

# Learning From Data

## Lecture 22

### Neural Networks and Overfitting

Approximation vs. Generalization  
 Regularization and Early Stopping  
 Minimizing  $E_{in}$  More Efficiently ☹

M. Magdon-Ismail  
 CSCI 4100/6100

## RECAP: Neural Networks and Fitting the Data

### Forward Propagation:

$$\mathbf{x} = \mathbf{x}^{(0)} \xrightarrow{w^{(1)}} \mathbf{s}^{(1)} \xrightarrow{\theta} \mathbf{x}^{(1)} \xrightarrow{w^{(2)}} \mathbf{s}^{(2)} \dots \xrightarrow{w^{(L)}} \mathbf{s}^{(L)} \xrightarrow{\theta} \mathbf{x}^{(L)} = h(\mathbf{x})$$

$$\mathbf{s}^{(l)} = (W^{(l)})^T \mathbf{x}^{(l-1)} \quad \mathbf{x}^{(l)} = \begin{bmatrix} 1 \\ \theta(\mathbf{s}^{(l)}) \end{bmatrix}$$

(Compute  $h$  and  $E_{in}$ )

Choose  $W = \{W^{(1)}, W^{(2)}, \dots, W^{(L)}\}$  to minimize  $E_{in}$

### Gradient descent:

$$W(t+1) \leftarrow W(t) - \eta \nabla E_{in}(W(t))$$

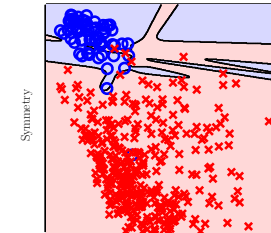
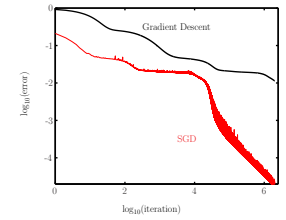
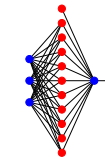
Compute gradient  $\rightarrow$  need  $\frac{\partial e}{\partial W^{(l)}} \rightarrow$  need  $\delta^{(l)} = \frac{\partial e}{\partial \mathbf{s}^{(l)}}$

$$\frac{\partial e}{\partial W^{(l)}} = \mathbf{x}^{(l-1)} (\delta^{(l)})^T$$

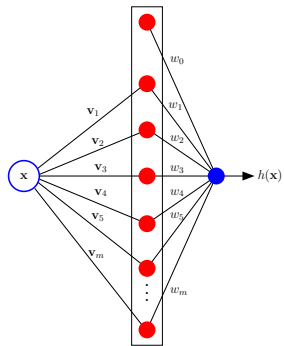
### Backpropagation:

$$\delta^{(1)} \leftarrow \delta^{(2)} \dots \leftarrow \delta^{(L-1)} \leftarrow \delta^{(L)}$$

$$\delta^{(l)} = \theta'(\mathbf{s}^{(l)}) \otimes [W^{(l+1)} \delta^{(l+1)}]_1^{d^{(l)}}$$



## 2-Layer Neural Network



$$h(\mathbf{x}) = \theta \left( w_0 + \sum_{j=1}^m w_j \theta(\mathbf{v}_j^T \mathbf{x}) \right)$$

## The Neural Network has a Tunable Transform

Neural Network

$$h(\mathbf{x}) = \theta \left( w_0 + \sum_{j=1}^m w_j \theta(\mathbf{v}_j^T \mathbf{x}) \right)$$

Nonlinear Transform

$$h(\mathbf{x}) = \theta \left( w_0 + \sum_{j=1}^d w_j \Phi_j(\mathbf{x}) \right)$$

k-RBF-Network

$$h(\mathbf{x}) = \theta \left( w_0 + \sum_{j=1}^k w_j \phi(\|\mathbf{x} - \boldsymbol{\mu}_j\|) \right)$$

$$E_{in} = O \left( \frac{1}{m} \right)$$

↑  
approximation

## Generalization

MLP:

$$d_{VC} = O(md \log(md))$$

$$m = \sqrt{N}$$

(convergence to optimal for MLP, just like  $k$ -NN)

*semi-parametric* because you still have to learn parameters.

tanh :

$$d_{VC} = O(md(m + d))$$

## Regularization – Weight Decay

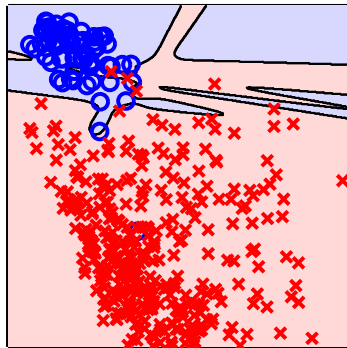
$$E_{\text{aug}}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N (h(\mathbf{x}_n; \mathbf{w}) - y_n)^2 + \frac{\lambda}{N} \sum_{\ell, i, j} (w_{ij}^{(\ell)})^2$$

$$\frac{\partial E_{\text{aug}}(\mathbf{w})}{\partial W^{(\ell)}} = \frac{\partial E_{\text{in}}(\mathbf{w})}{\partial W^{(\ell)}} + \frac{2\lambda}{N} W^{(\ell)}$$

