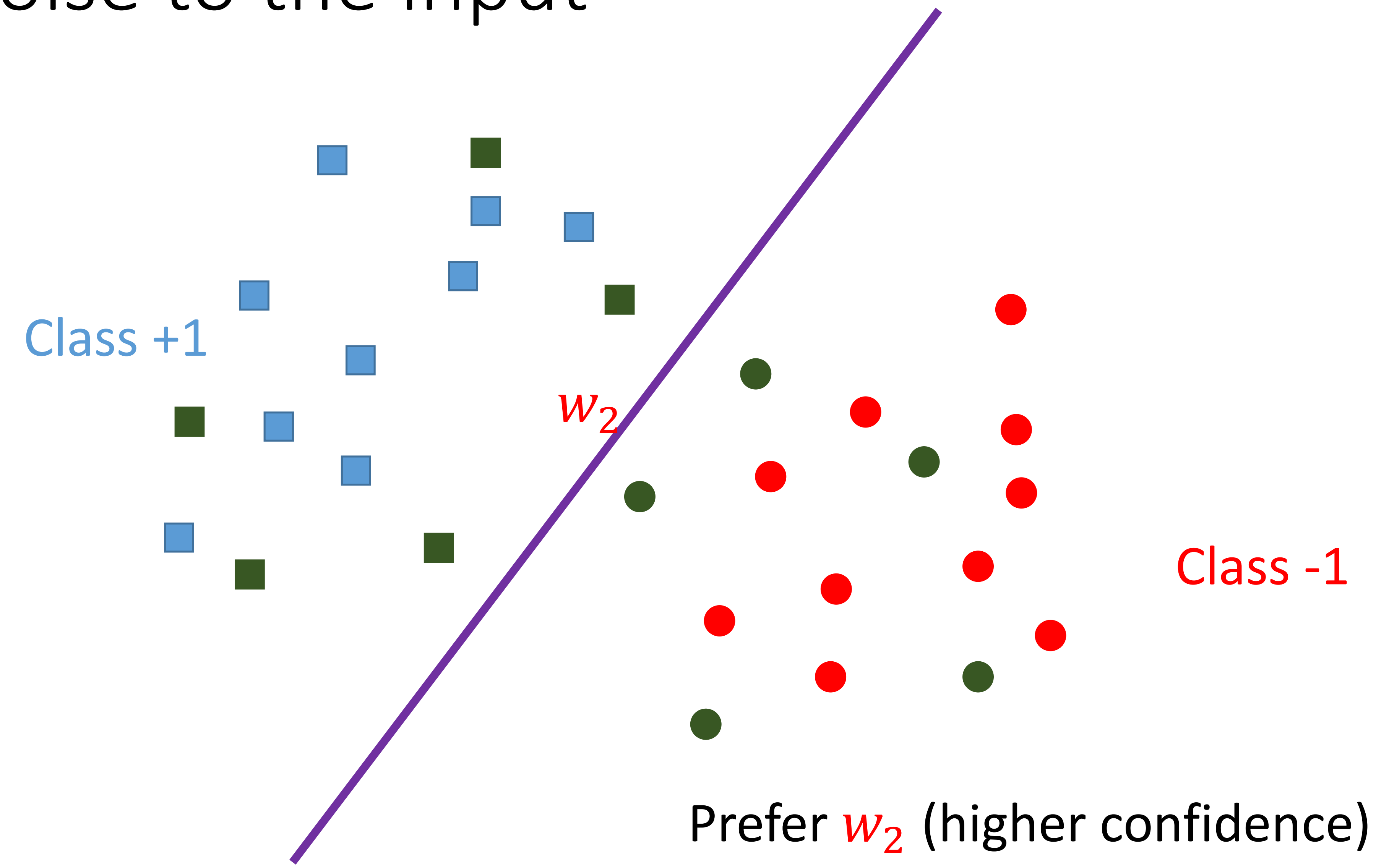
- Robustness to noise
  - Noise to the input
  - Noise to the weights
  - Noise to the output
- Data augmentation
- Early stopping
- Dropout

Multiple optimal solutions?

