

Gradient Descent

- how are we going to find the solution for

$$\arg \min_{b,w} \sum_{i=1}^n \ell(b + w^T x_i, y_i)$$

- e.g., Logistic Regression do not have closed form solution for

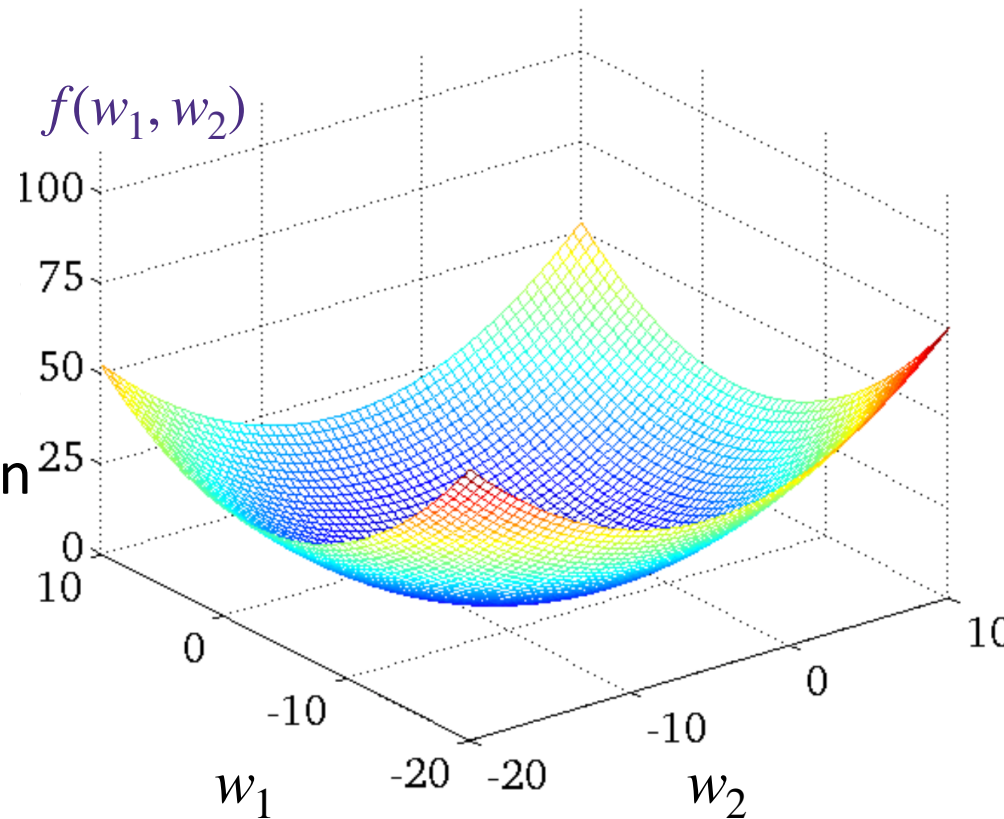
$$\nabla_{b,w} \mathcal{L}(b, w) = 0$$

Running example: linear regression

- **Given data:** $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$
- **Learning model parameters:**

$$\hat{w}_{\text{LS}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\|\mathbf{y} - \mathbf{X}w\|_2^2}_{f(w)}$$

- Although we know the optimal solution in a closed form, we will use this as a running example to understand GD



1-dimensional gradient descent

$$\min f(x)$$

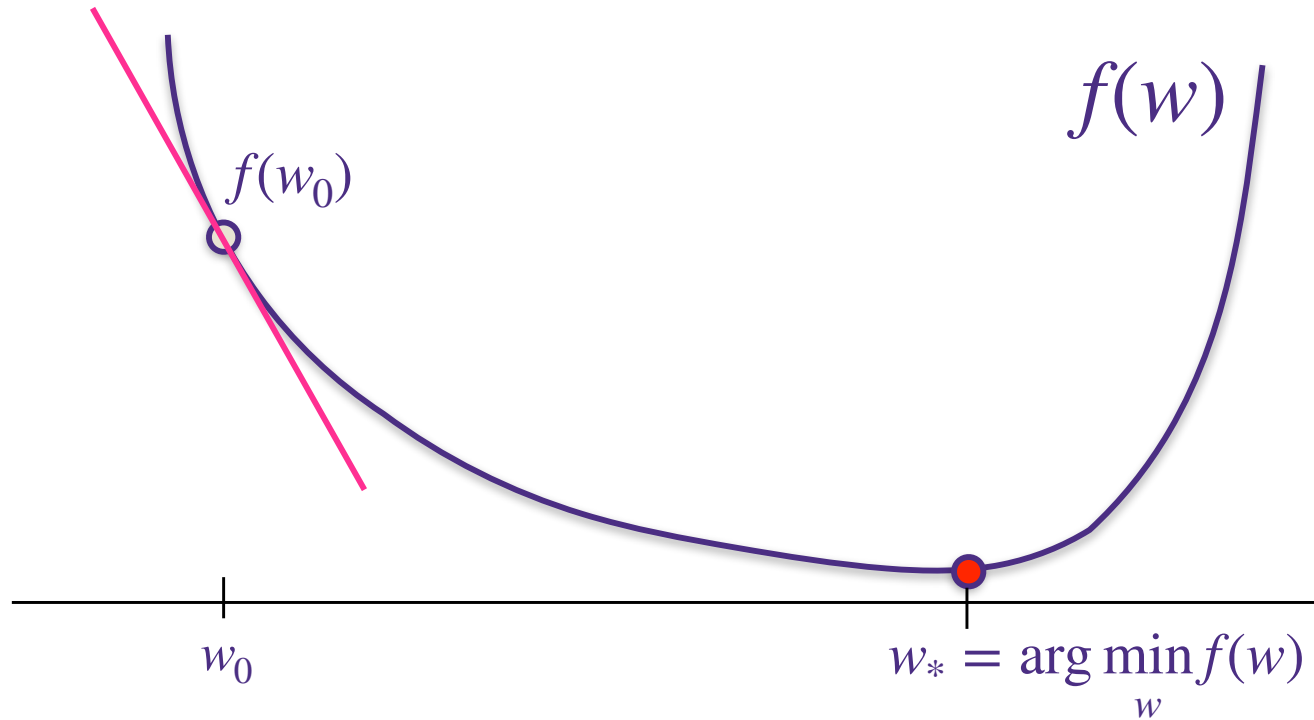
Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

$$\underbrace{f(w)}_{\approx} f(w_0) + (w - w_0) \left. \frac{df(w)}{dw} \right|_{w=w_0}$$

is very close to $f(w)$



1-dimensional gradient descent

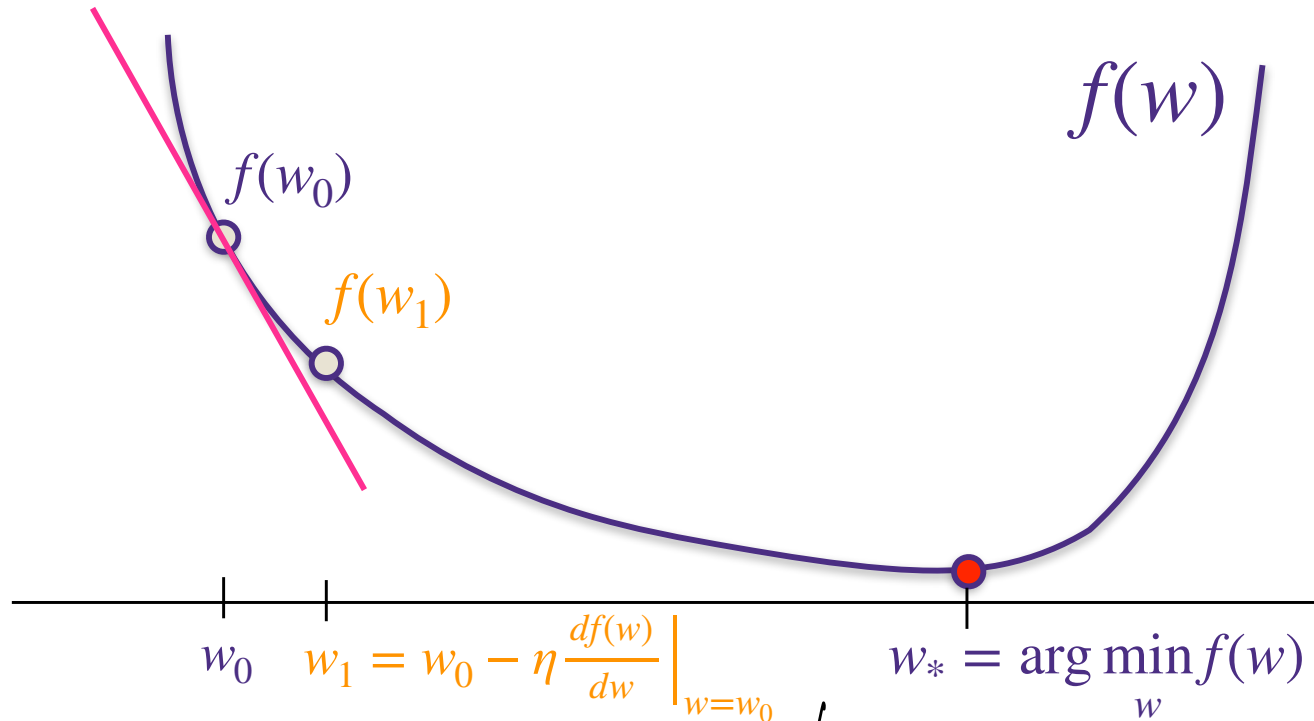
Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

$$f(w_0) + (w - w_0) \frac{df(w)}{dw} \Big|_{w=w_0}$$

is very close to $f(w)$



Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw} \Big|_{w=w_0}$ then

$$f(w_0) - \eta \left(\frac{df(w)}{dw} \Big|_{w=w_0} \right)^2$$

is very close to $f(w_1) < f(w_0)$

if $w - w_0$ has same direction as $\frac{df(w)}{dw} \Big|_{w=w_0}$

$$w_1 - w_0 \parallel - \frac{df(w)}{dw} \Big|_{w=w_0}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

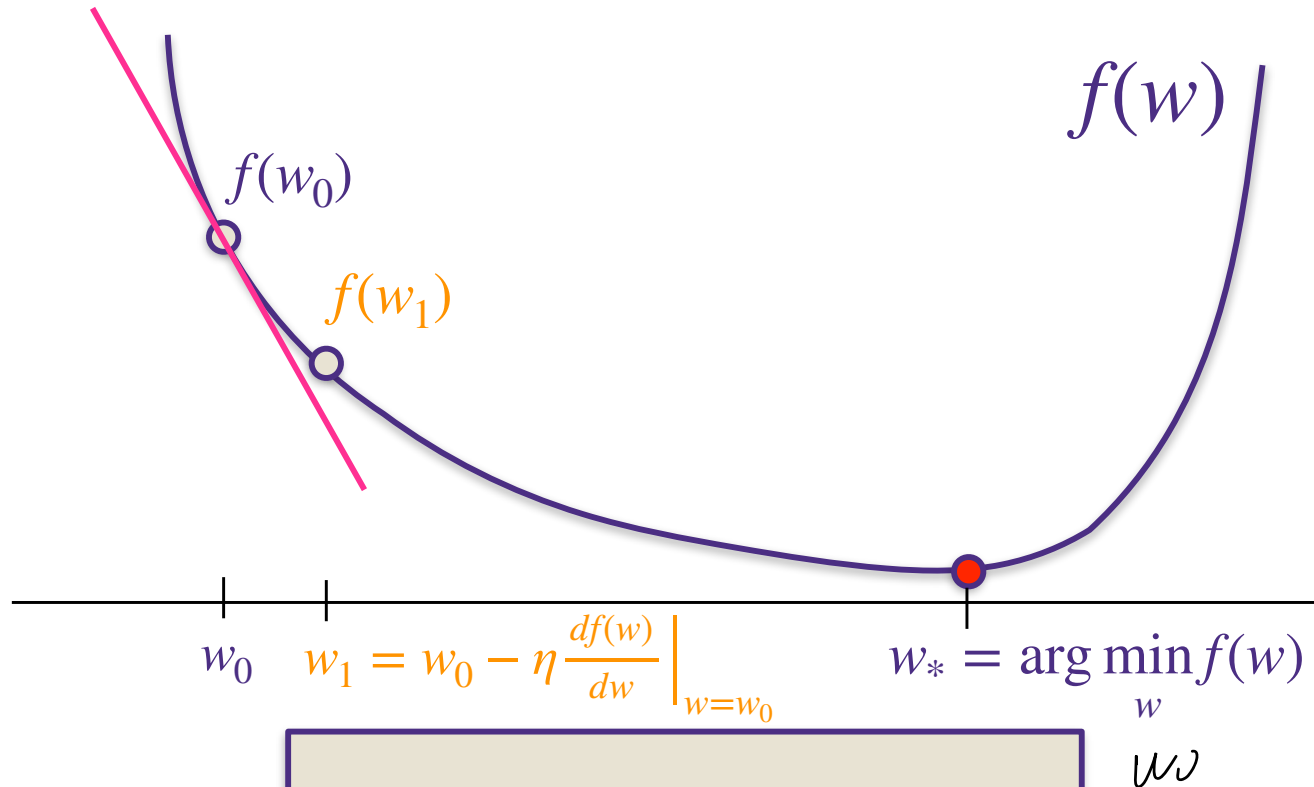
$$f(w_0) + (w - w_0) \frac{df(w)}{dw} \Big|_{w=w_0}$$

is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw} \Big|_{w=w_0}$ then

$f(w_0) - \eta \left(\frac{df(w)}{dw} \Big|_{w=w_0} \right)^2$
is very close to $f(w_1) < f(w_0)$