↑  
backpropagation

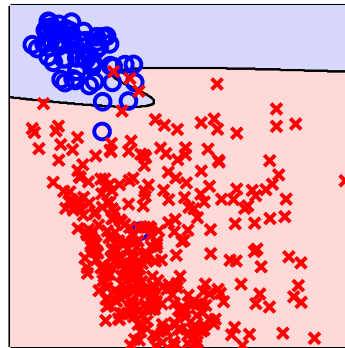
## Weight Decay with Digits Data

No Weight Decay



Average Intensity

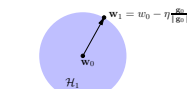
Weight Decay,  $\lambda = 0.01$



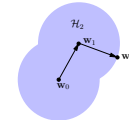
Average Intensity

## Early Stopping

Gradient Descent



$$\mathcal{H}_1 = \{w : \|w - w_0\| \leq \eta\}$$



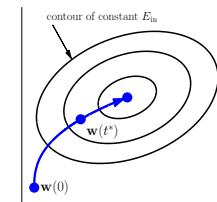
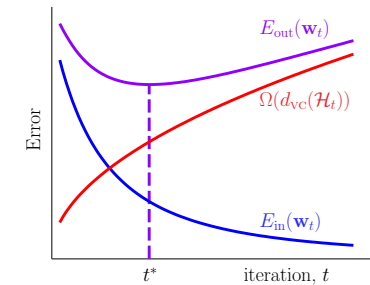
$$\mathcal{H}_2 = \mathcal{H}_1 \cup \{w : \|w - w_1\| \leq \eta\}$$



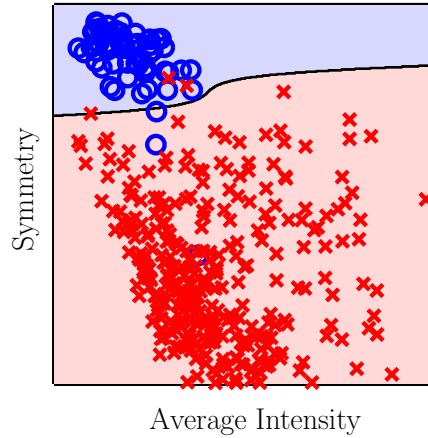
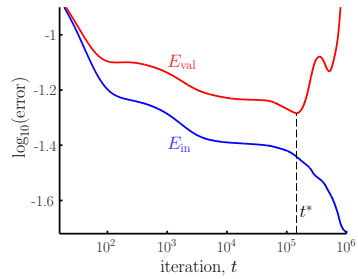
$$\mathcal{H}_3 = \mathcal{H}_2 \cup \{w : \|w - w_2\| \leq \eta\}$$

Each iteration explores a larger  $\mathcal{H}$

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3 \subset \mathcal{H}_4 \subset \dots$$



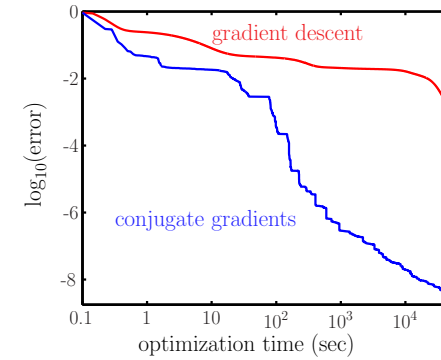
## Early Stopping on Digits Data



Use a validation set to determine  $t^*$   
Output  $w^*$ , do not retrain with all the data till  $t^*$ .

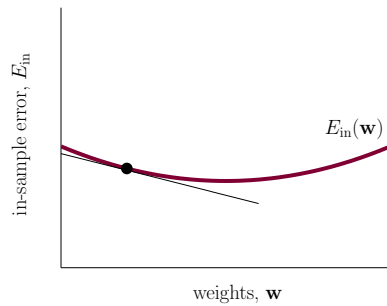
## Minimizing $E_{in}$

1. Use regression for classification
2. Use better algorithms than gradient descent

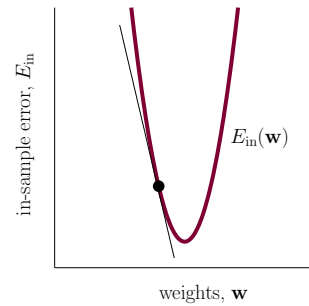


## Beefing Up Gradient Descent

Determine the gradient  $\mathbf{g}$



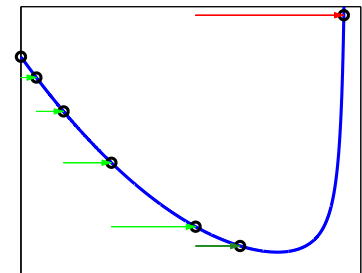
Shallow: use large  $\eta$ .



Deep: use small  $\eta$ .

