

Neural Network Part 2: Regularization

CS 760@UW-Madison





Goals for the lecture

you should understand the following concepts

- regularization
- different views of regularization
- norm constraint
- data augmentation
- early stopping
- dropout
- batch normalization

What is regularization?



- In general: any method to **prevent overfitting** or **help the optimization**
- Specifically: additional terms in the training optimization objective to prevent overfitting or help the optimization

Example: regression using polynomials

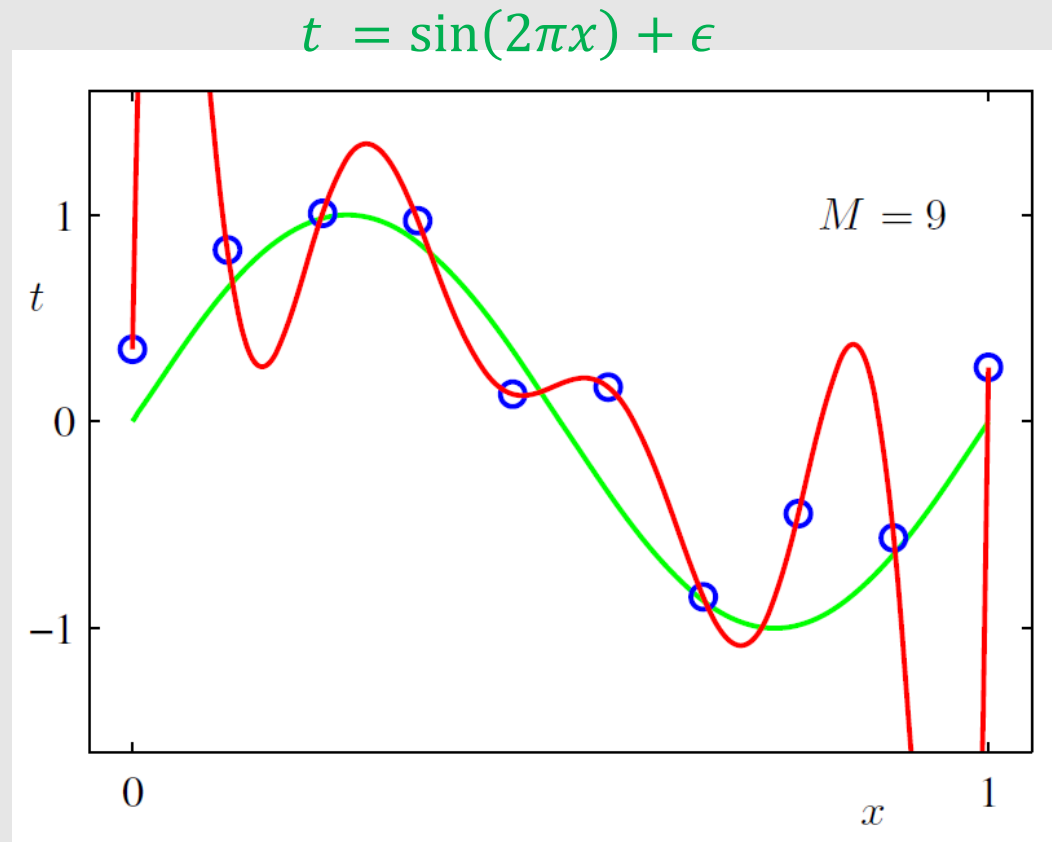


Figure from *Machine Learning and Pattern Recognition*, Bishop

Example: regression using polynomials

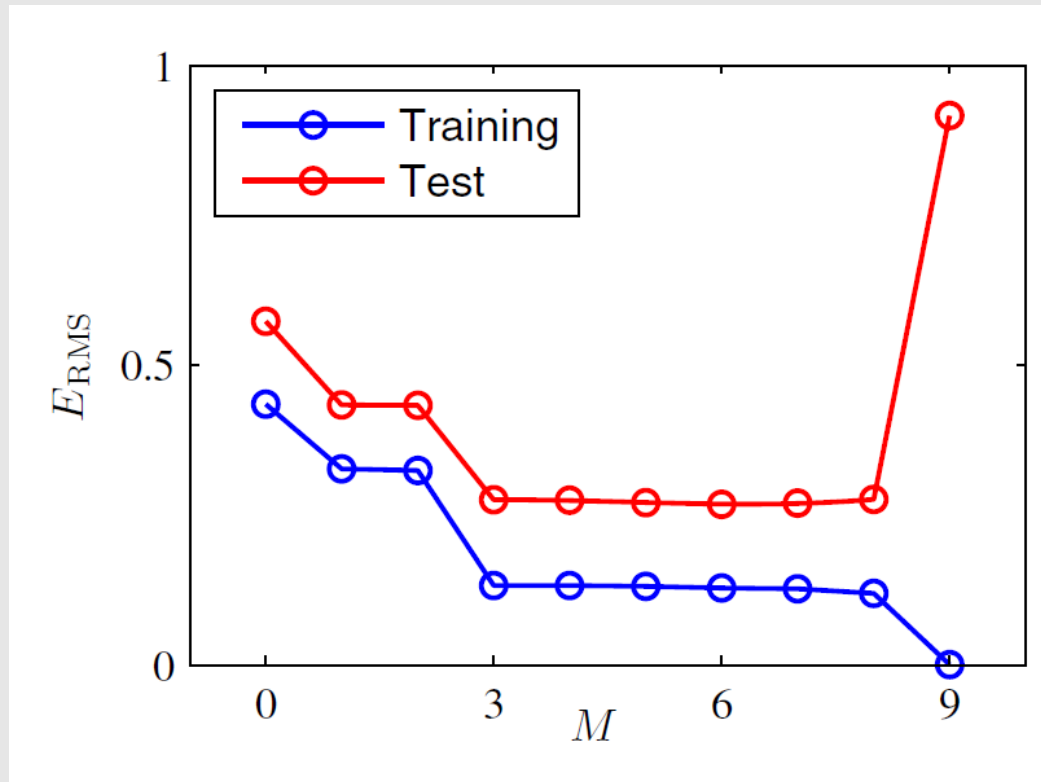


Figure from *Machine Learning and Pattern Recognition*, Bishop

Overfitting



- Key: empirical loss and expected loss are different
- Smaller the data set, larger the difference between the two
- Larger the hypothesis class, easier to find a hypothesis that fits the difference between the two
 - Thus has small training error but large test error (overfitting)
- Larger data set helps
- Throwing away useless hypotheses also helps (regularization)