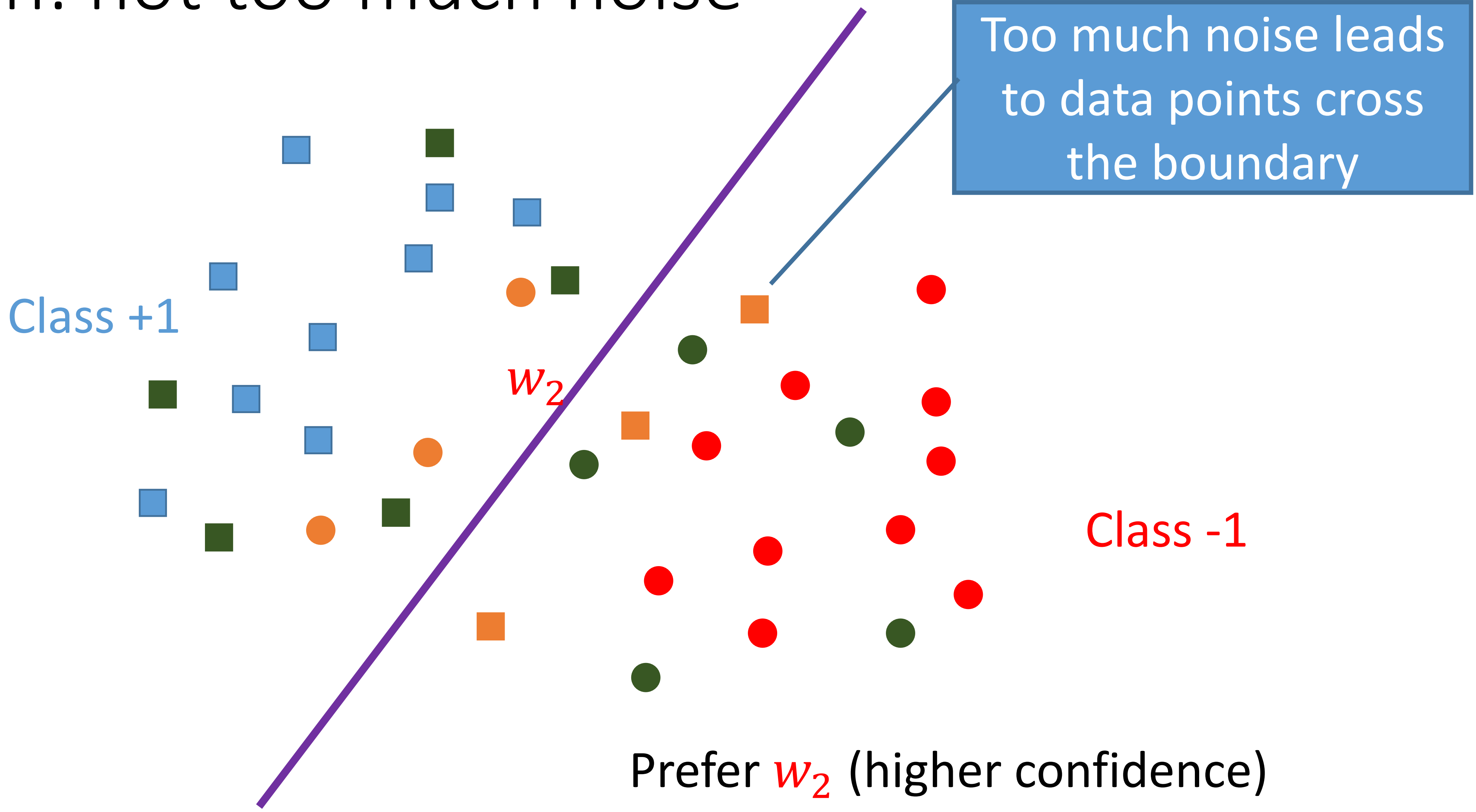


Prefer w_2 (higher confidence)

Add noise to the input



Caution: not too much noise



Equivalence to weight decay

- Suppose the hypothesis is $f(x) = w^T x$, noise is $\epsilon \sim N(0, \lambda I)$
- After adding noise, the loss is

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x+\epsilon) - y]^2 = \mathbb{E}_{x,y,\epsilon}[f(x) + w^T\epsilon - y]^2$$

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x) - y]^2 + 2\mathbb{E}_{x,y,\epsilon}[w^T\epsilon(f(x) - y)] + \mathbb{E}_{x,y,\epsilon}[w^T\epsilon]^2$$

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f(x) - y]^2 + \lambda||w||^2$$