Gradient descent

For $k=0,1,2,3,\dots$

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw} \Big|_{w=w_k}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

$$f(w_0) + (w - w_0) \frac{df(w)}{dw} \Big|_{w=w_0}$$

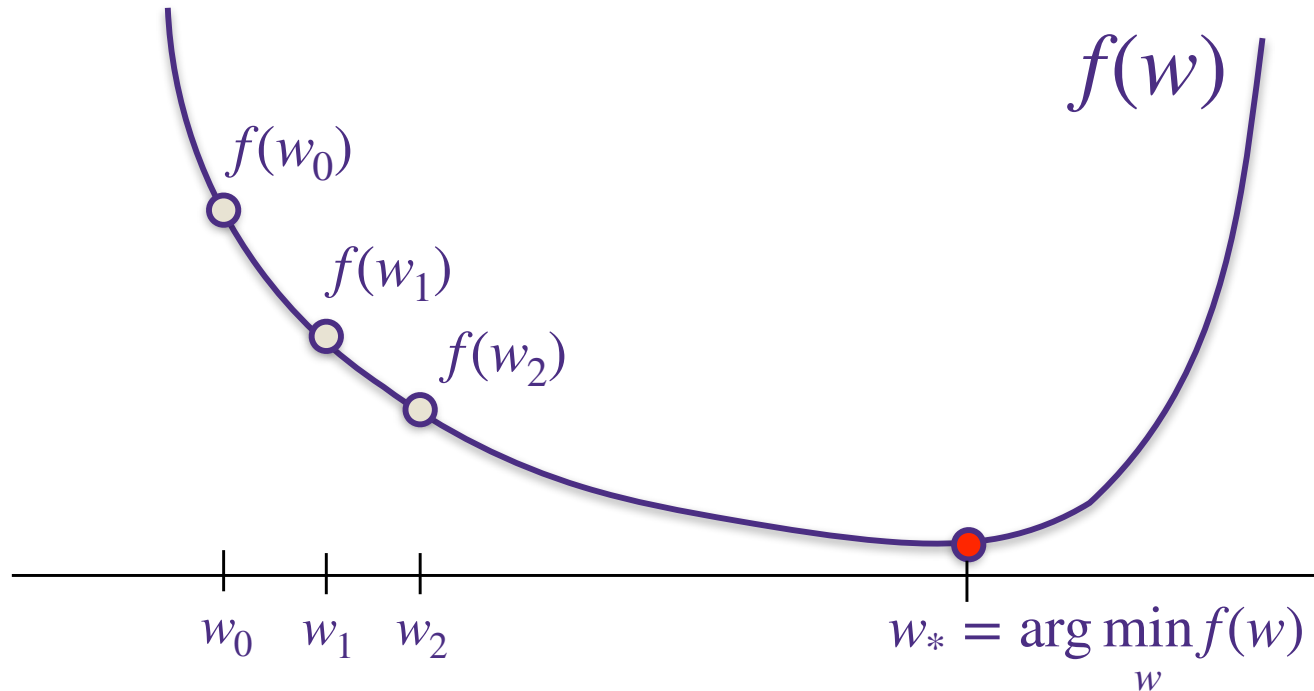
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw} \Big|_{w=w_0}$ then

$$f(w_0) - \eta \left(\frac{df(w)}{dw} \Big|_{w=w_0} \right)^2$$

is very close to $f(w_1) < f(w_0)$



Gradient descent

For $k=0,1,2,3,\dots$

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw} \Big|_{w=w_k}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

$$f(w_0) + (w - w_0) \frac{df(w)}{dw} \Big|_{w=w_0}$$

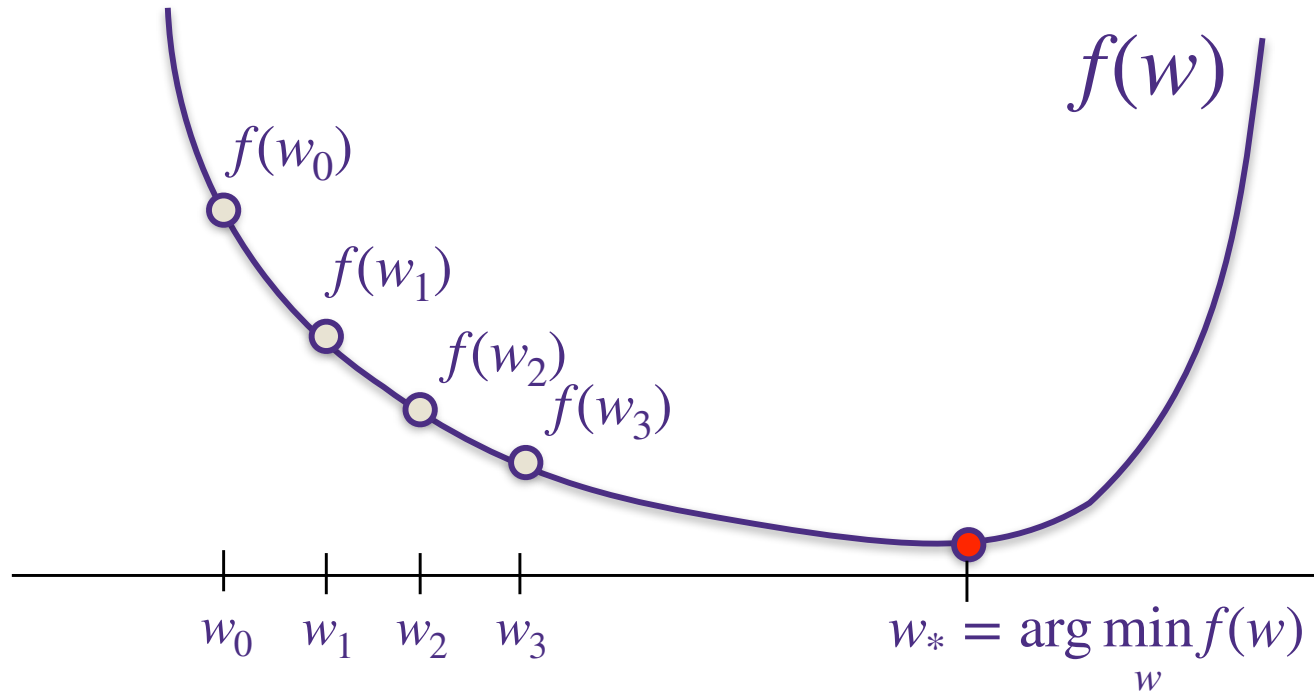
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw} \Big|_{w=w_0}$ then

$$f(w_0) - \eta \left(\frac{df(w)}{dw} \Big|_{w=w_0} \right)^2$$

is very close to $f(w_1) < f(w_0)$



Gradient descent

For $k=0,1,2,3,\dots$

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw} \Big|_{w=w_k}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

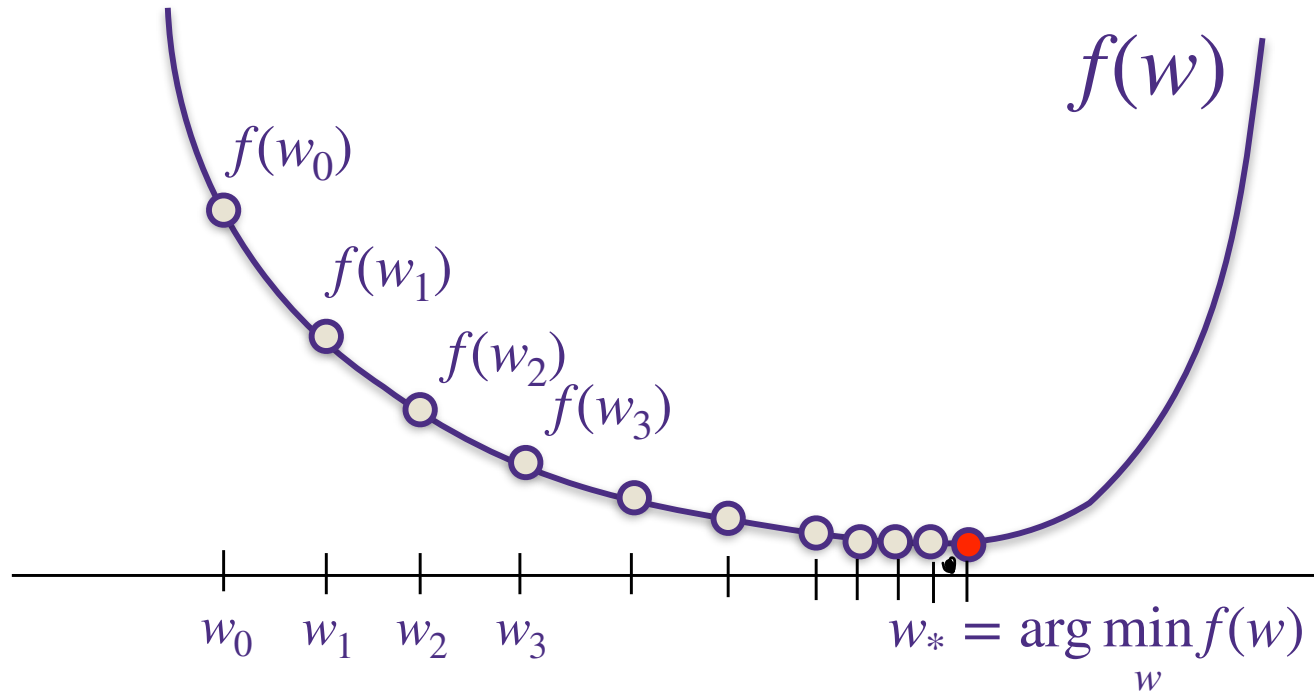
$$f(w_0) + (w - w_0) \frac{df(w)}{dw} \Big|_{w=w_0}$$

is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw} \Big|_{w=w_0}$ then

$f(w_0) - \eta \left(\frac{df(w)}{dw} \Big|_{w=w_0} \right)^2$
is very close to $f(w_1) < f(w_0)$



Gradient descent

For $k=0,1,2,3,\dots$

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw} \Big|_{w=w_k}$$

Note that as $k \rightarrow \infty$ we have $\frac{df(w)}{dw} \Big|_{w=w_k} \rightarrow 0$

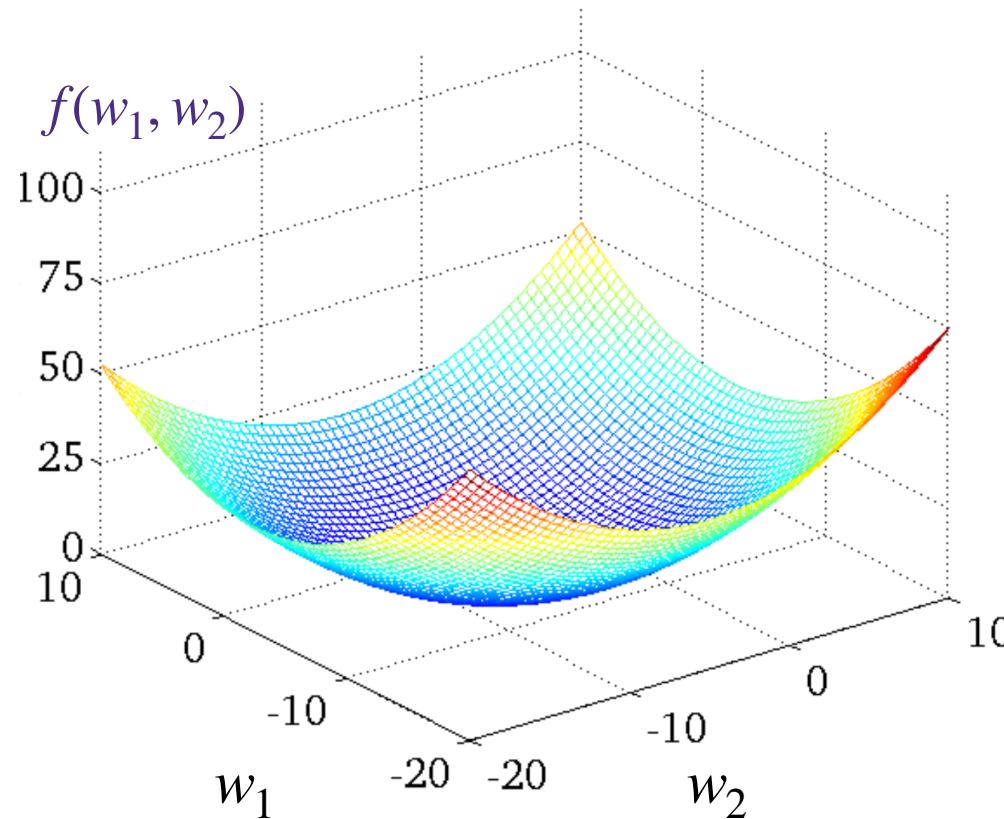
Running example: linear regression

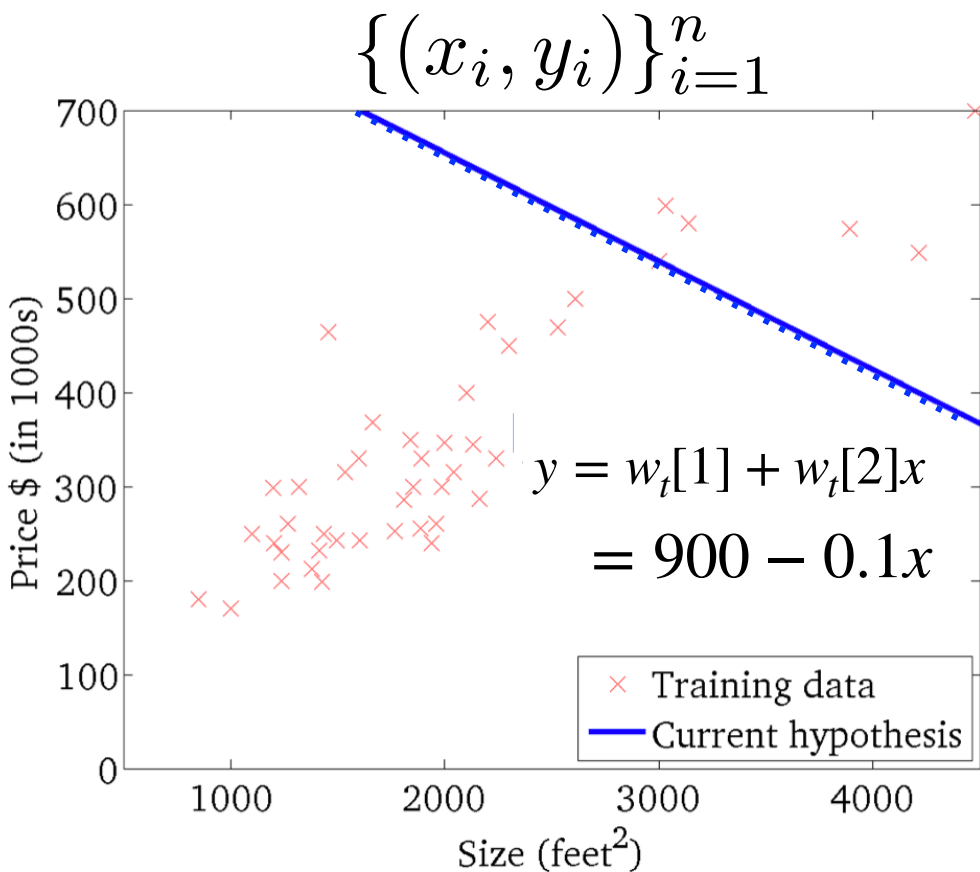
- **Given data:** $\{(x_i, y_i)\}_{i=1}^n$ $x_i \in \mathbb{R}^d$ $y_i \in \mathbb{R}$
- **Learning model parameters:**

$$\hat{w}_{\text{LS}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\|\mathbf{y} - \mathbf{X}w\|_2^2}_{f(w)}$$