Different views of regularization

Regularization as hard constraint



- Training objective

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

subject to: $f \in \mathcal{H}$

- When parametrized

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

subject to: $\theta \in \Omega$

Regularization as hard constraint



- When Ω measured by some quantity R

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

subject to: $R(\theta) \leq r$

- Example: l_2 regularization

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

subject to: $\|\theta\|_2^2 \leq r^2$

Regularization as soft constraint



- The hard-constraint optimization is equivalent to soft-constraint

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

for some regularization parameter $\lambda^* > 0$

- Example: l_2 regularization

$$\min_{\theta} \hat{L}_R(\theta) = \frac{1}{n} \sum_{i=1}^n l(\theta, x_i, y_i) + \lambda^* \|\theta\|_2^2$$

Regularization as soft constraint



- Showed by Lagrangian multiplier method

$$\mathcal{L}(\theta, \lambda) := \hat{L}(\theta) + \lambda[R(\theta) - r]$$

- Suppose θ^* is the optimal for hard-constraint optimization

$$\theta^* = \operatorname{argmin}_{\theta} \max_{\lambda \geq 0} \mathcal{L}(\theta, \lambda) := \hat{L}(\theta) + \lambda[R(\theta) - r]$$

- Suppose λ^* is the corresponding optimal for max

$$\theta^* = \operatorname{argmin}_{\theta} \mathcal{L}(\theta, \lambda^*) := \hat{L}(\theta) + \lambda^*[R(\theta) - r]$$

Regularization as Bayesian prior



- Bayesian view: everything is a distribution
- Prior over the hypotheses: $p(\theta)$
- Posterior over the hypotheses: $p(\theta | \{x_i, y_i\})$
- Likelihood: $p(\{x_i, y_i\} | \theta)$

- Bayesian rule:

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

Regularization as Bayesian prior



- Bayesian rule:

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

- Maximum A Posteriori (MAP):

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

Regularization

MLE loss

Regularization as Bayesian prior



- Example: l_2 loss with l_2 regularization

$$\min_{\theta} \hat{L}_R(\theta) = \frac{1}{n} \sum_{i=1}^n (f_{\theta}(x_i) - y_i)^2 + \lambda^* \|\theta\|_2^2$$

- Correspond to a normal likelihood $p(x, y | \theta)$ and a normal prior $p(\theta)$

Three views



- Typical choice for optimization: soft-constraint

$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \lambda R(\theta)$$

- Hard constraint and Bayesian view: conceptual; or used for derivation

Three views



- Hard-constraint preferred if
 - Know the explicit bound $R(\theta) \leq r$
 - Soft-constraint causes trapped in a local minima while projection back to feasible set leads to stability
- Bayesian view preferred if
 - Domain knowledge easy to represent as a prior



Examples of Regularization

Classical regularization



- Norm penalty
 - l_2 regularization
 - l_1 regularization
- Robustness to noise
 - Noise to the input
 - Noise to the weights

l_2 regularization



$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \frac{\alpha}{2} \|\theta\|_2^2$$

- Effect on (stochastic) gradient descent
- Effect on the optimal solution

Effect on gradient descent



- Gradient of regularized objective

$$\nabla \hat{L}_R(\theta) = \nabla \hat{L}(\theta) + \alpha \theta$$

- Gradient descent update

$$\theta \leftarrow \theta - \eta \nabla \hat{L}_R(\theta) = \theta - \eta \nabla \hat{L}(\theta) - \eta \alpha \theta = (1 - \eta \alpha) \theta - \eta \nabla \hat{L}(\theta)$$

- Terminology: weight decay

Effect on the optimal solution



- Consider a quadratic approximation around θ^*

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + (\theta - \theta^*)^T \nabla \hat{L}(\theta^*) + \frac{1}{2} (\theta - \theta^*)^T H (\theta - \theta^*)$$

- Since θ^* is optimal, $\nabla \hat{L}(\theta^*) = 0$

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + \frac{1}{2} (\theta - \theta^*)^T H (\theta - \theta^*)$$

$$\nabla \hat{L}(\theta) \approx H (\theta - \theta^*)$$

Effect on the optimal solution



- Gradient of regularized objective

$$\nabla \hat{L}_R(\theta) \approx H(\theta - \theta^*) + \alpha\theta$$

- On the optimal θ_R^*

$$0 = \nabla \hat{L}_R(\theta_R^*) \approx H(\theta_R^* - \theta^*) + \alpha\theta_R^*$$