Add noise to the weights

- For the loss on each data point, add a noise term to the weights before computing the prediction

$$\epsilon \sim N(0, \eta I), w' = w + \epsilon$$

- Prediction: $f_{w'}(x)$ instead of $f_w(x)$
- Loss becomes

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f_{w+\epsilon}(x) - y]^2$$

Add noise to the weights

- Loss becomes

$$L(f) = \mathbb{E}_{x,y,\epsilon}[f_{w+\epsilon}(x) - y]^2$$

- To simplify, use Taylor expansion
- $f_{w+\epsilon}(x) \approx f_w(x) + \epsilon^T \nabla f(x) + \frac{\epsilon^T \nabla^2 f(x) \epsilon}{2}$
- Plug in
- $L(f) \approx \mathbb{E}[f_w(x) - y]^2 + \underbrace{\eta\mathbb{E}[(f_w(x) - y)\nabla^2 f_w(x)]}_{\text{Small so can be ignored}} + \underbrace{\eta\mathbb{E}||\nabla f_w(x)||^2}_{\text{Regularization term}}$

Data augmentation

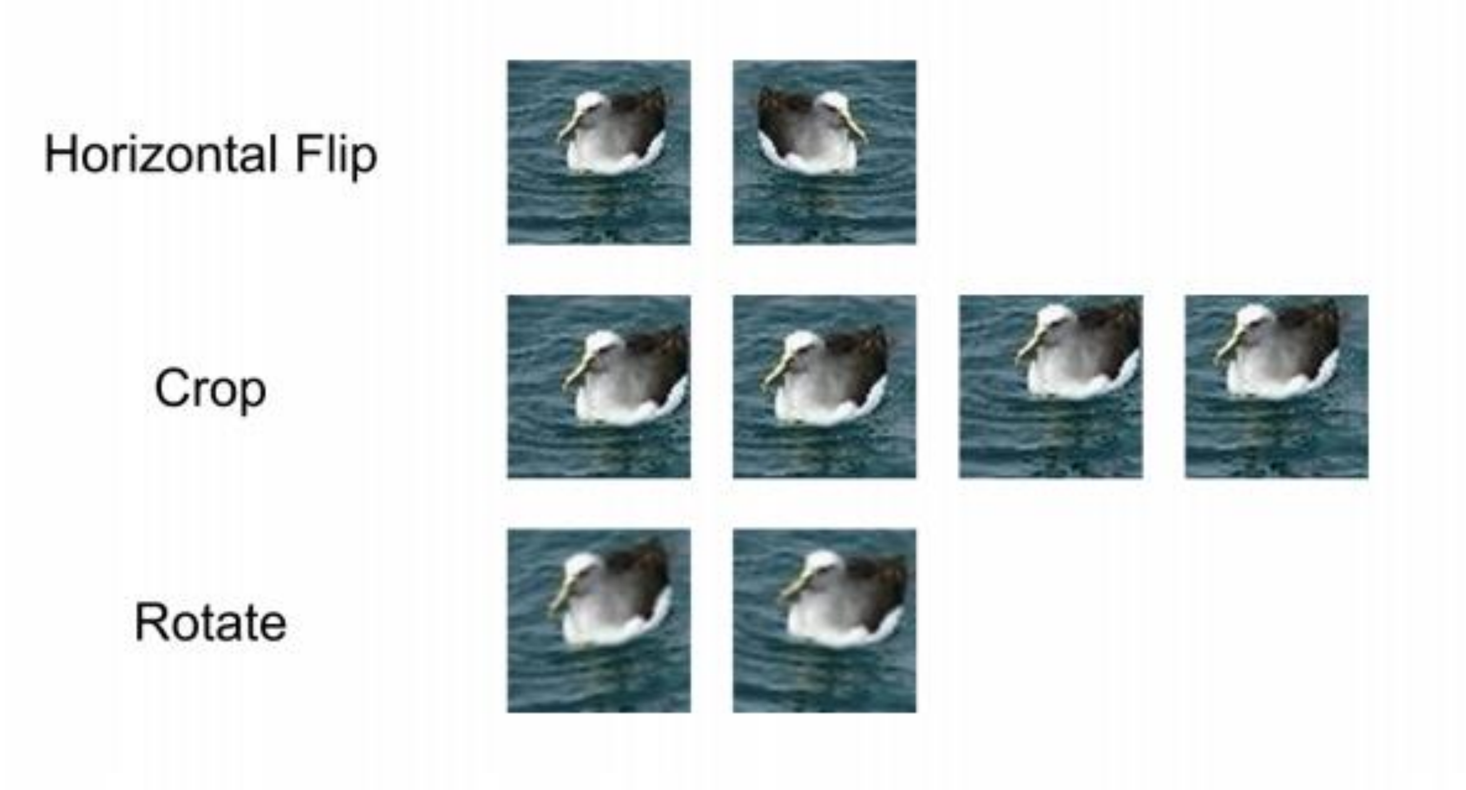


Figure from *Image Classification with Pyramid Representation and Rotated Data Augmentation on Torch 7,* by Keven Wang

Data augmentation

- Adding noise to the input: a special kind of augmentation

- Be careful about the transformation applied:
  - Example: classifying ‘b’ and ‘d’
  - Example: classifying ‘6’ and ‘9’

Early stopping

- Idea: don't train the network to too small training error

- Recall overfitting: Larger the hypothesis class, easier to find a hypothesis that fits the difference between the two

- Prevent overfitting: do not push the hypothesis too much; use validation error to decide when to stop