- **Gradient descent:**

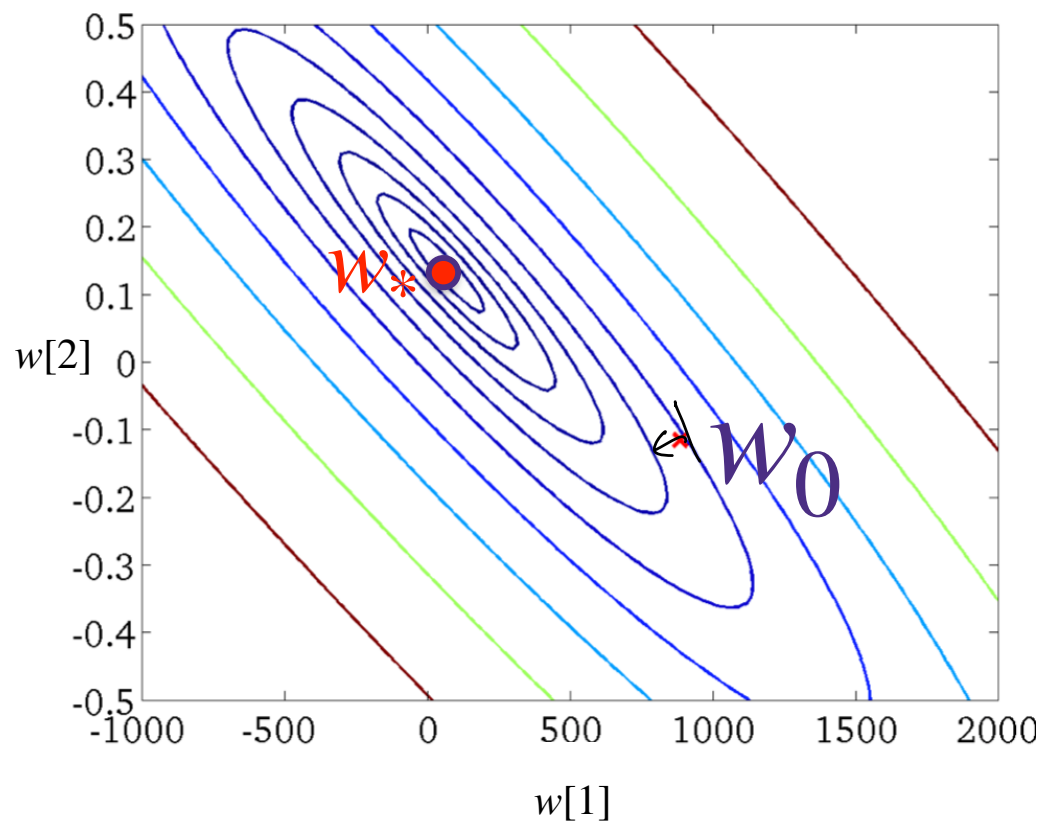
- Initialize: $w_0 = 0$
- **For** $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$





Evolution of the predictor

- $w_0 = (900, -0.1)$
- For $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



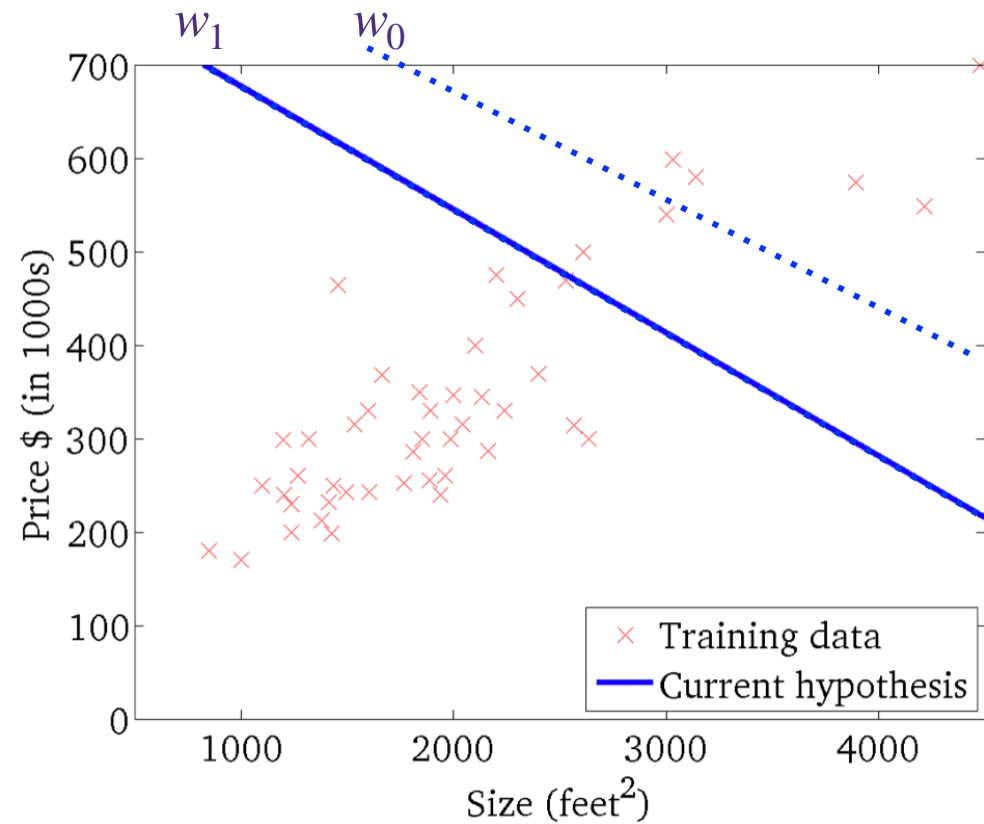
GD dynamics in the Parameter space

- Which direction will the GD move?

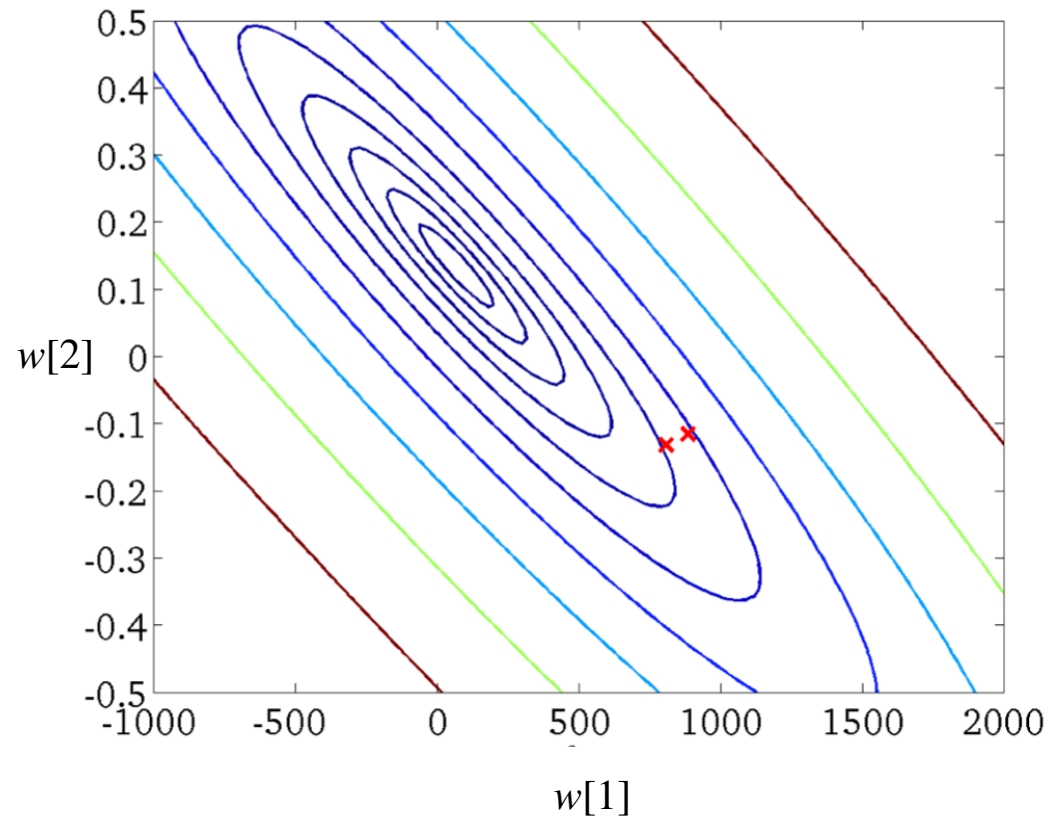
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

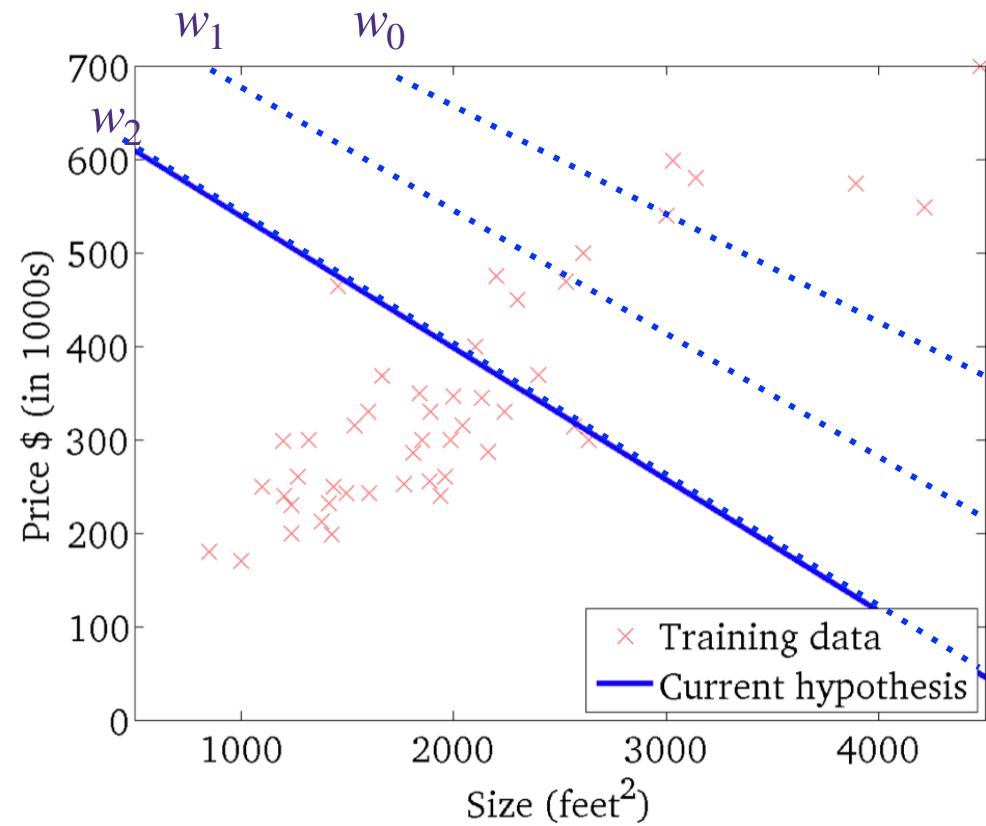


GD dynamics in the Parameter space

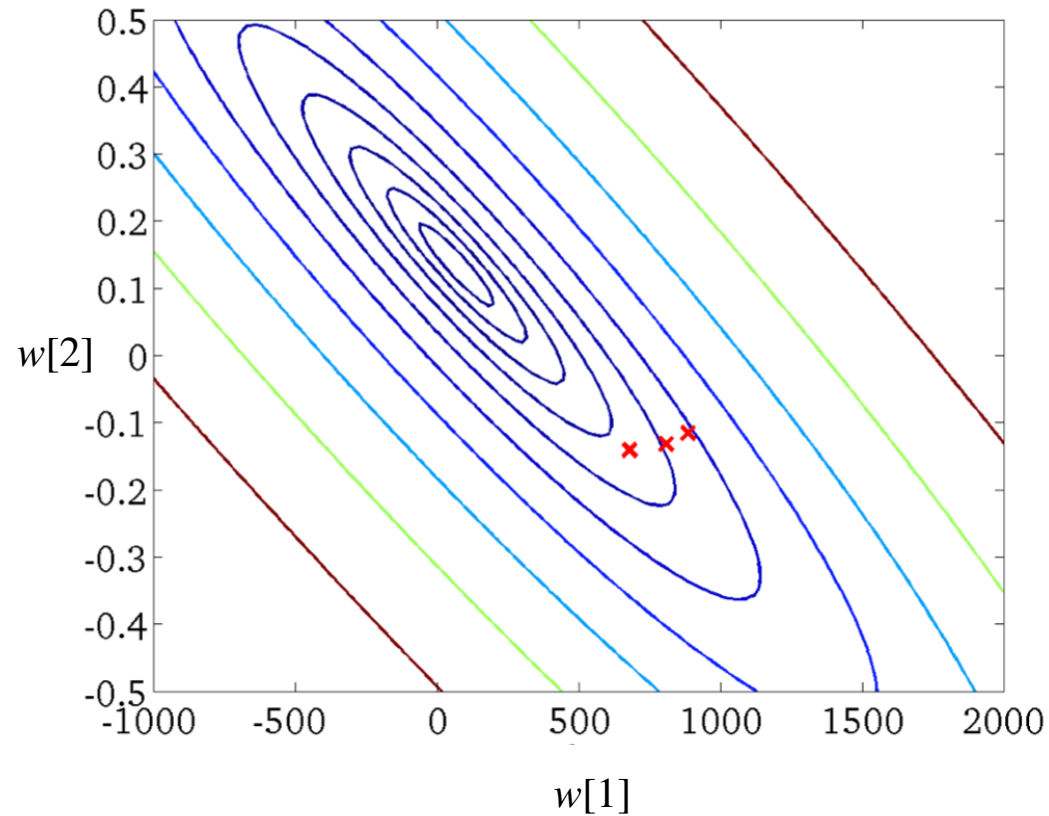
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