$$\theta_R^* \approx (H + \alpha I)^{-1} H \theta^*$$

Effect on the optimal solution



- The optimal

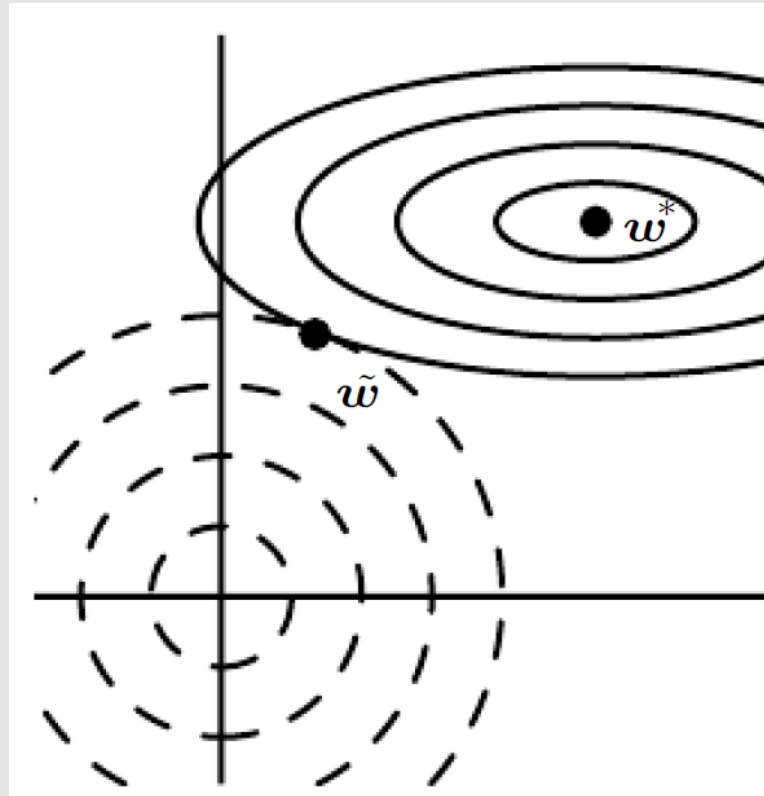
$$\theta_R^* \approx (H + \alpha I)^{-1} H \theta^*$$

- Suppose H has eigen-decomposition $H = Q\Lambda Q^T$ and assume $(\Lambda + \alpha I)^{-1}$ exists:

$$\theta_R^* \approx (H + \alpha I)^{-1} H \theta^* = Q(\Lambda + \alpha I)^{-1} \Lambda Q^T \theta^*$$

- Effect: rescale along eigenvectors of H

Effect on the optimal solution



Notations:

$$\theta^* = w^*, \theta_R^* = \tilde{w}$$

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

l_1 regularization



$$\min_{\theta} \hat{L}_R(\theta) = \hat{L}(\theta) + \alpha \|\theta\|_1$$

- Effect on (stochastic) gradient descent
- Effect on the optimal solution

Effect on gradient descent



- Gradient of regularized objective

$$\nabla \hat{L}_R(\theta) = \nabla \hat{L}(\theta) + \alpha \text{sign}(\theta)$$

where **sign** applies to each element in θ

- Gradient descent update

$$\theta \leftarrow \theta - \eta \nabla \hat{L}_R(\theta) = \theta - \eta \nabla \hat{L}(\theta) - \eta \alpha \text{sign}(\theta)$$

Effect on the optimal solution



- Consider a quadratic approximation around θ^*

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + (\theta - \theta^*)^T \nabla \hat{L}(\theta^*) + \frac{1}{2} (\theta - \theta^*)^T H(\theta - \theta^*)$$

- Since θ^* is optimal, $\nabla \hat{L}(\theta^*) = 0$

$$\hat{L}(\theta) \approx \hat{L}(\theta^*) + \frac{1}{2} (\theta - \theta^*)^T H(\theta - \theta^*)$$



Effect on the optimal solution

- Further assume that H is diagonal and positive ($H_{ii} > 0, \forall i$)
 - not true in general but assume for getting some intuition
- The regularized objective is (ignoring constants)

$$\hat{L}_R(\theta) \approx \sum_i \frac{1}{2} H_{ii} (\theta_i - \theta_i^*)^2 + \alpha |\theta_i|$$

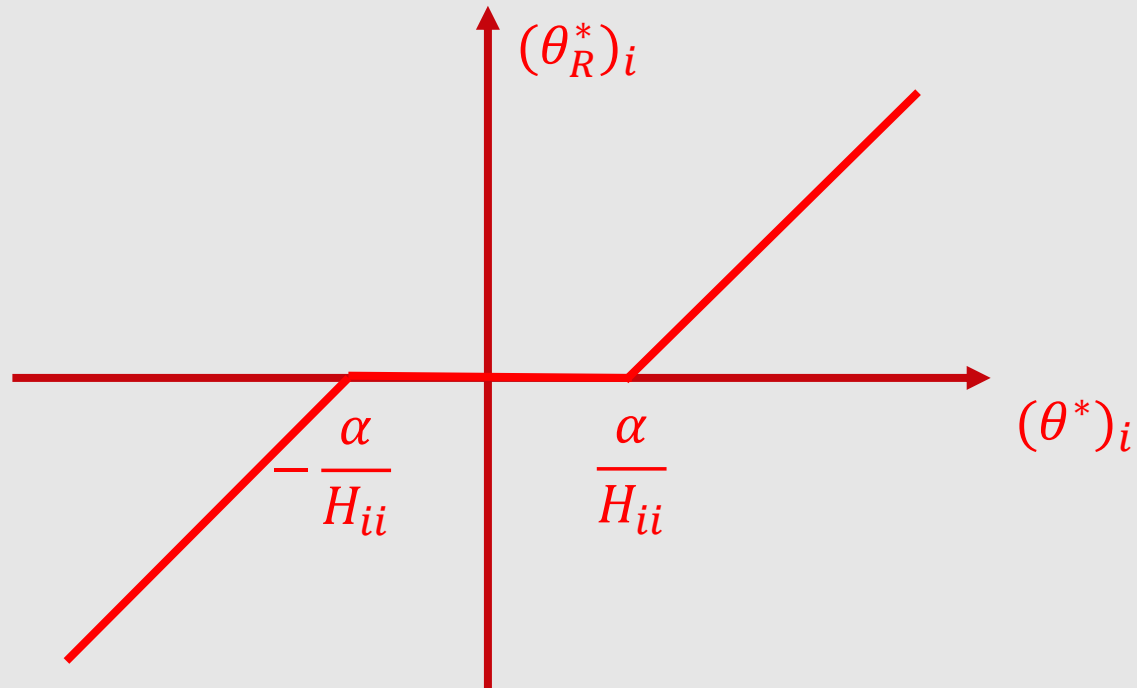
- The optimal θ_R^*

$$(\theta_R^*)_i \approx \begin{cases} \max \left\{ \theta_i^* - \frac{\alpha}{H_{ii}}, 0 \right\} & \text{if } \theta_i^* \geq 0 \\ \min \left\{ \theta_i^* + \frac{\alpha}{H_{ii}}, 0 \right\} & \text{if } \theta_i^* < 0 \end{cases}$$