Early stopping

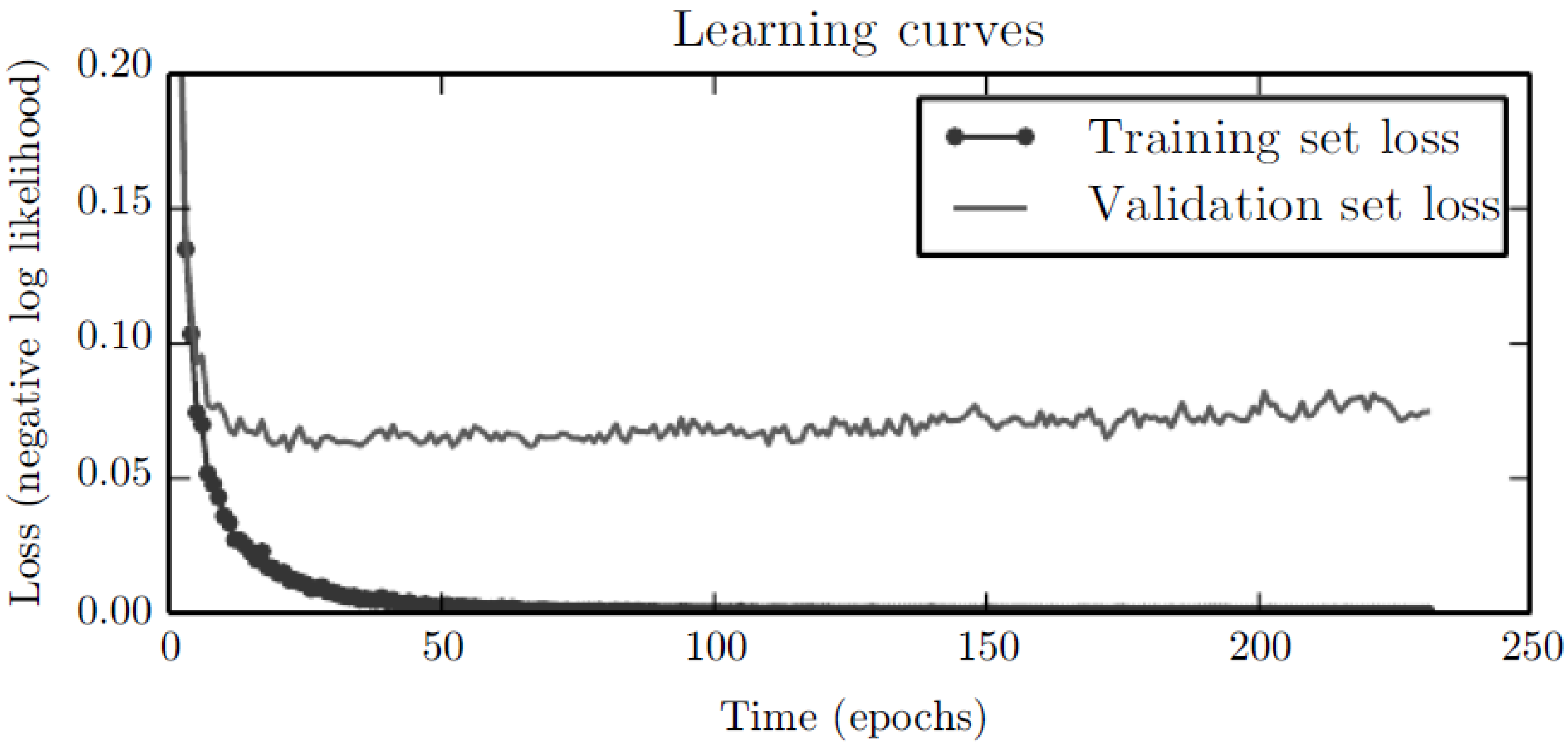


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

Early stopping

- When training, also output validation error
- Every time validation error improved, store a copy of the weights
- When validation error not improved for some time, stop
- Return the copy of the weights stored

Early stopping

- hyperparameter selection: training step is the hyperparameter

- Advantage
  - Efficient: along with training; only store an extra copy of weights
  - Simple: no change to the model/algo

- Disadvantage: need validation data

Early stopping

- Strategy to get rid of the disadvantage
  - After early stopping of the first run, train a second run and reuse validation data

- How to reuse validation data
  1. Start fresh, train with both training data and validation data up to the previous number of epochs
  2. Start from the weights in the first run, train with both training data and validation data util the validation loss < the training loss at the early stopping point