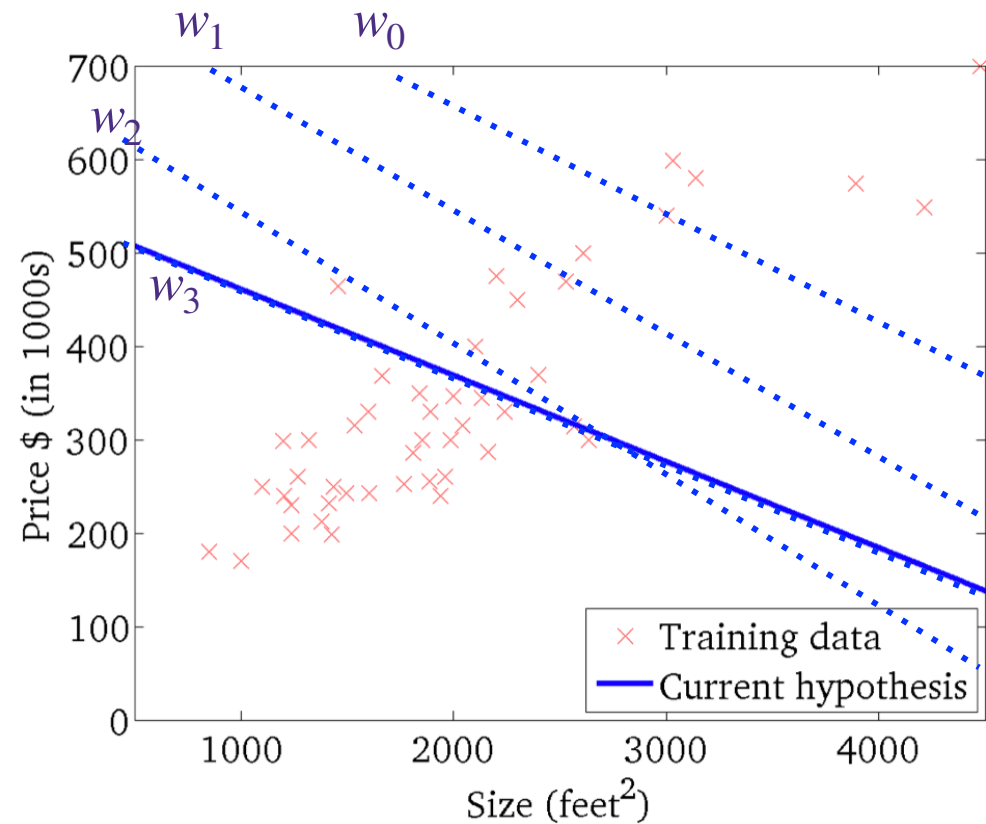


GD dynamics in the Parameter space

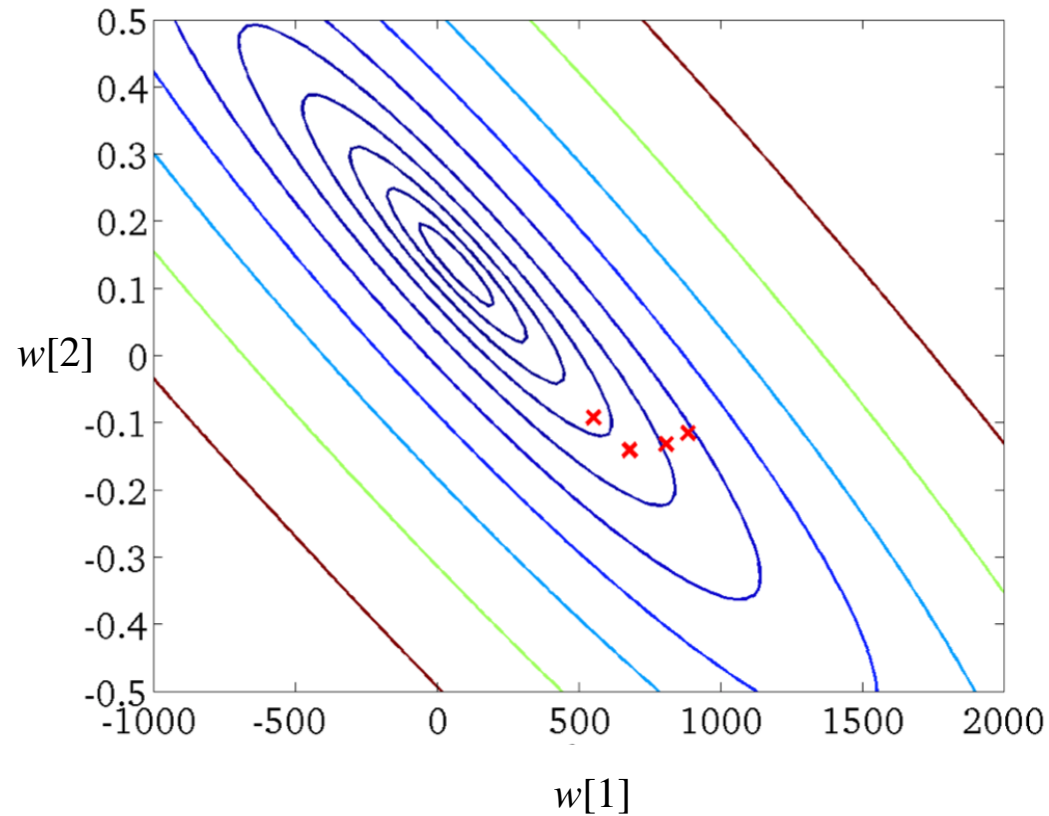
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

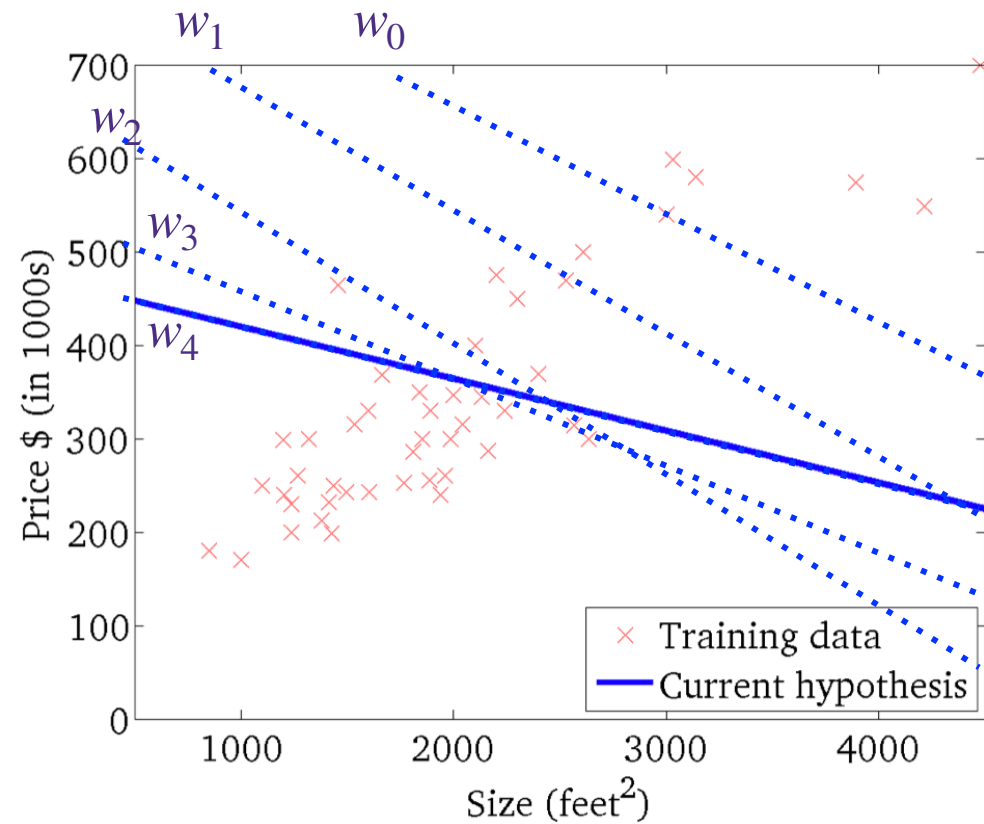


GD dynamics in the Parameter space

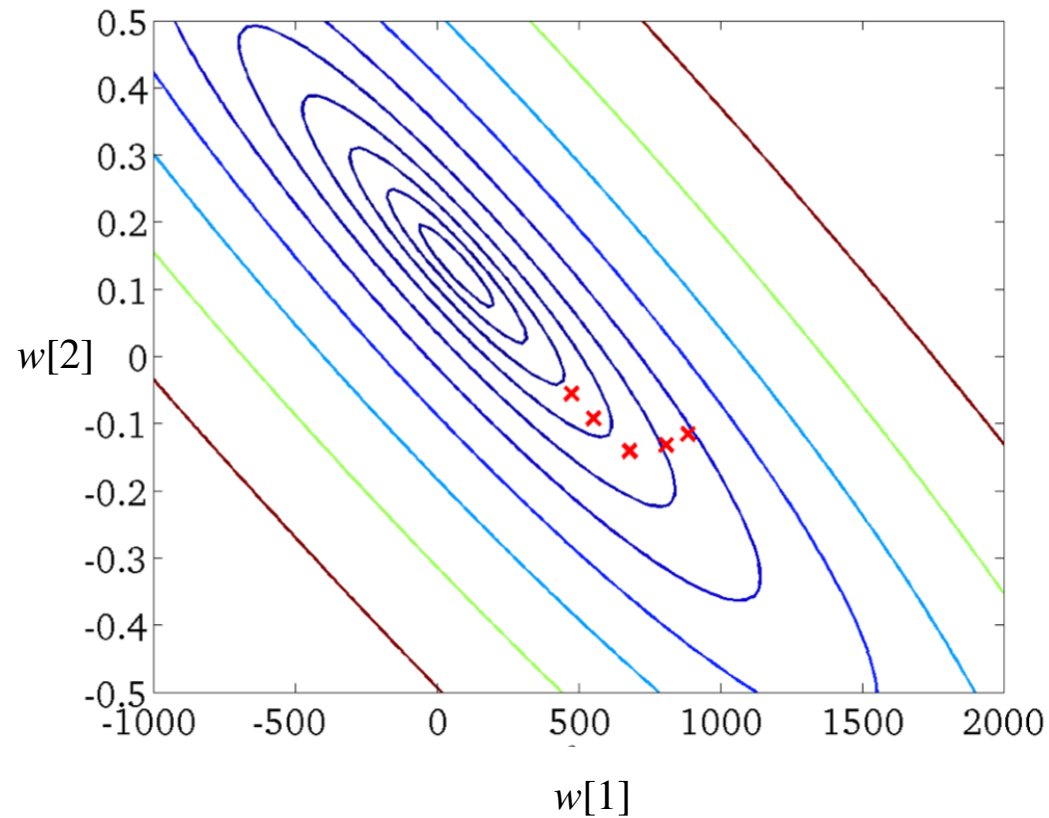
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

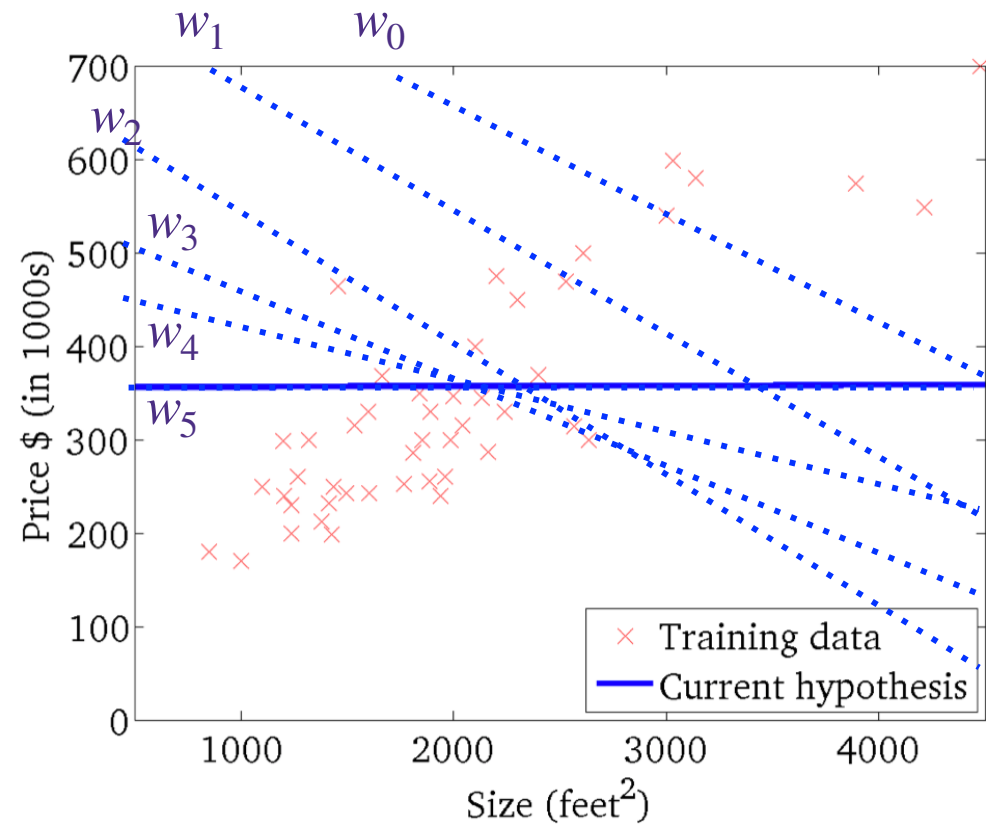


GD dynamics in the Parameter space

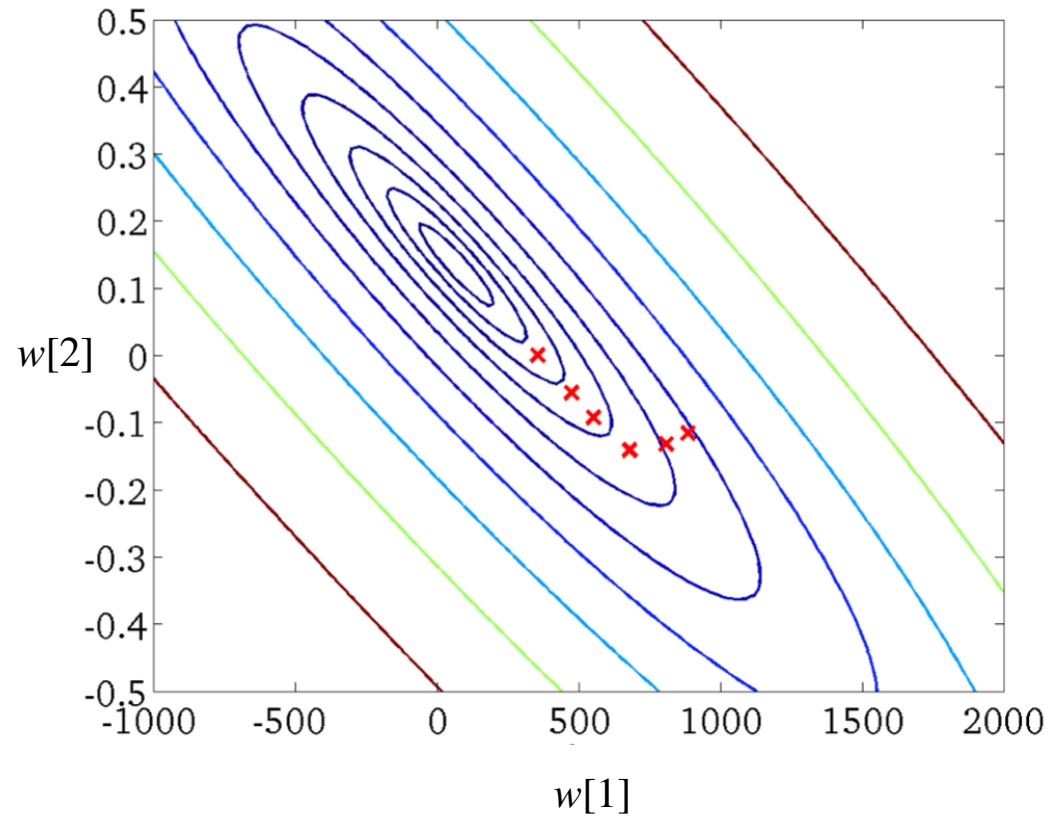
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