Effect on the optimal solution



- Effect: induce sparsity



Effect on the optimal solution



- Further assume that H is diagonal
- Compact expression for the optimal θ_R^*

$$(\theta_R^*)_i \approx \text{sign}(\theta_i^*) \max\{|\theta_i^*| - \frac{\alpha}{H_{ii}}, 0\}$$

Bayesian view

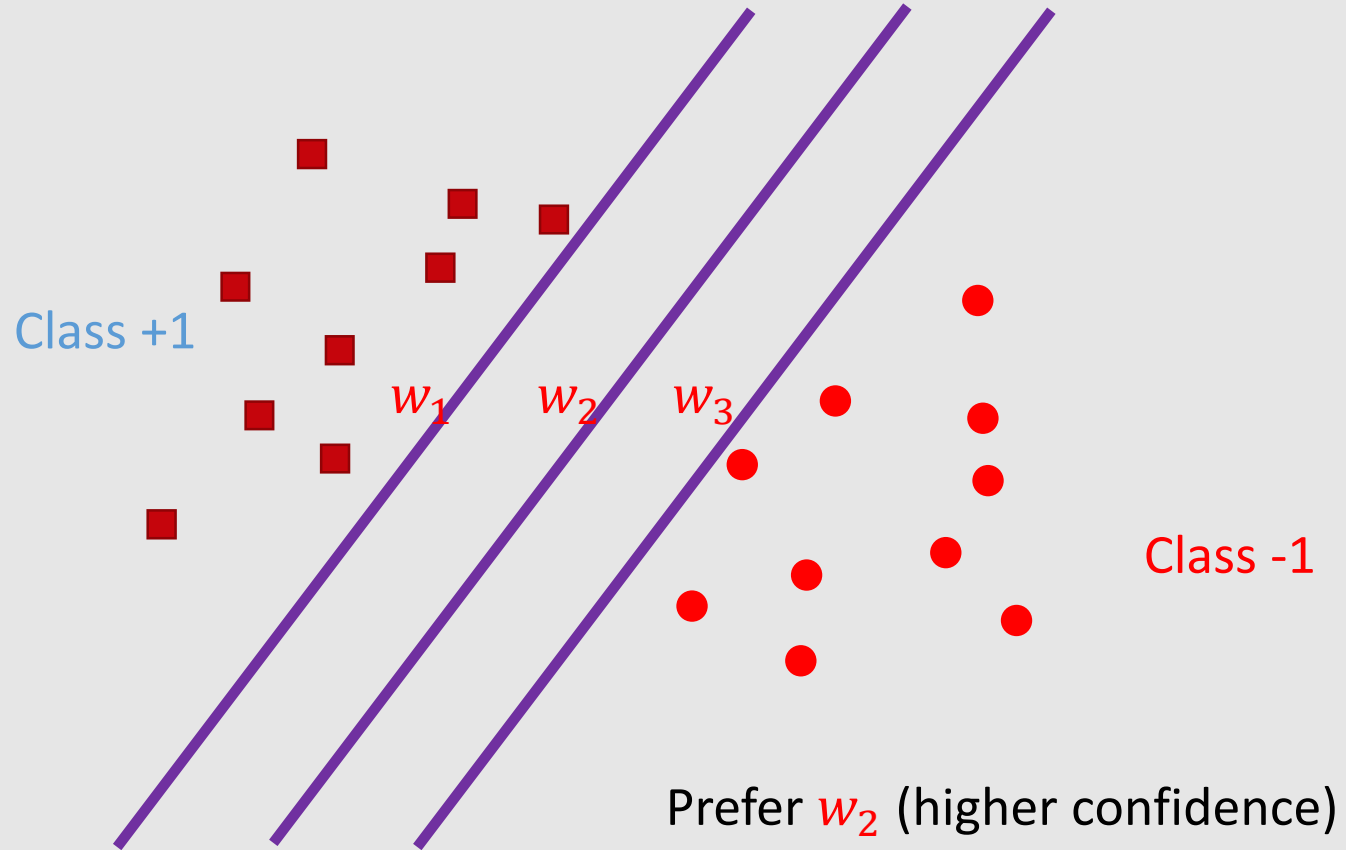