Early stopping as a regularizer

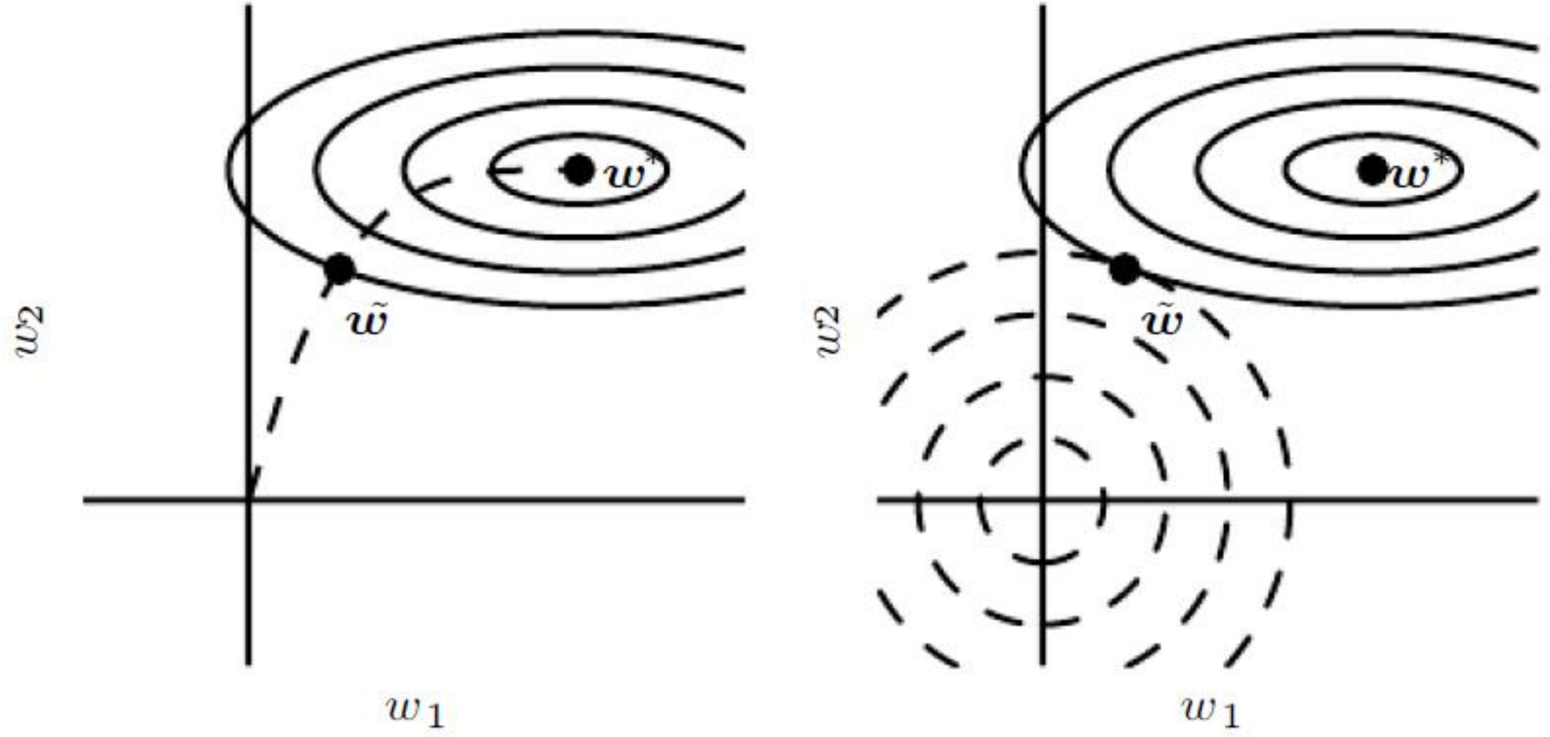


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Dropout

- Randomly select weights to update

- More precisely, in each update step
  - Randomly sample a different binary mask to all the input and hidden units
  - Multiple the mask bits with the units and do the update as usual