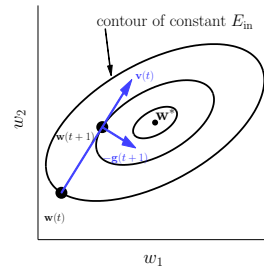
## Variable Learning Rate Gradient Descent

1. Initialize  $\mathbf{w}(0)$ , and  $\eta_0$  at  $t = 0$ . Set  $\alpha > 1$  and  $\beta < 1$ .
2. **while** stopping criterion has not been met **do**
3. Let  $\mathbf{g}(t) = \nabla E_{in}(\mathbf{w}(t))$ , and set  $\mathbf{v}(t) = -\mathbf{g}(t)$ .
4. **if**  $E_{in}(\mathbf{w}(t) + \eta_t \mathbf{v}(t)) < E_{in}(\mathbf{w}(t))$  **then**
5. **accept:**  $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta_t \mathbf{v}(t)$ ;  
**increment  $\eta$ :**  $\eta_{t+1} = \alpha \eta_t$ .  $\alpha \in [1.05, 1.1]$
6. **else**
7. **reject:**  $\mathbf{w}(t+1) = \mathbf{w}(t)$ ;  
**decrease  $\eta$ :**  $\eta_{t+1} = \beta \eta_t$ .  $\beta \in [0.7, 0.8]$
8. **end if**
9. Iterate to the next step,  $t \leftarrow t + 1$ .
10. **end while**

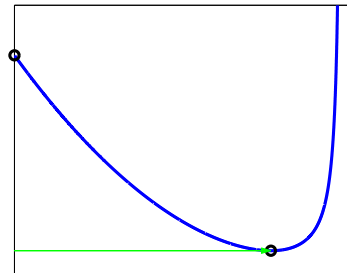
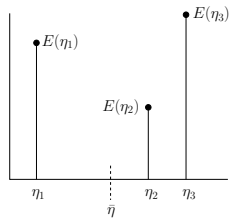


## Steepest Descent - Line Search

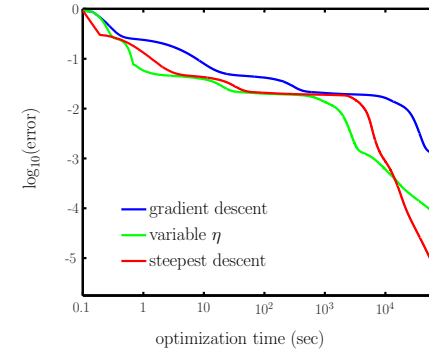
- 1: Initialize  $w(0)$  and set  $t = 0$ ;
- 2: **while** stopping criterion has not been met **do**
- 3: Let  $\mathbf{g}(t) = \nabla E_{in}(\mathbf{w}(t))$ , and set  $\mathbf{v}(t) = -\mathbf{g}(t)$ .
- 4: Let  $\eta^* = \operatorname{argmin}_{\eta} E_{in}(\mathbf{w}(t) + \eta \mathbf{v}(t))$ .
- 5:  $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta^* \mathbf{v}(t)$ .
- 6: Iterate to the next step,  $t \leftarrow t + 1$ .
- 7: **end while**



How to accomplish the line search (step 4)?  
Simple bisection (binary search) suffices in practice



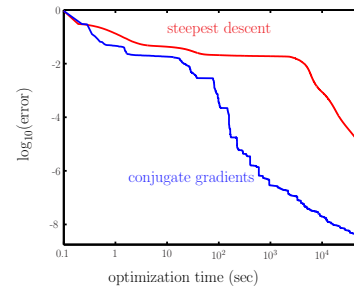
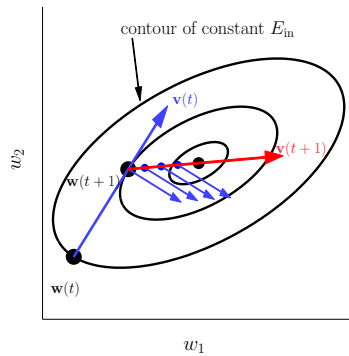
## Comparison of Optimization Heuristics



Method	Optimization Time		
	10 sec	1,000 sec	50,000 sec
Gradient Descent	0.122	0.0214	0.0113
<b>Stochastic Gradient Descent</b>	<b>0.0203</b>	<b>0.000447</b>	<b><math>1.6310 \times 10^{-5}</math></b>
Variable Learning Rate	0.0432	0.0180	0.000197
Steepest Descent	0.0497	0.0194	0.000140

## Conjugate Gradients

1. Line search just like steepest descent.
2. Choose a better direction than  $-\mathbf{g}$



Method	Optimization Time		
	10 sec	1,000 sec	50,000 sec
Stochastic Gradient Descent	0.0203	0.000447	$1.6310 \times 10^{-5}$
Steepest Descent	0.0497	0.0194	0.000140
<b>Conjugate Gradients</b>	<b>0.0200</b>	<b><math>1.13 \times 10^{-6}</math></b>	<b><math>2.73 \times 10^{-9}</math></b>

There are better algorithms (eg. Levenberg-Marquardt), but we will stop here ☹