- l_1 regularization corresponds to Laplacian prior

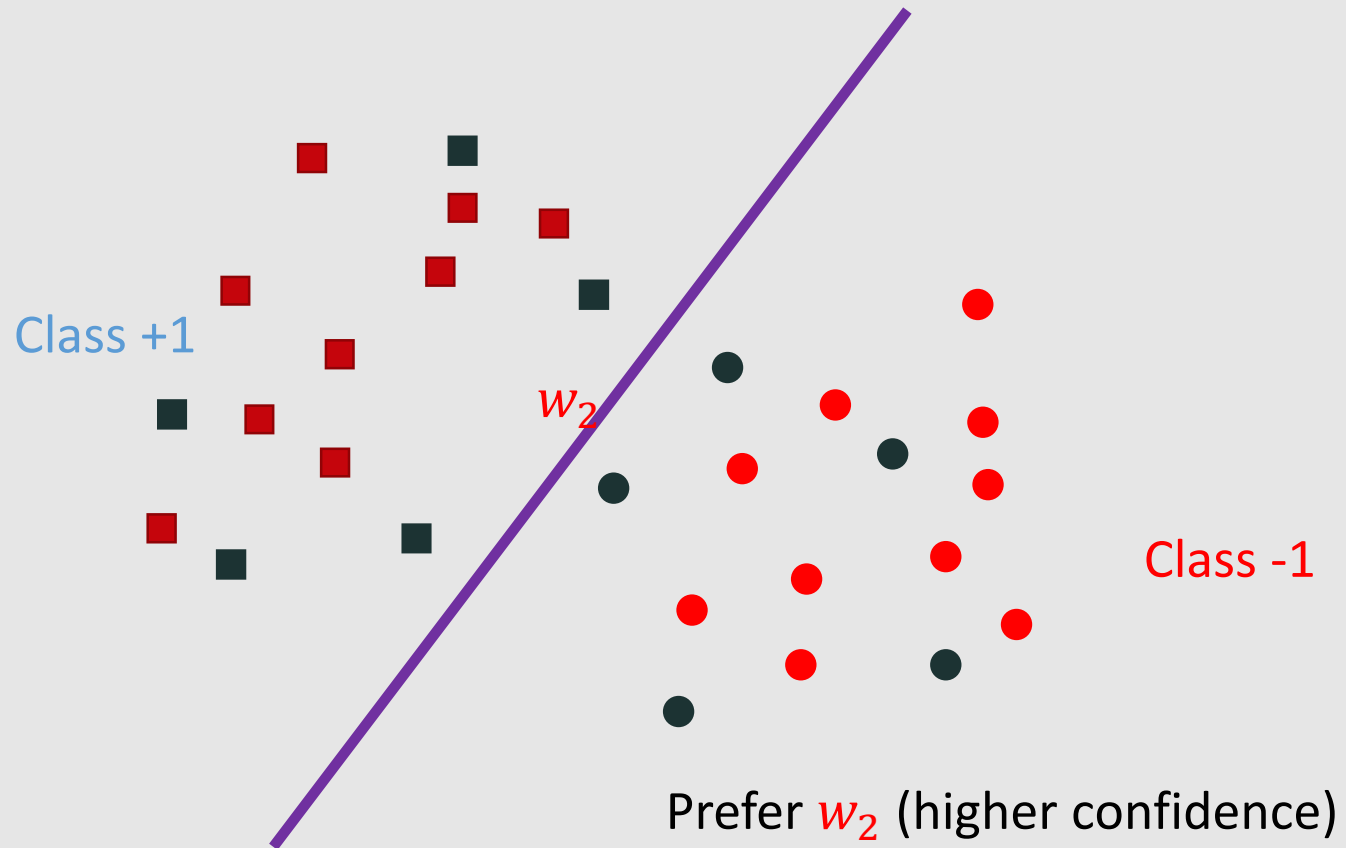
$$p(\theta) \propto \exp(\alpha \sum_i |\theta_i|)$$

$$\log p(\theta) = \alpha \sum_i |\theta_i| + \text{constant} = \alpha \|\theta\|_1 + \text{constant}$$

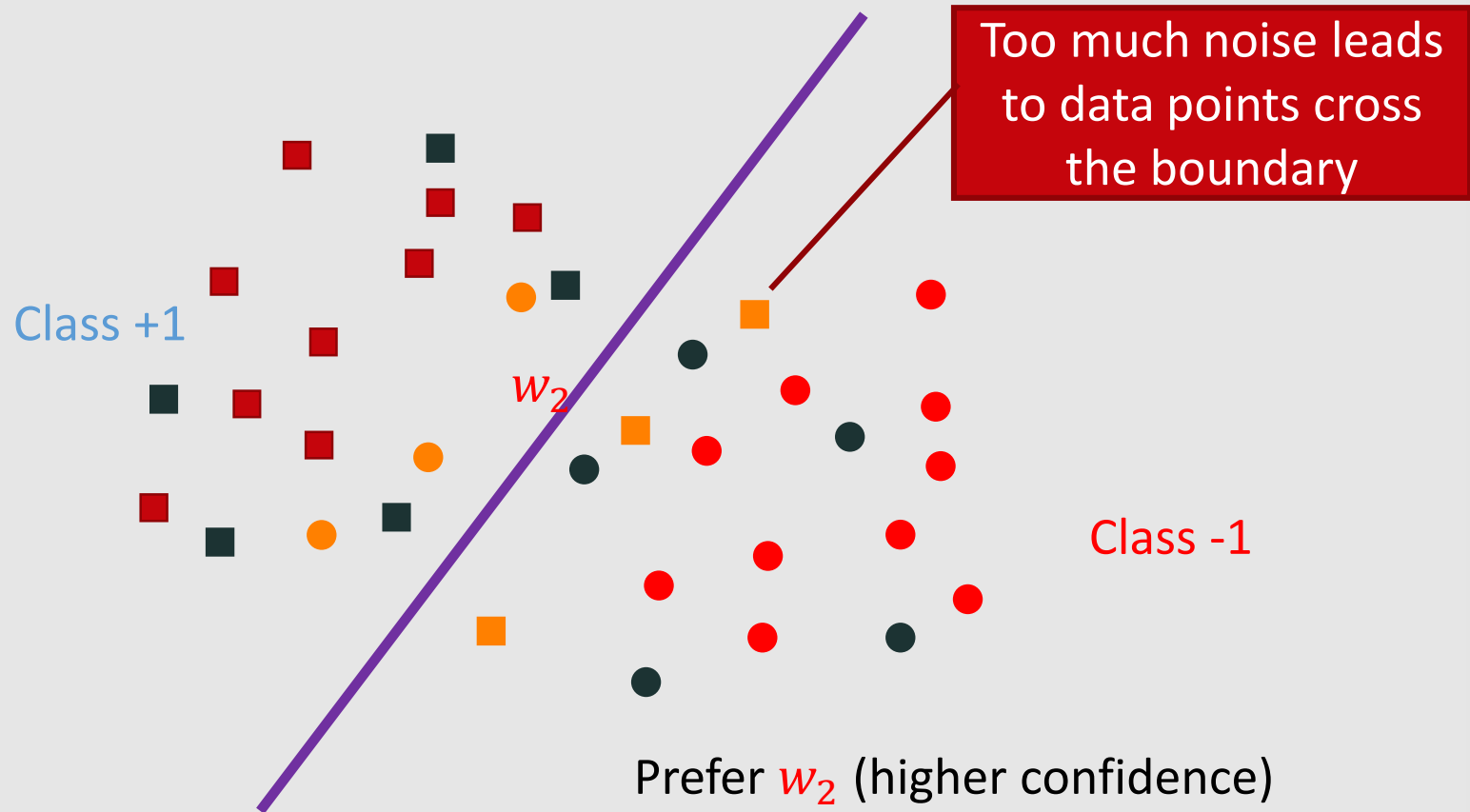
Multiple optimal solutions?