- Typical dropout probability: 0.2 for input and 0.5 for hidden units

Dropout

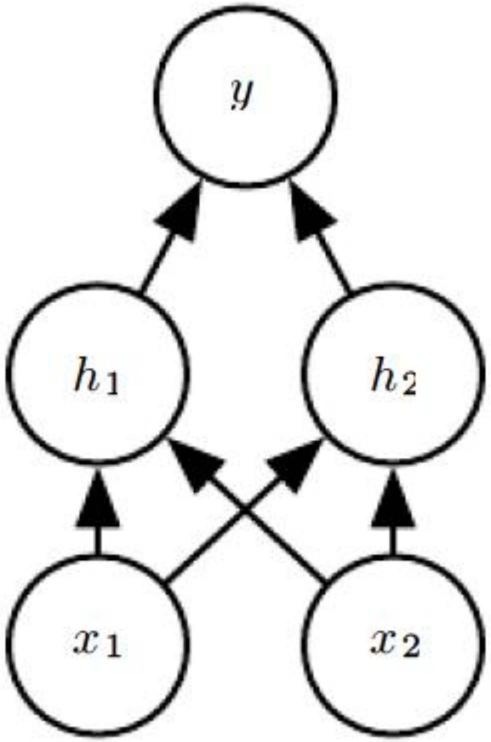


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

Dropout

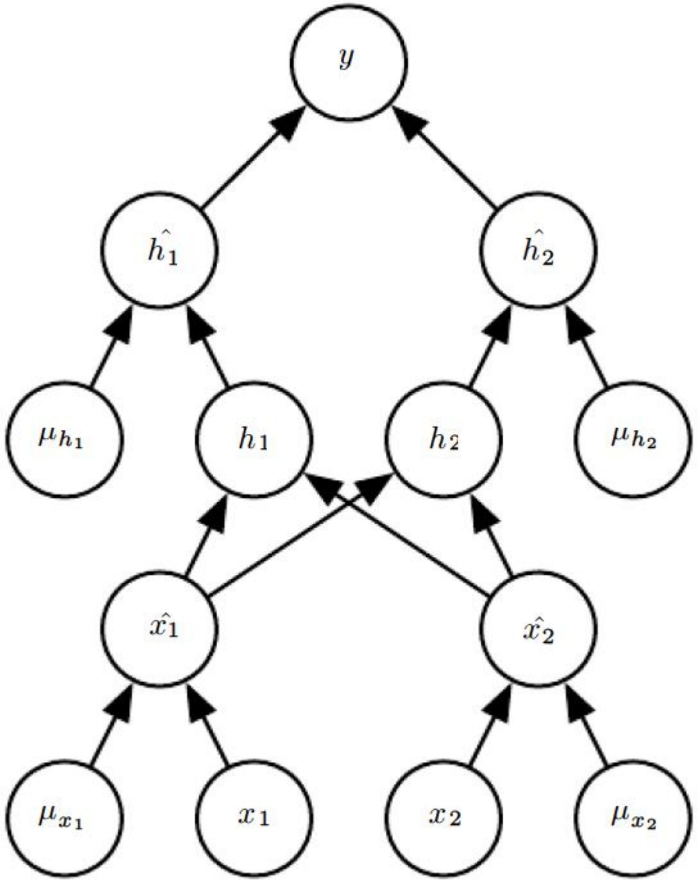


Figure from *Deep Learning*, Goodfellow, Bengio and Courville