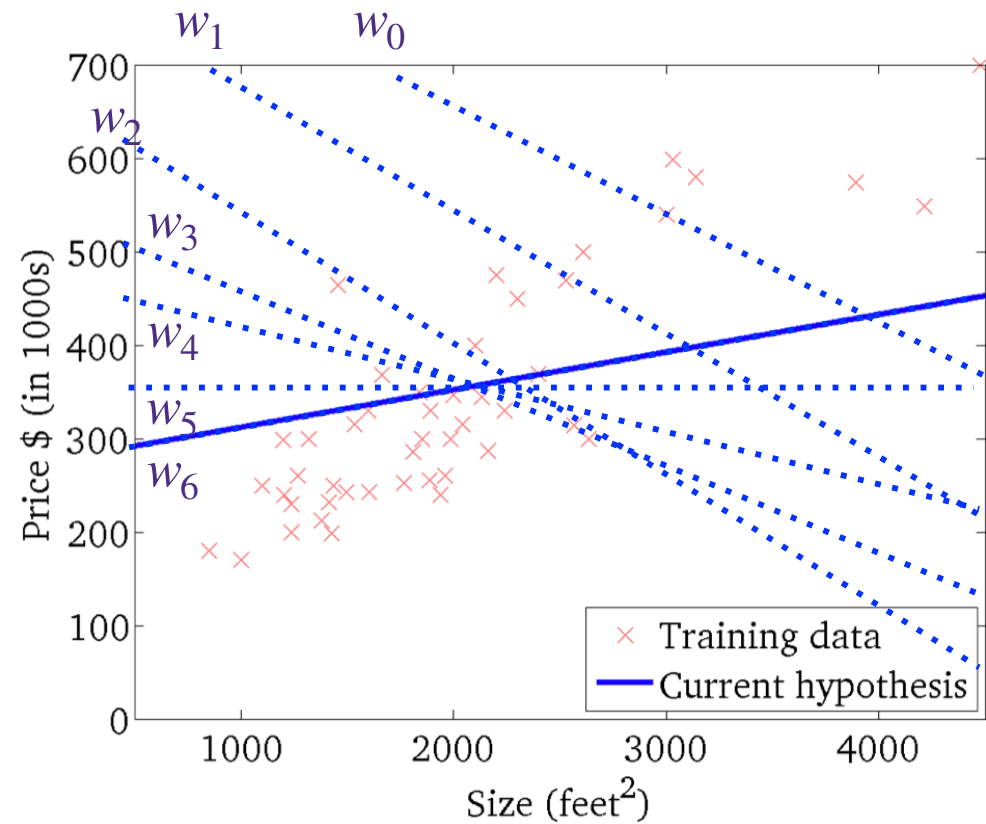


GD dynamics in the Parameter space

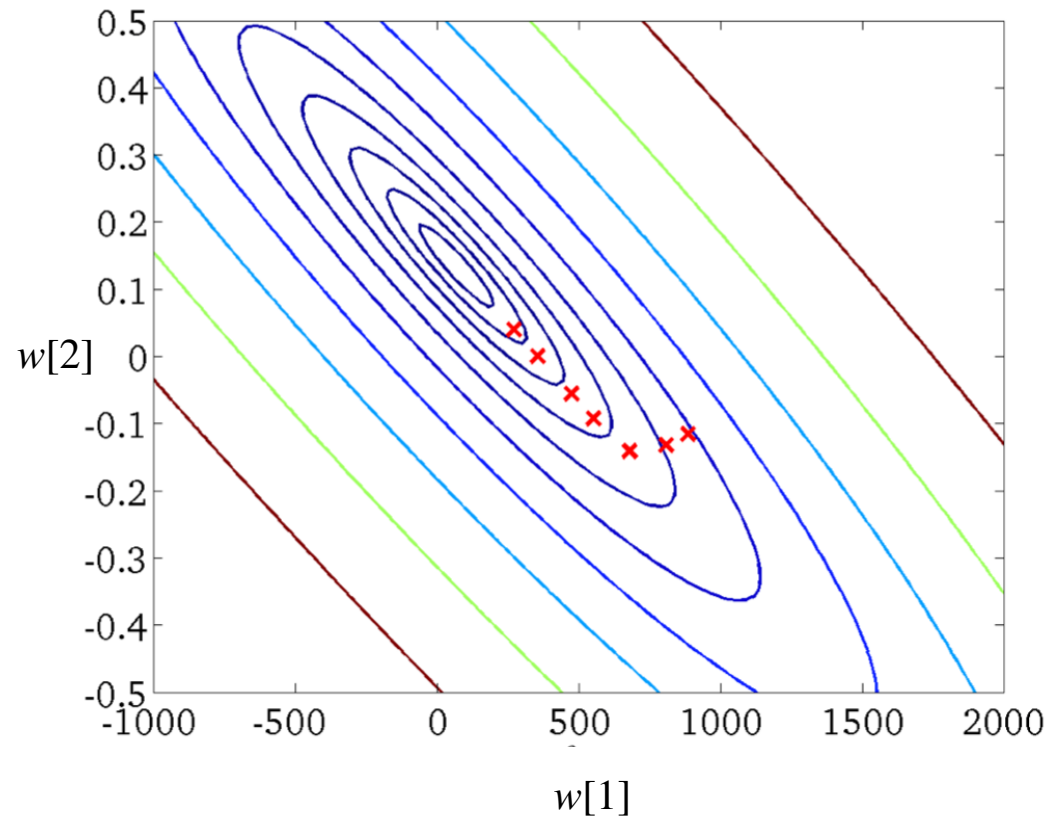
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

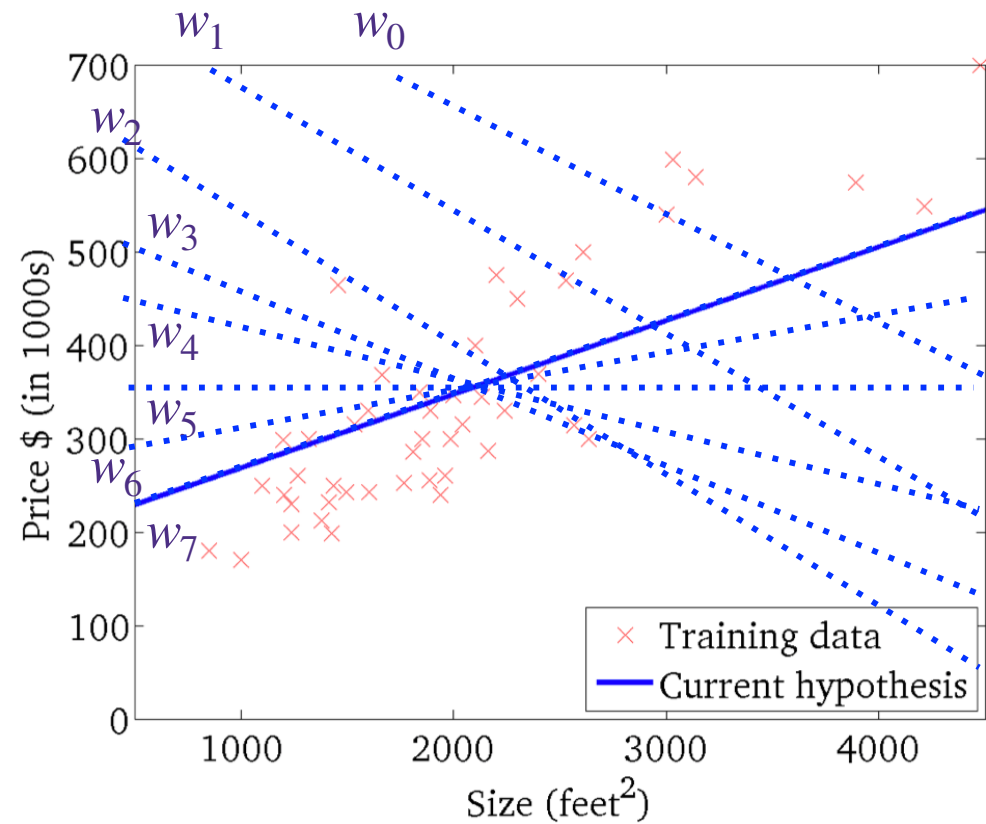


GD dynamics in the Parameter space

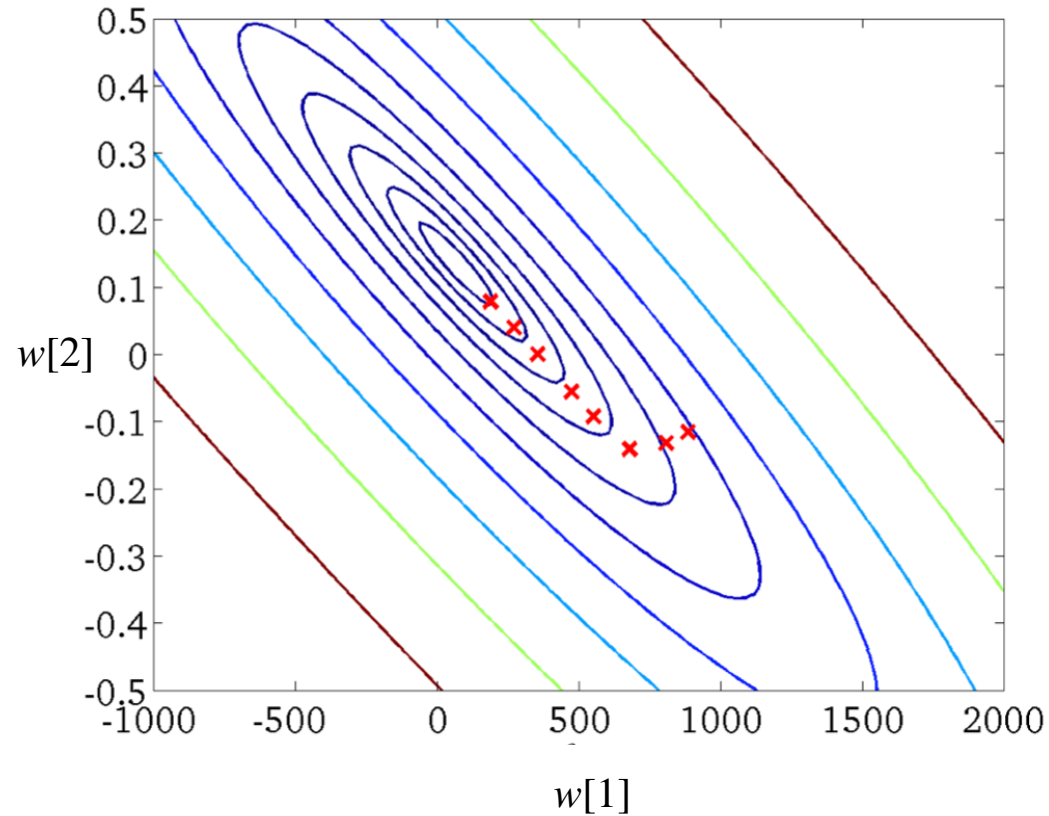
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor

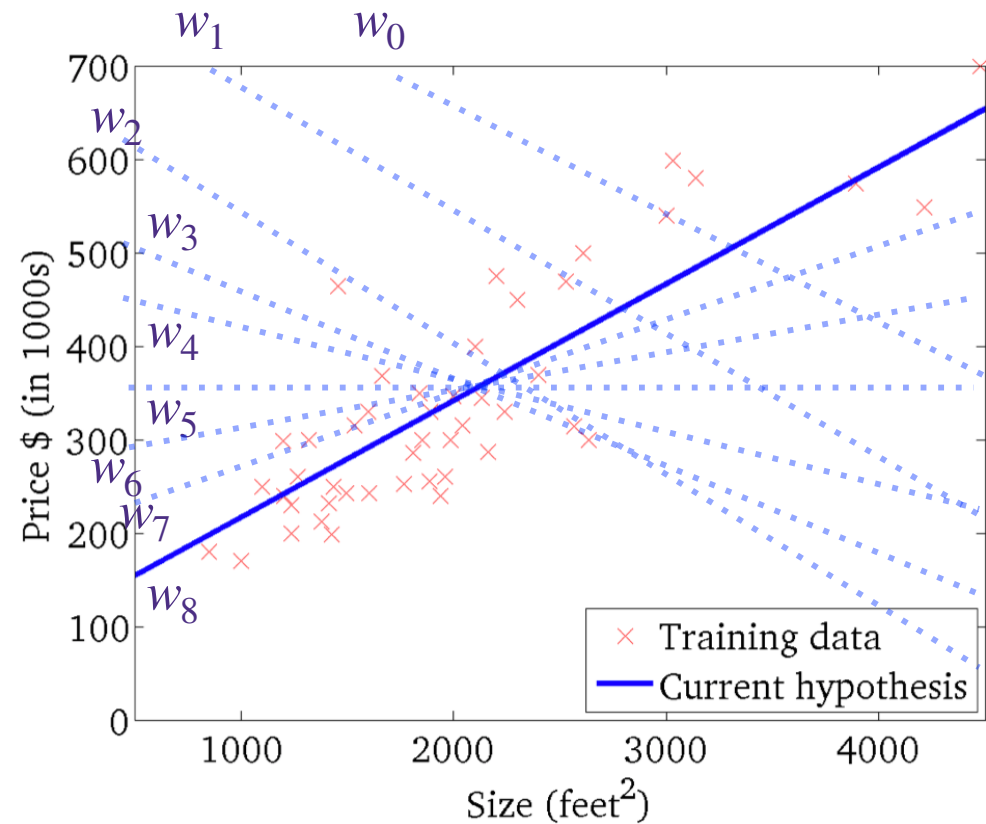


GD dynamics in the Parameter space

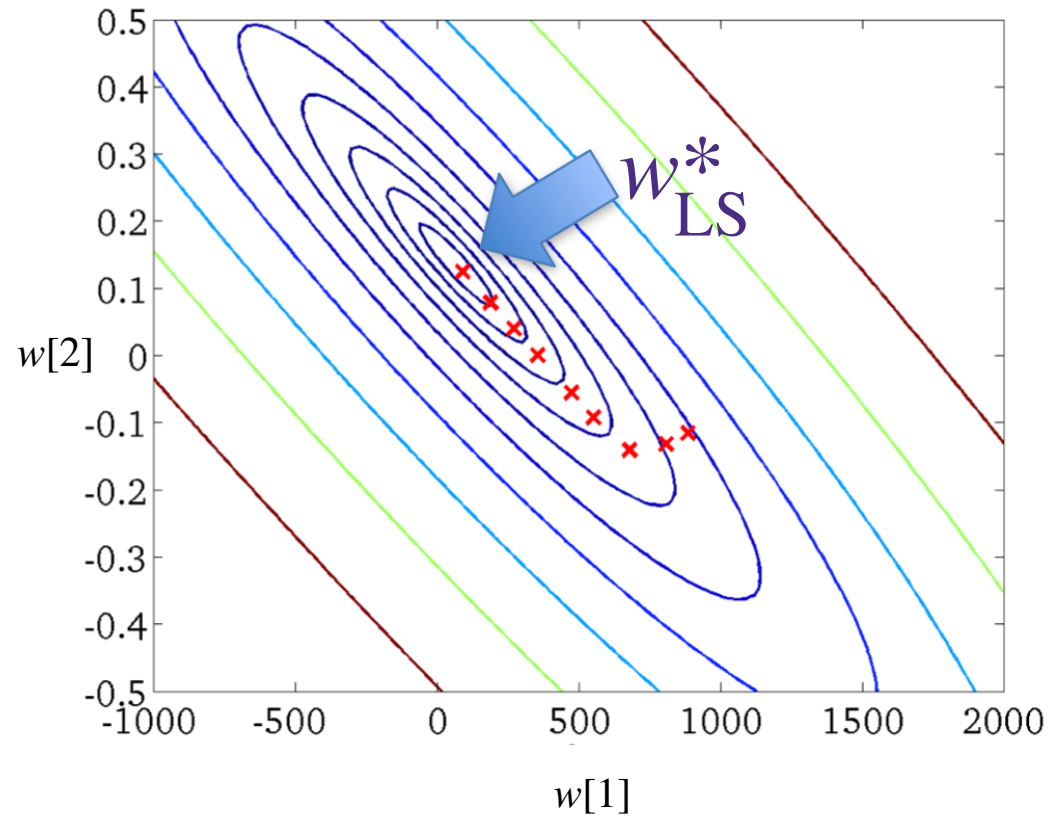
- $w_0 = (900, -0.1)$

- For $t=0,1,2,\dots$

- $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



Evolution of the predictor



GD dynamics in the Parameter space

Gradient descent for linear regression

- In this example of linear regression, we can derive exactly the gradient descent trajectory
- Initialize: $w_0 = 0$
- **For** $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