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

- $f_{w+\epsilon}(x) \approx f_w(x) + \epsilon^T \nabla f_w(x)$

- Plug in

- $L(f) \approx \mathbb{E}[f_w(x) - y]^2 + \eta \mathbb{E}[\|\nabla f_w(x)\|^2]$


Regularization term

Other types of regularizations



- Data augmentation
- Early stopping
- Dropout
- Batch Normalization

Data augmentation

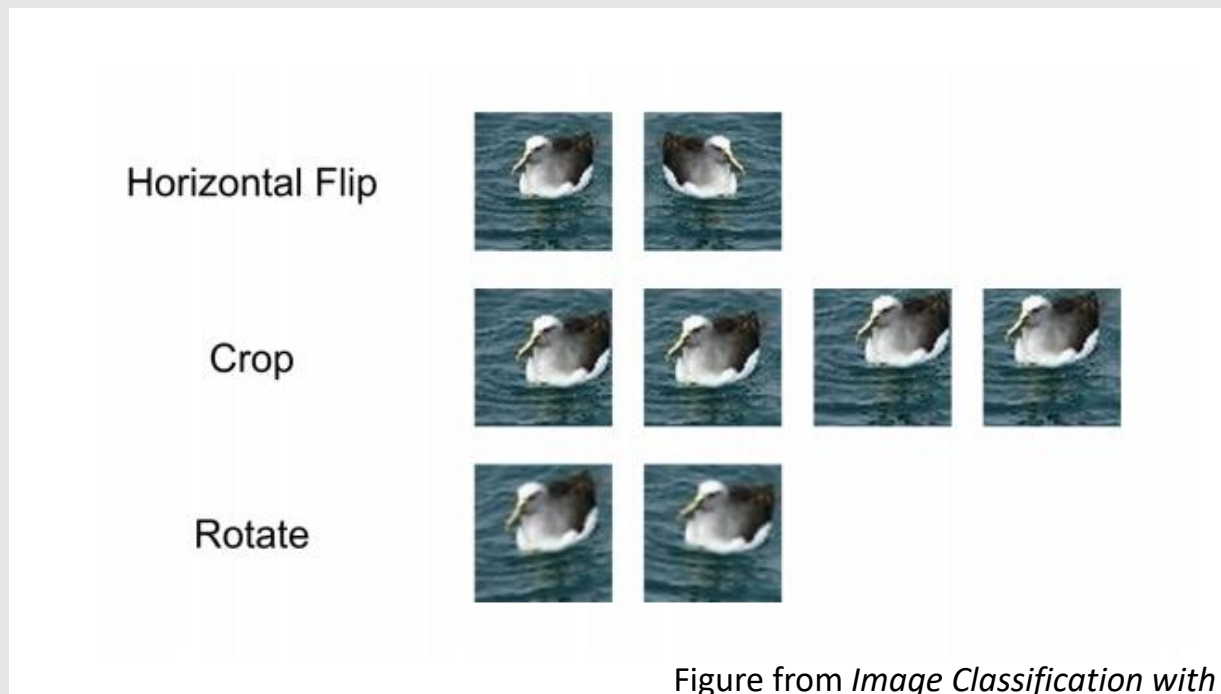


Figure from *Image Classification with Pyramid Representation and Rotated Data Augmentation on Torch 7*, by Keven Wang