Dropout

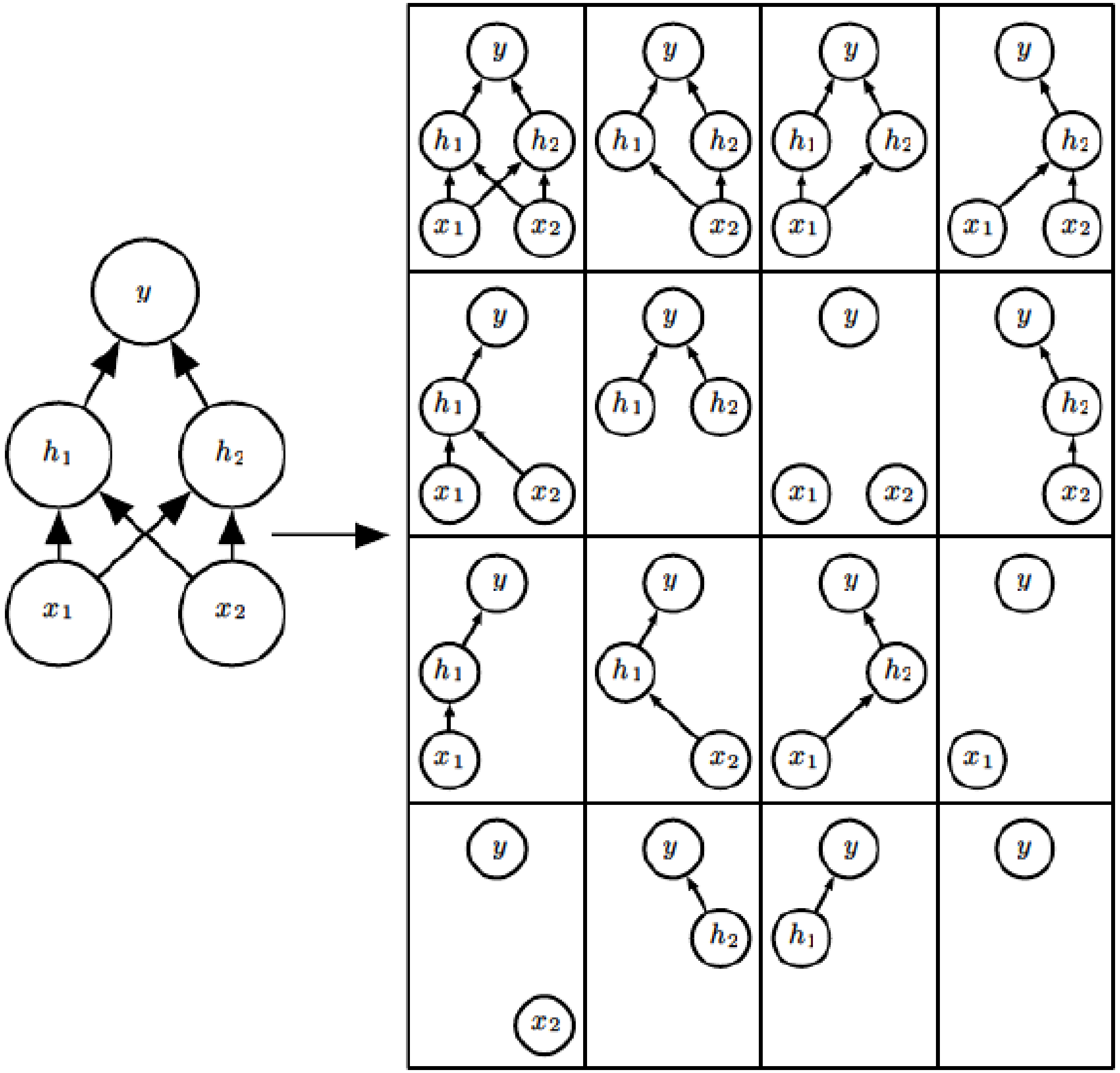


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

What regularizations are frequently used?

- l_2 regularization
- Early stopping
- Dropout

- Data augmentation if the transformations known/easy to implement