$$\nabla f(w_t) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t)$$

For linear regression, we have

$$\hat{w}_{\text{LS}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\|\mathbf{y} - \mathbf{X}w\|_2^2}_{f(w)}$$

Gradient descent for linear regression

- In this example of linear regression, we can derive exactly the gradient descent trajectory
- Initialize: $w_0 = 0$
- **For** $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

For linear regression, we have

$$\hat{w}_{\text{LS}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\|y - Xw\|_2^2}_{f(w)}$$

$$\nabla f(w_t) = -2X^T(y - Xw_t)$$

$$w_{t+1} = w_t + \eta 2X^T(y - Xw_t) = (I - 2\eta X^T X)w_t + 2\eta X^T y$$

Gradient descent for linear regression

- In this example of linear regression, we can derive exactly the gradient descent trajectory
- Initialize: $w_0 = 0$
- **For** $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

For linear regression, we have

$$\hat{w}_{\text{LS}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\|y - Xw\|_2^2}_{f(w)}$$

$$\nabla f(w_t) = -2X^T(y - Xw_t)$$

$$w_{t+1} = w_t + \eta 2X^T(y - Xw_t) = (I - 2\eta X^T X)w_t + 2\eta X^T y$$

Let the least-squares solution be $w^* = (X^T X)^{-1} X^T y$

$$\begin{aligned} \underbrace{w_{t+1} - w^*} &= (I - 2\eta X^T X)w_t + 2\eta X^T y - w^* \\ &= (I - 2\eta X^T X)(w_t - w^*) + 2\eta X^T y - 2\eta X^T X w^* \\ &= \underbrace{(I - 2\eta X^T X)(w_t - w^*)} \end{aligned}$$

Gradient descent for linear regression

- Initialize: $w_0 = 0$
- For $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

For linear regression, we have

$$\hat{w}_{\text{LS}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\|y - Xw\|_2^2}_{f(w)}$$

$$\nabla f(w_t) = -2X^T(y - Xw_t)$$

$$w_{t+1} = w_t + \eta 2X^T(y - Xw_t) = (I - 2\eta X^T X)w_t + 2\eta X^T y$$

Let the least-squares solution be $w^* = (X^T X)^{-1} X^T y$

$$\begin{aligned} w_{t+1} - w^* &= (I - 2\eta X^T X)w_t + 2\eta X^T y - w^* \\ &= (I - 2\eta X^T X)(w_t - w^*) + 2\eta X^T y - 2\eta X^T X w^* \\ &= \underbrace{(I - 2\eta X^T X)}_{\text{in } 1-d'} \underbrace{(w_t - w^*)}_{\text{in } 1-d'} \end{aligned}$$

in 1-d',
 $I - 2\eta X^T X$
 $\Rightarrow 0 < 1 - 2\eta \cdot \lambda^2 < 1$
 $\eta = \frac{2}{2\lambda^2}$

$t \rightarrow \infty, w_t \rightarrow w^*$

How do you choose step size?

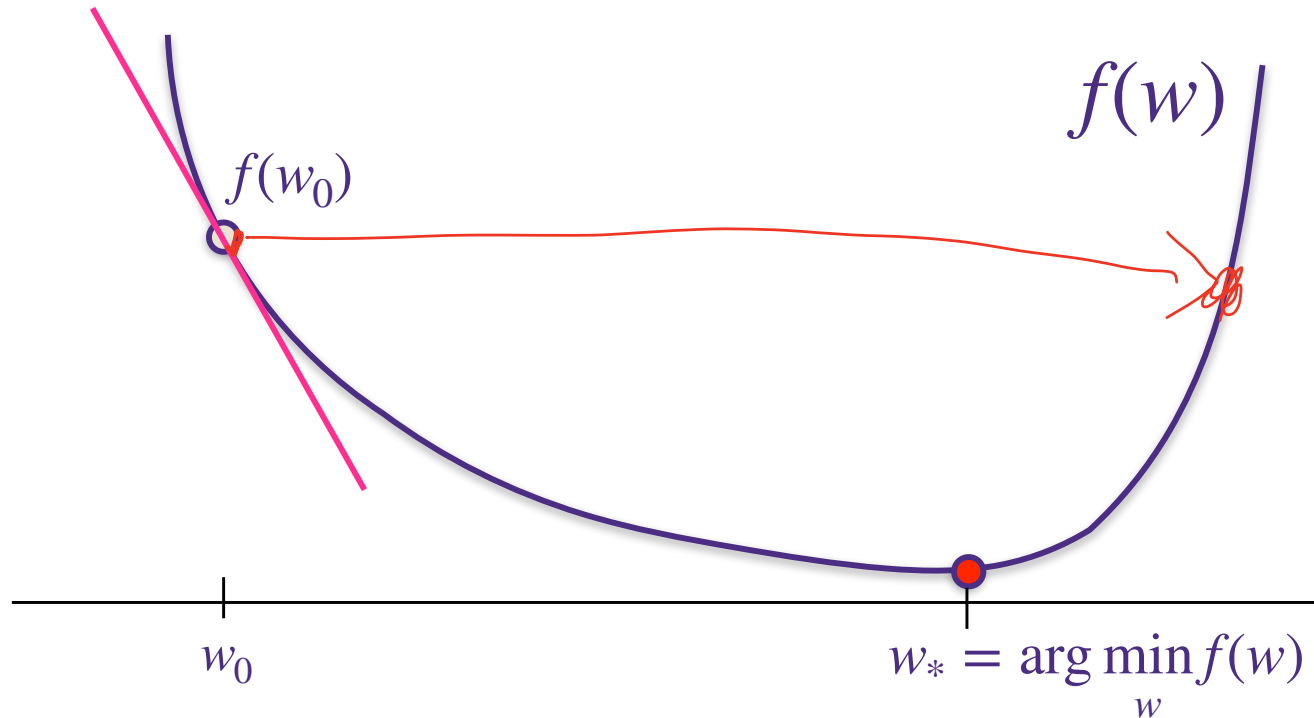
Let w_0 be an initial guess. How can we improve this solution?

Taylor series approximation:

For w very close to w_0 we have

$$f(w_0) + (w - w_0) \left. \frac{df(w)}{dw} \right|_{w=w_0}$$

is very close to $f(w)$



If η too big, does not converge!

If η too small, converges very, very slowly.

In practice: choose the largest value of η that converges (guess and check)

Stochastic Gradient Descent

Machine Learning Problems

- Given data:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n} \sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

time complexity $\mathcal{O}(n)$

Machine Learning Problems

- Given data:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters:

$$\sum_{i=1}^n \ell_i(w)$$

time complexity
per iteration
 $O(1)$

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n} \sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \Big|_{w=w_t}$$

I_t drawn uniform at
random from $\{1, \dots, n\}$

$$\mathbb{E}[\nabla \ell_{I_t}(w)] = \nabla_w \left(\frac{1}{n} \sum_{i=1}^n \ell_i(w) \right)$$

$I_t \sim \text{Unif}(\{1, \dots, n\})$

Machine Learning Problems

- Learning a model's parameters:

$$\sum_{i=1}^n \ell_i(w)$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \Big|_{w=w_t}$$

I_t drawn uniform at
random from $\{1, \dots, n\}$

$$\mathbb{E}[\nabla \ell_{I_t}(w)] = \nabla \ell(w)$$

Mini-batch SGD

$$1 \leq B \leq n$$

often

$$B \ll n$$

- Instead of one iterate, average B stochastic gradient together
- Advantages:
 - Smaller variance: the variance of the stochastic gradient is smaller by a factor of $1/\sqrt{B}$
 - Parallelization: each gradient in the mini-batch can be computed in parallel

$$\|w\|_2^2$$

- If you have regularizer, $\frac{1}{n} \sum_{i=1}^n \ell_i(w) + r(w)$, then update with the stochastic gradient of the loss and gradient of the regularizer

Fully-Connected Neural Networks



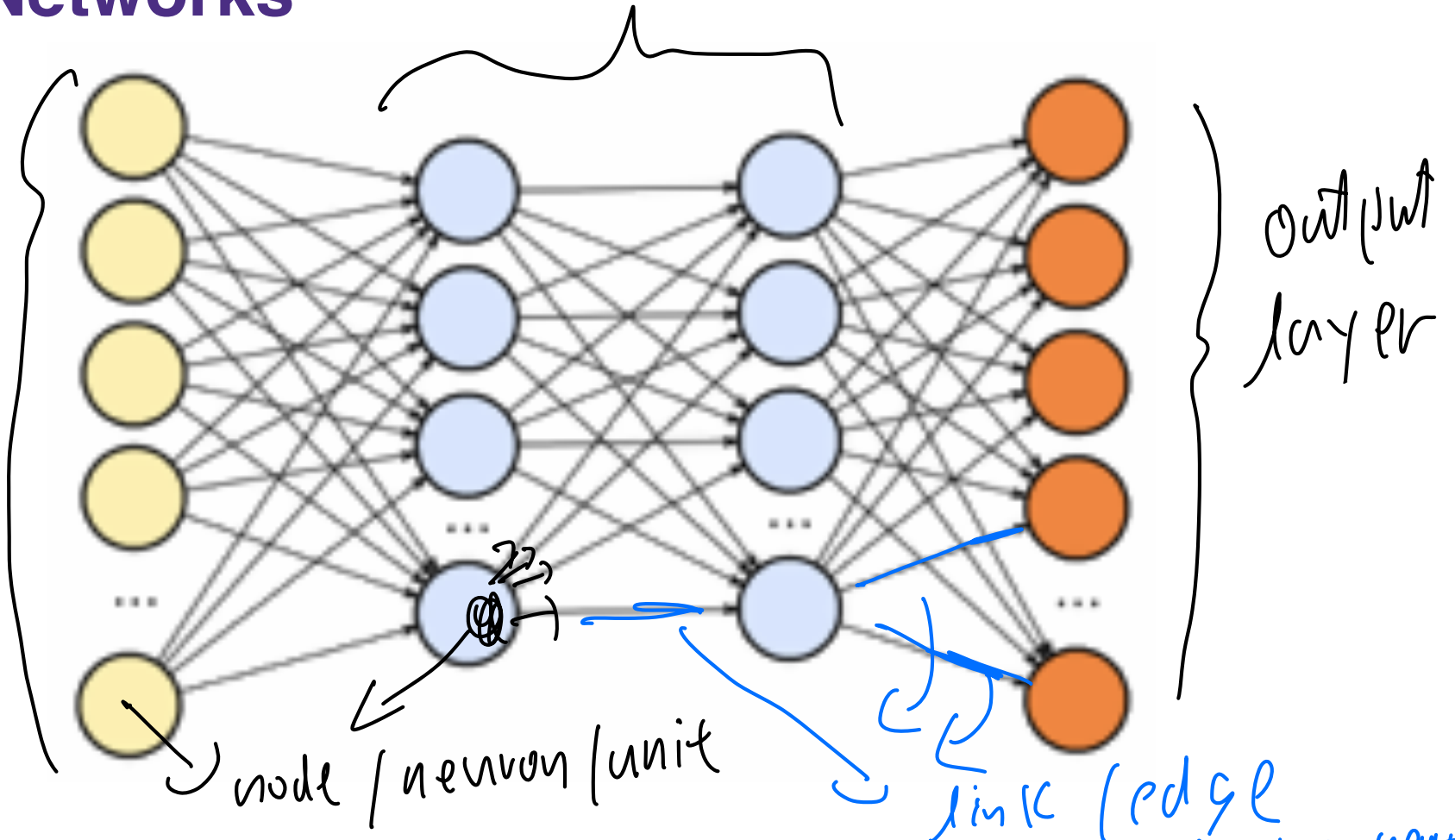
Neural Networks

- Origins: Algorithms that try to mimic the brain.
- Widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence from 10s: state-of-the-art techniques for many applications:
 - Computer Vision
 - Natural language processing: e.g., GPT
 - Speech recognition
 - Decision-making / control problems (AlphaGo, Dota, robots)
- Limited theory:
 - Non-convexity
 - Model are complex but generalization error is small

Neural Networks

intermediate layer

input
layer



each node
1) input
2) activation
3) output

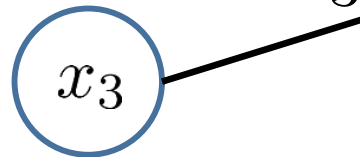
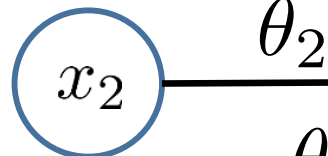
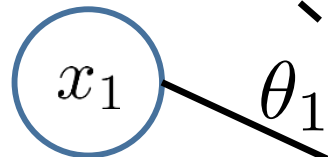
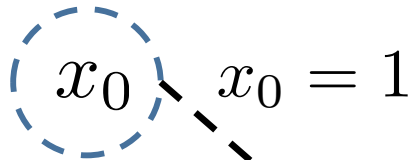
link (edge)
• maps output of a neuron
to the input of a neuron
• each link has
a weight $\in \mathbb{R}$