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

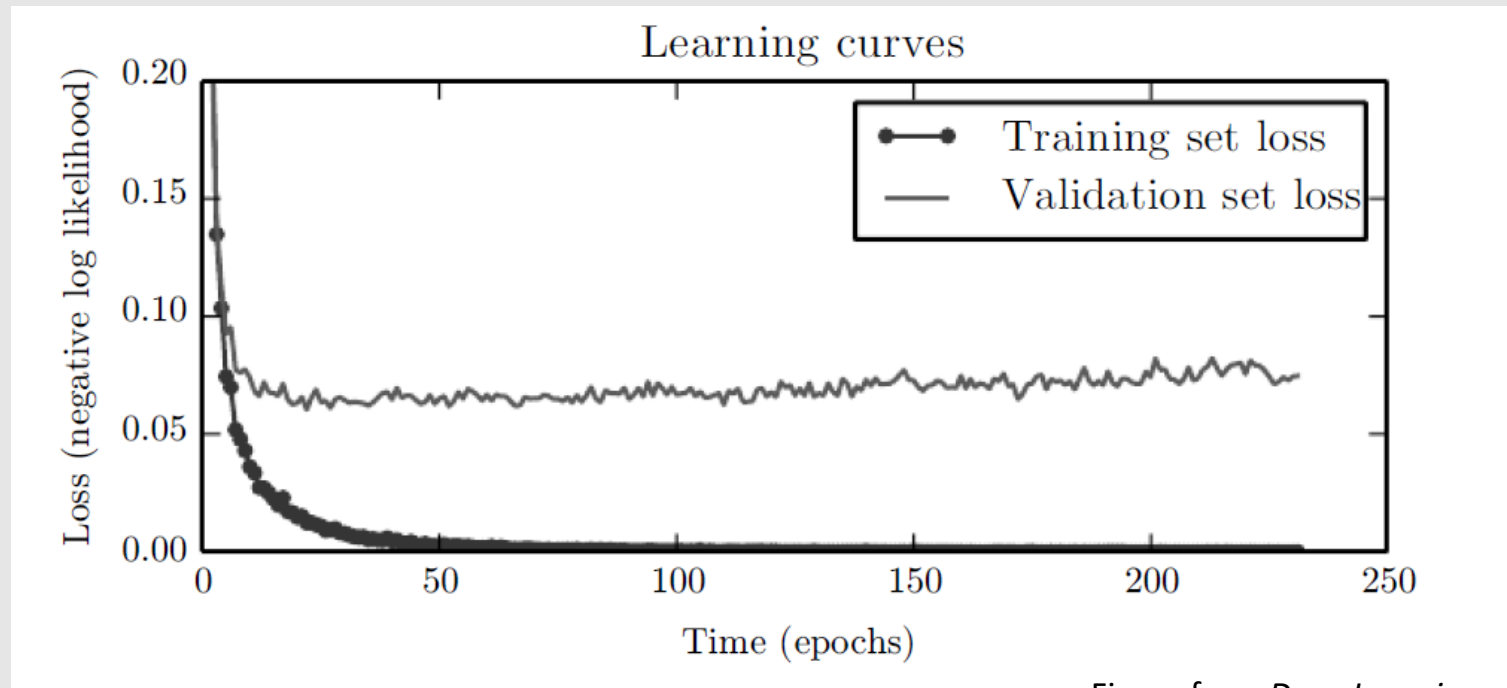


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

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 as a regularizer

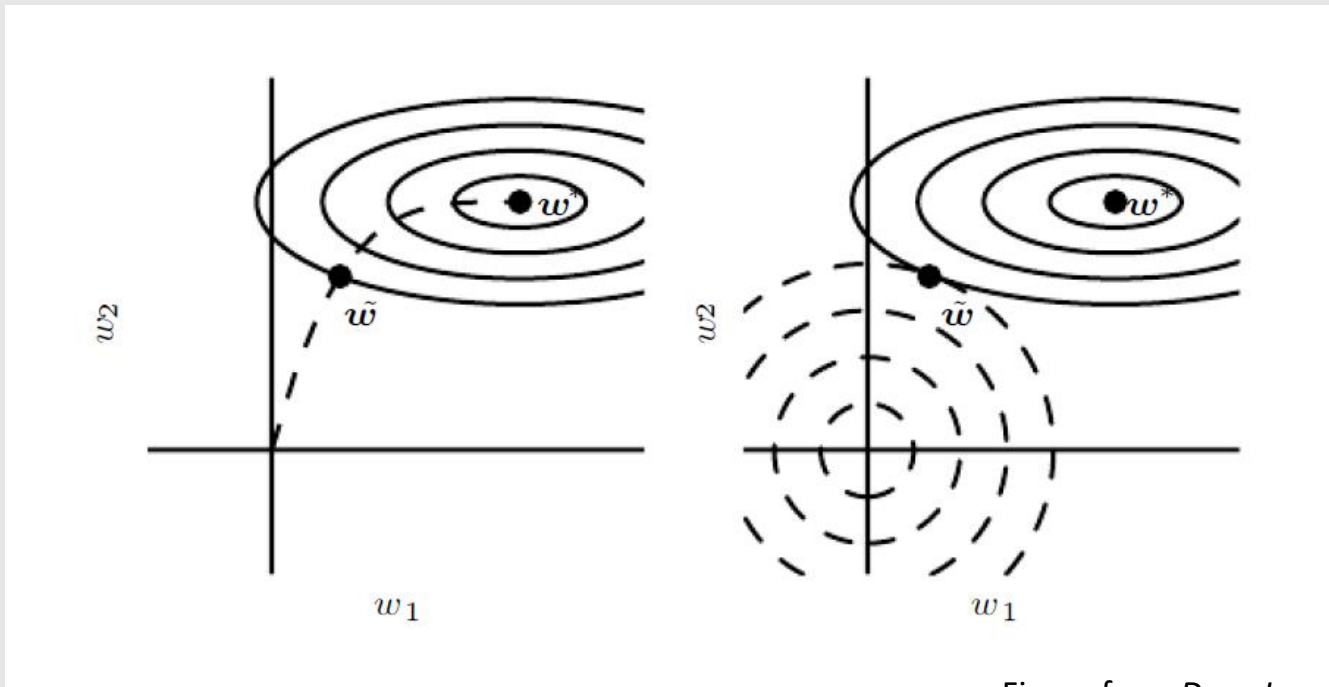


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

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

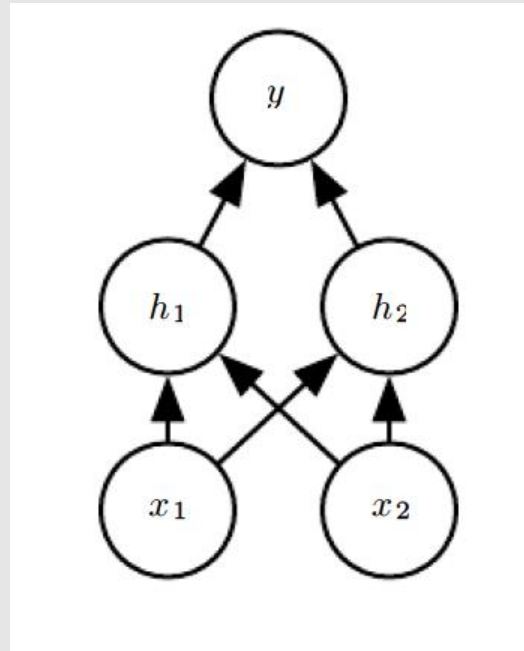


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

Dropout

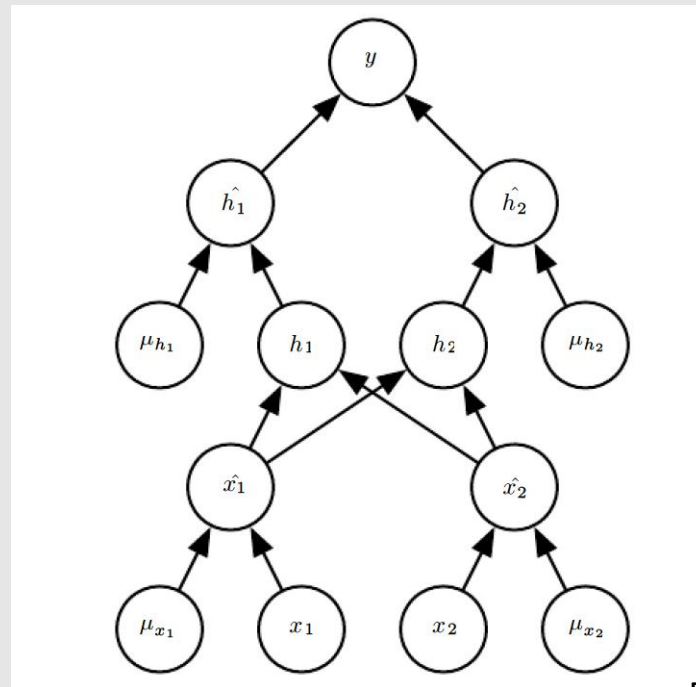


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

Dropout

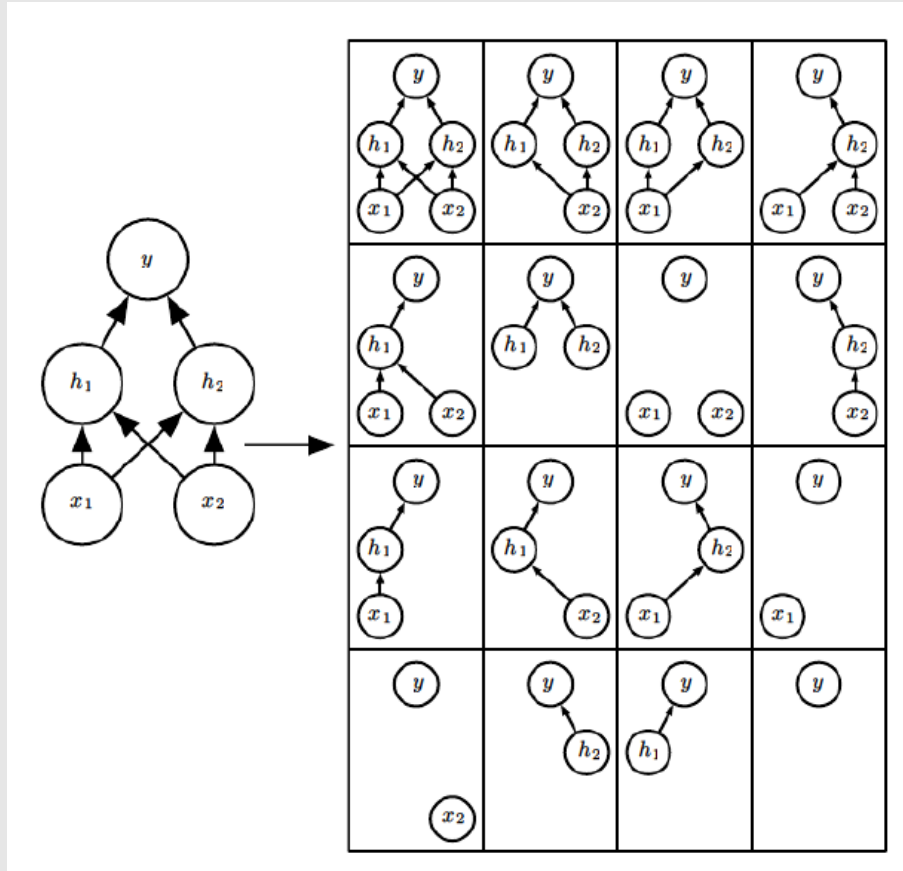


Figure from *Deep Learning*,
Goodfellow, Bengio and Courville

Batch Normalization



- If outputs of earlier layers are uniform or change greatly on one round for one mini-batch, then neurons at next levels can't keep up: they output all high (or all low) values
- Next layer doesn't have ability to change its outputs with learning-rate-sized changes to its input weights
- We say the layer has “saturated”

Another View of Problem



- In ML, we assume future data will be drawn from same probability distribution as training data
- For a hidden unit, after training, the earlier layers have new weights and hence generate input data for this hidden unit from a *new* distribution
- Want to reduce this *internal covariate shift* for the benefit of later layers



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Comments on Batch Normalization



- First three steps are just like standardization of input data, but with respect to only the data in mini-batch. Can take derivative and incorporate the learning of last step parameters into backpropagation.
- Note last step can completely un-do previous 3 steps
- But if so this un-doing is driven by the *later* layers, not the *earlier* layers; later layers get to “choose” whether they want standard normal inputs or not

What regularizations are frequently used?



- l_2 regularization
- Early stopping
- Dropout/Batch Normalization

- Data augmentation if the transformations known/easy to implement



THANK YOU

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Matt Gormley, Elad Hazan, Tom Dietterich, and Pedro Domingos.