$\{x^i, y^i\}_{i=1}^n$

Single Node

$\eta \quad x_0=1, \theta_0=2$
 $x_0=2, \theta_0=1$

“bias unit”



$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$\boldsymbol{\theta}^T \mathbf{x} = \theta_0 \cdot x_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \theta_3 \cdot x_3$

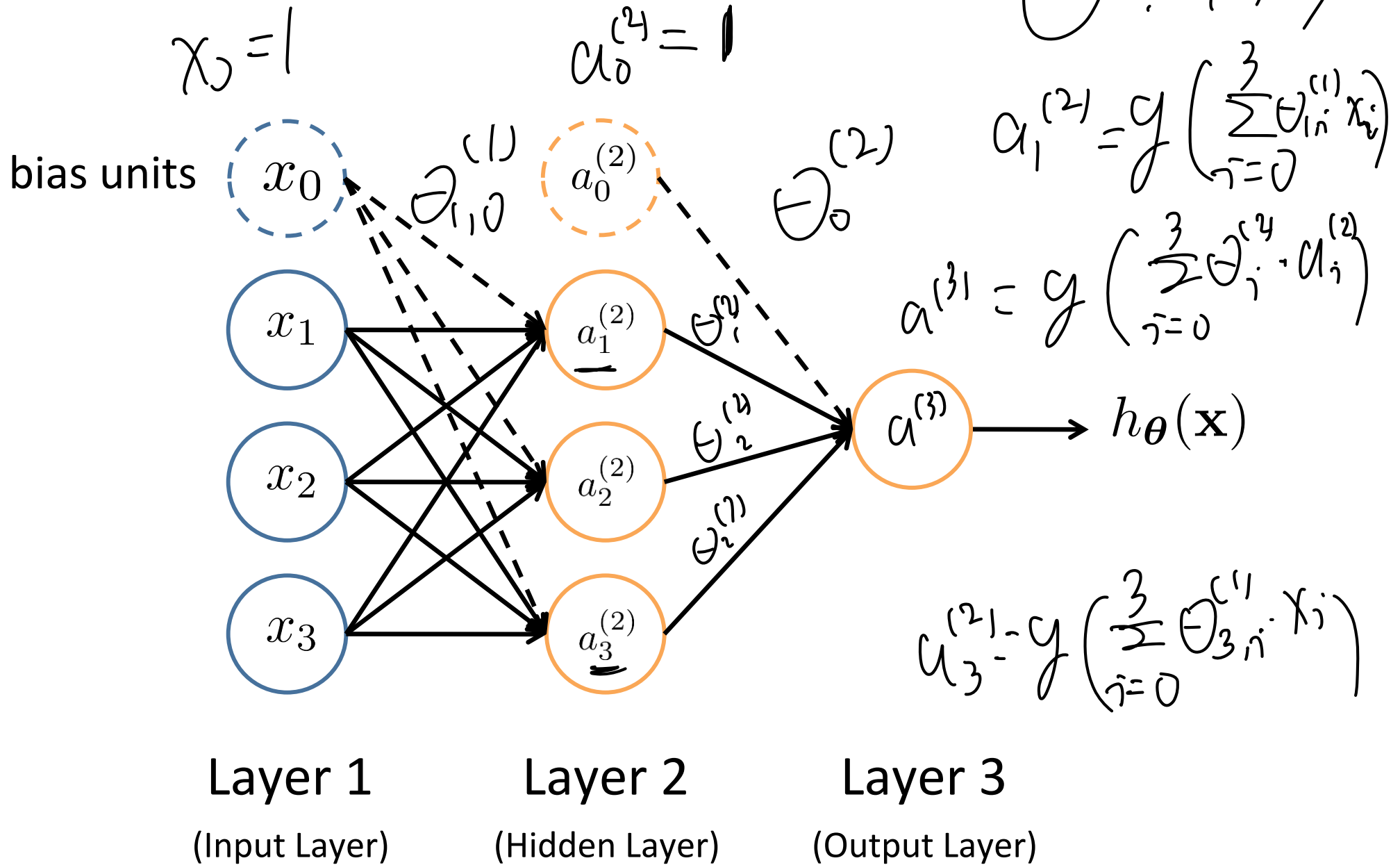
A circle divided vertically, with a summation symbol Σ on the left and an integral symbol $\int$ on the right. Four arrows point into it from the left, labeled θ_0 (dashed), θ_1 , θ_2 , and θ_3 . An arrow points out to the right, labeled $h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x})$.

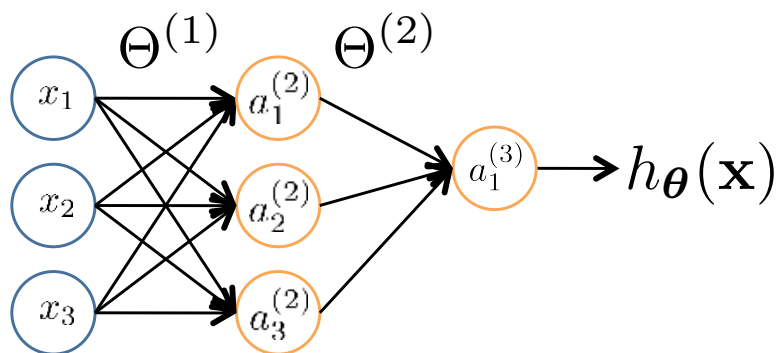
Binary
Logistic
Regression

$$= \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \frac{1}{1 + e^{-z}}$
 from linear $\rightarrow$ non-linear

Neural Network $\Theta : 4 \times 3$





$a_i^{(j)}$ = “activation” of unit i in layer j
 $\Theta^{(j)}$ = weight matrix stores parameters from layer j to layer $j + 1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3)$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3)$$

$$h_{\Theta}(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

If network has s_j units in layer j and s_{j+1} units in layer $j+1$,
 then $\Theta^{(j)}$ has dimension $s_{j+1} \times (s_j+1)$.

$$\Theta^{(1)} \in \mathbb{R}^{3 \times 4} \quad \Theta^{(2)} \in \mathbb{R}^{1 \times 4}$$

(ignore bias)

Multi-layer Neural Network - Binary Classification

$\mathbb{R}^d$

$$\Theta^{(1)}: \mathbb{R}^{m \times d}$$

$a^{(1)} = x$

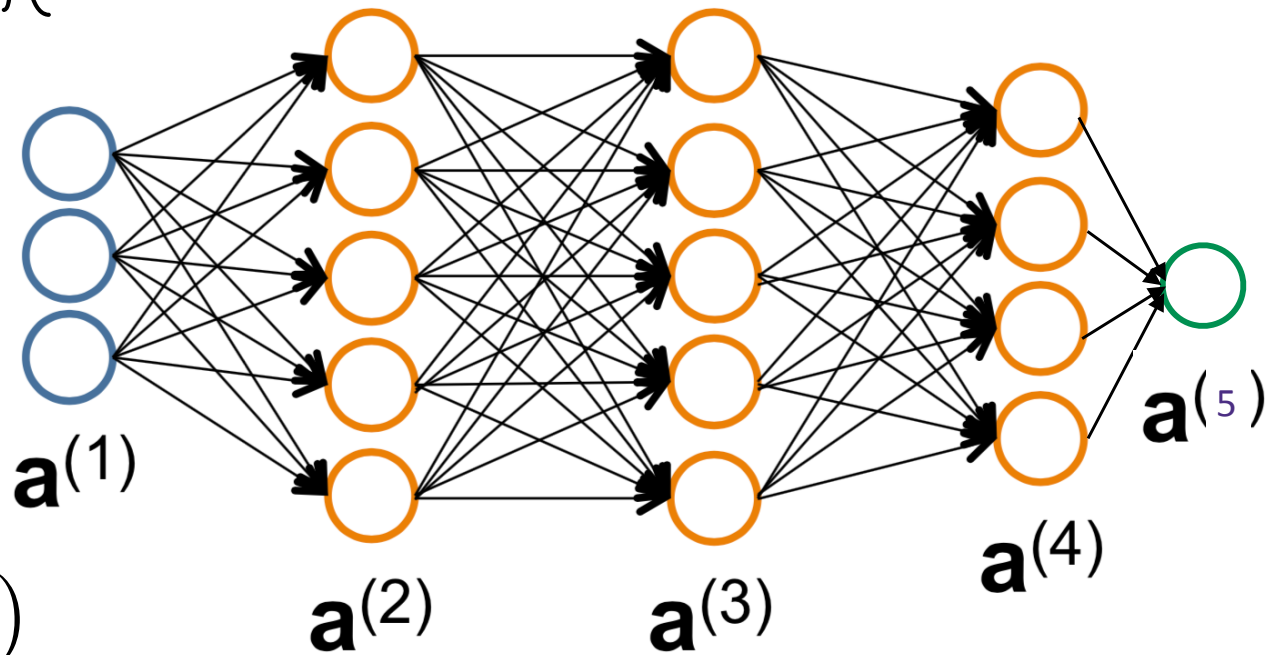
$a^{(2)} = g(\underbrace{\Theta^{(1)} a^{(1)}}_{\mathbb{R}^m})$

$\vdots$

$$a^{(l+1)} = g(\Theta^{(l)} a^{(l)})$$

$\vdots$

$$\hat{y} = g(\Theta^{(L)} a^{(L)})$$



$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Binary
Logistic
Regression

Multi-layer Neural Network - Binary Classification

$$a^{(1)} = x$$

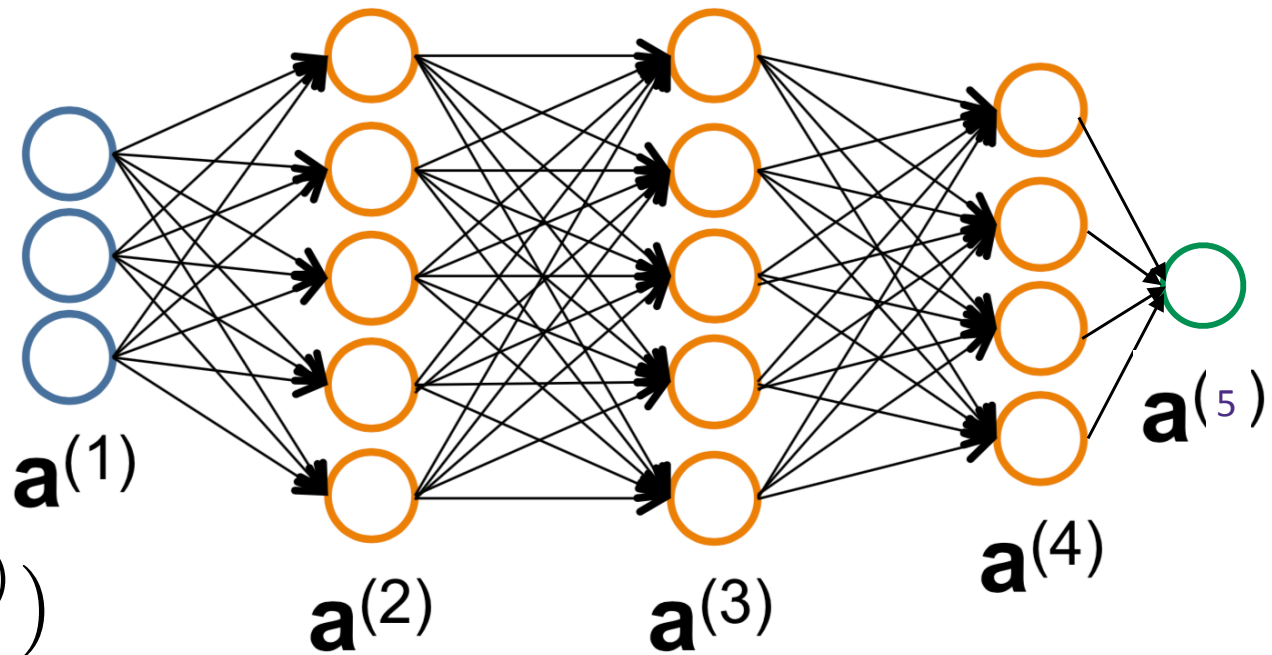
$$a^{(2)} = \sigma(\Theta^{(1)} a^{(1)})$$

⋮

$$a^{(l+1)} = \sigma(\Theta^{(l)} a^{(l)})$$

⋮

$$\hat{y} = g(\Theta^{(L)} a^{(L)})$$



$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$\sigma(z) = \max\{0, z\} \quad g(z) = \frac{1}{1 + e^{-z}}$$

ReLU: rectified linear unit

Binary
Logistic
Regression

Multiple Output Units: One-vs-Rest

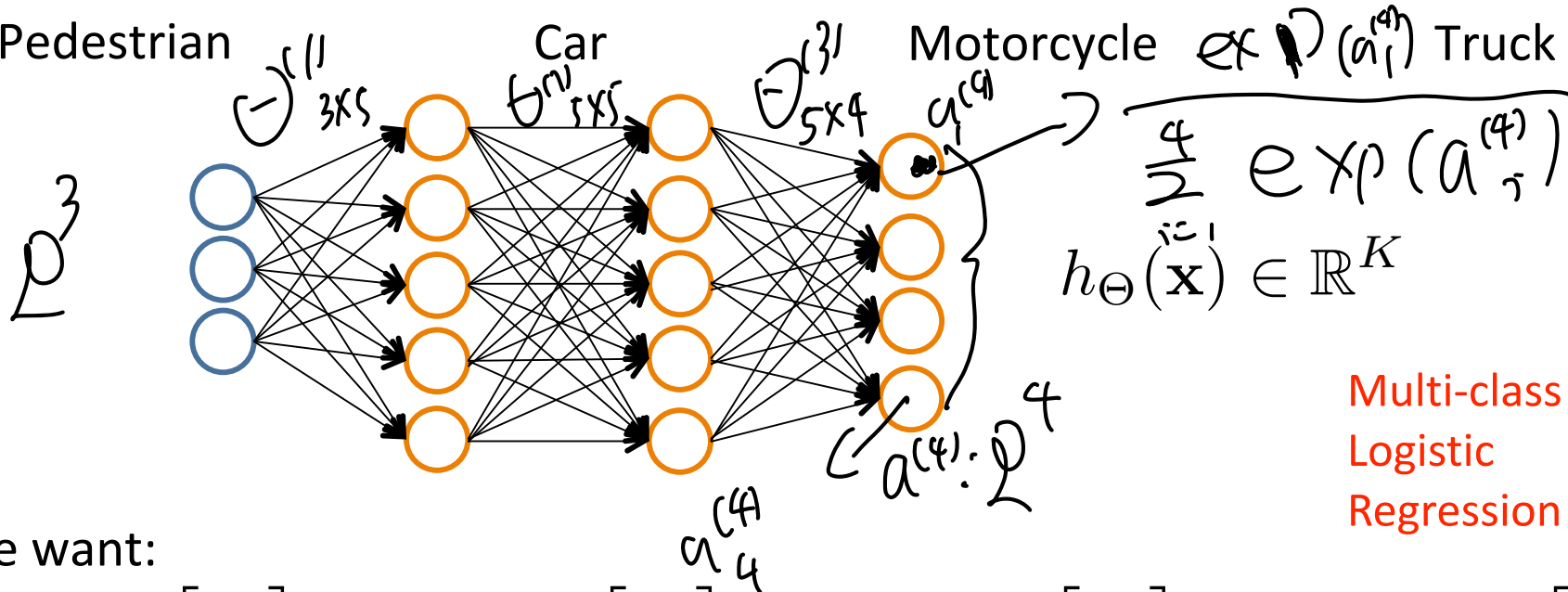


Pedestrian

Car

Motorcycle

Truck



We want:

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

when pedestrian

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

when car

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

when motorcycle

$$h_{\Theta}(\mathbf{x}) \approx \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

when truck

Multi-layer Neural Network - Regression

$$a^{(1)} = x$$

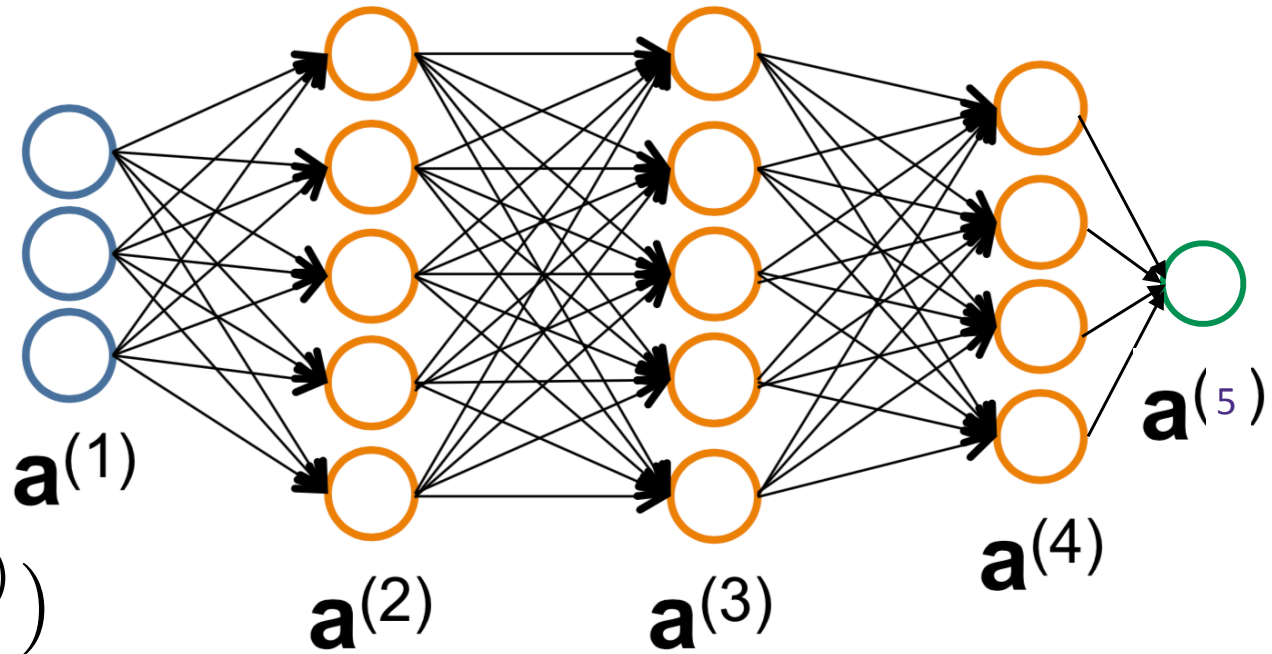
$$a^{(2)} = \sigma(\Theta^{(1)} a^{(1)})$$

$\vdots$

$$a^{(l+1)} = \sigma(\Theta^{(l)} a^{(l)})$$

$\vdots$

$$\hat{y} = \Theta^{(L)} a^{(L)}$$



$$L(y, \hat{y}) = (y - \hat{y})^2$$

$$\sigma(z) = \max\{0, z\}$$

Regression

Neural Network Optimization



Machine Learning Problems

- **Given data:**

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- **Learning a model's parameters:** $\sum_{i=1}^n \ell_i(w)$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Machine Learning Problems

- Given data:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$

Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n} \sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

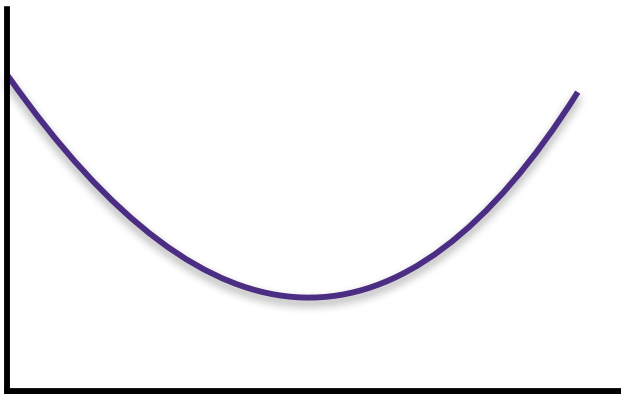
Gradient Descent

Initialize: $w_0 = 0$

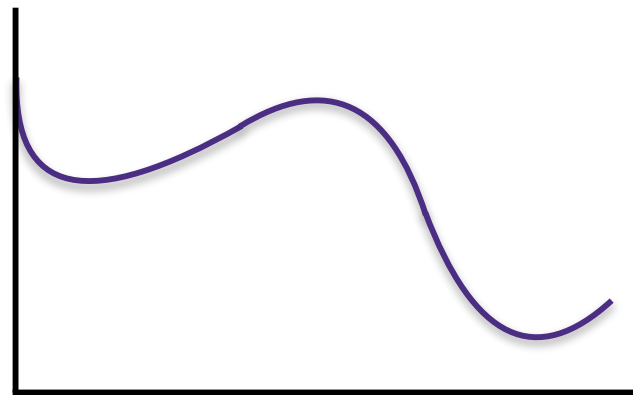
for $t = 1, 2, \dots$

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

Convex Function



Non-convex Function



Sub-Gradient Descent

Initialize: $w_0 = 0$

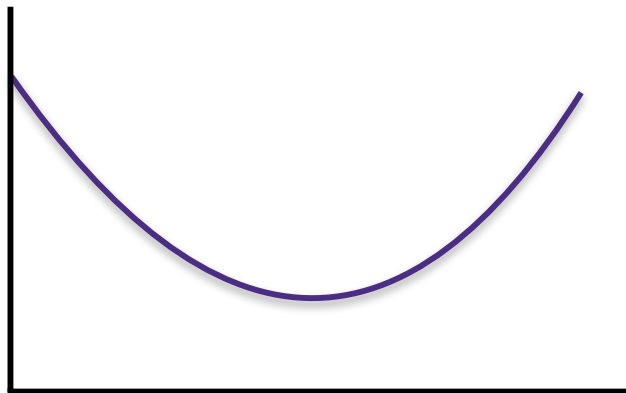
for $t = 1, 2, \dots$

Find any g_t such that $f(y) \geq f(w_t) + g_t^\top (y - w_t)$

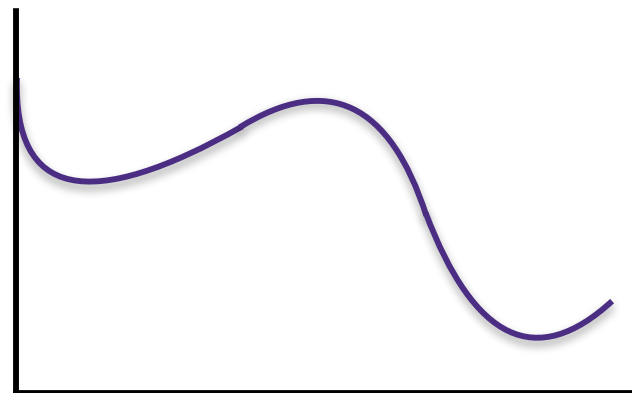
$$w_{t+1} = w_t - \eta g_t$$

g is a subgradient at x if $f(y) \geq f(x) + g^T (y - x)$

Convex Function



Non-convex Function



Machine Learning Problems

- Given data:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters: $\sum_{i=1}^n \ell_i(w)$

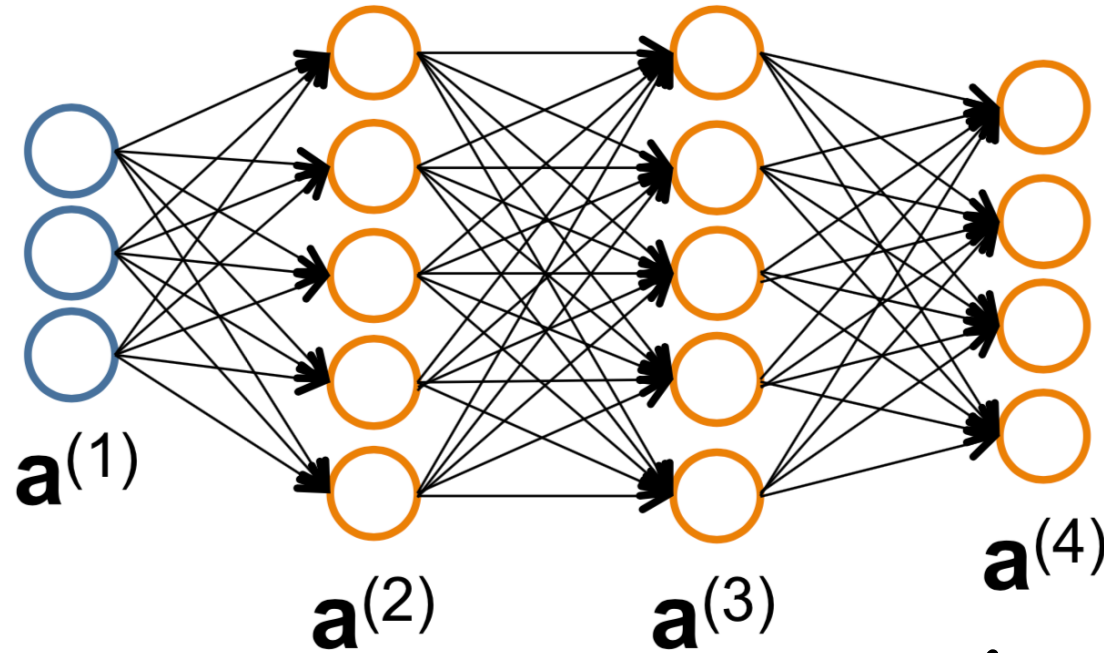
Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n} \sum_{i=1}^n \ell_i(w) \right) \Big|_{w=w_t}$$

Stochastic Gradient Descent:

$$w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) \Big|_{w=w_t} \quad I_t \text{ drawn uniform at random from } \{1, \dots, n\}$$

Gradient Computation on a Graph



Naive computation: node by node

L layers
time complexity $\mathcal{O}(L^2)$

A brief history

$$O(L^2) \rightarrow O(L)$$

- **Back propagation:** the workhorse for training neural networks. An algorithm that for a network with V nodes and E edges calculates that gradient in **linear time** $O(V+E)$.
- The name was introduced by Rumelhart, Hinton, Williams '86. Same idea was rediscovered multiple times. Also mentioned in Werbos' thesis '74 in the context of neural networks.
- **Control theory:** Kelly '60, Bryson '61 [**dynamic programming**]
- **Theoretical computer science:** Baur-Strassen lemma '83 [**algebraic circuits**]

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

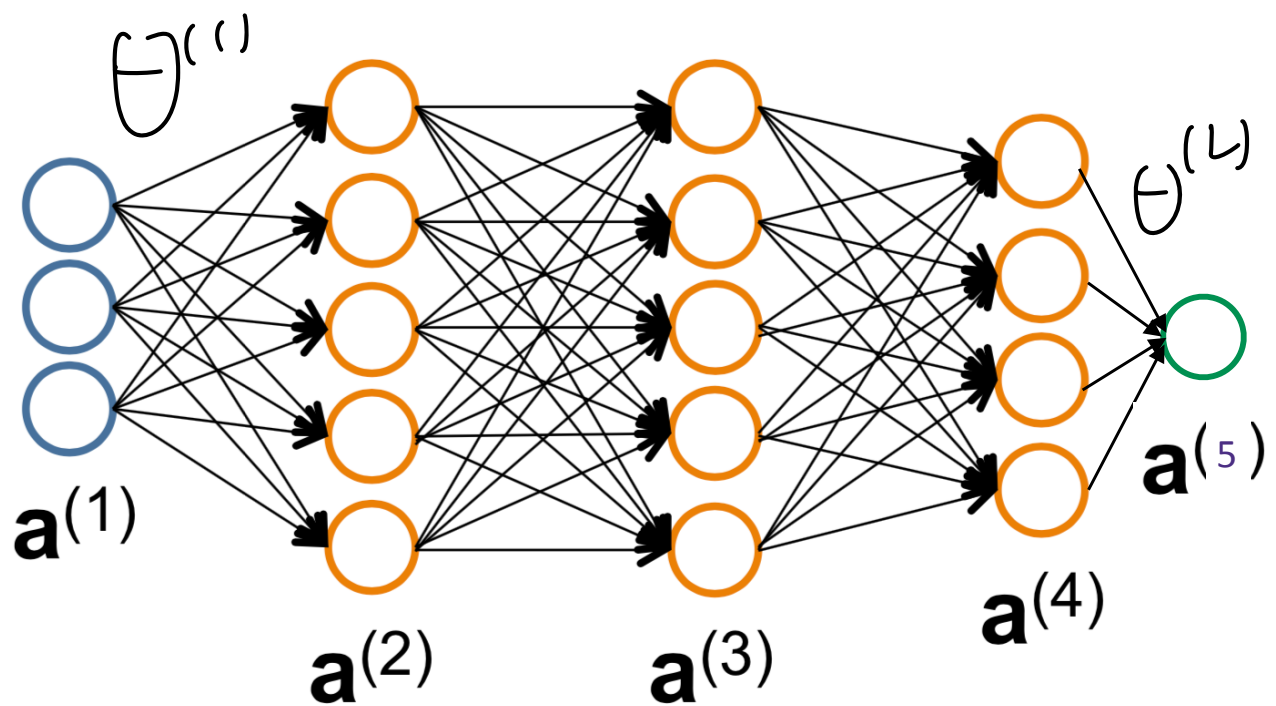
$$\vdots$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$

$$\hat{y} = g(\Theta^{(L)} a^{(L)})$$



$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$\text{Gradient Descent: } \Theta^{(l)} \leftarrow \Theta^{(l)} - \eta \nabla_{\Theta^{(l)}} L(y, \hat{y}) \quad \forall l$$

Forward Propagation

$z^{(l)}$, pre-activation
 $\Theta^{(l)}$

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$\vdots$

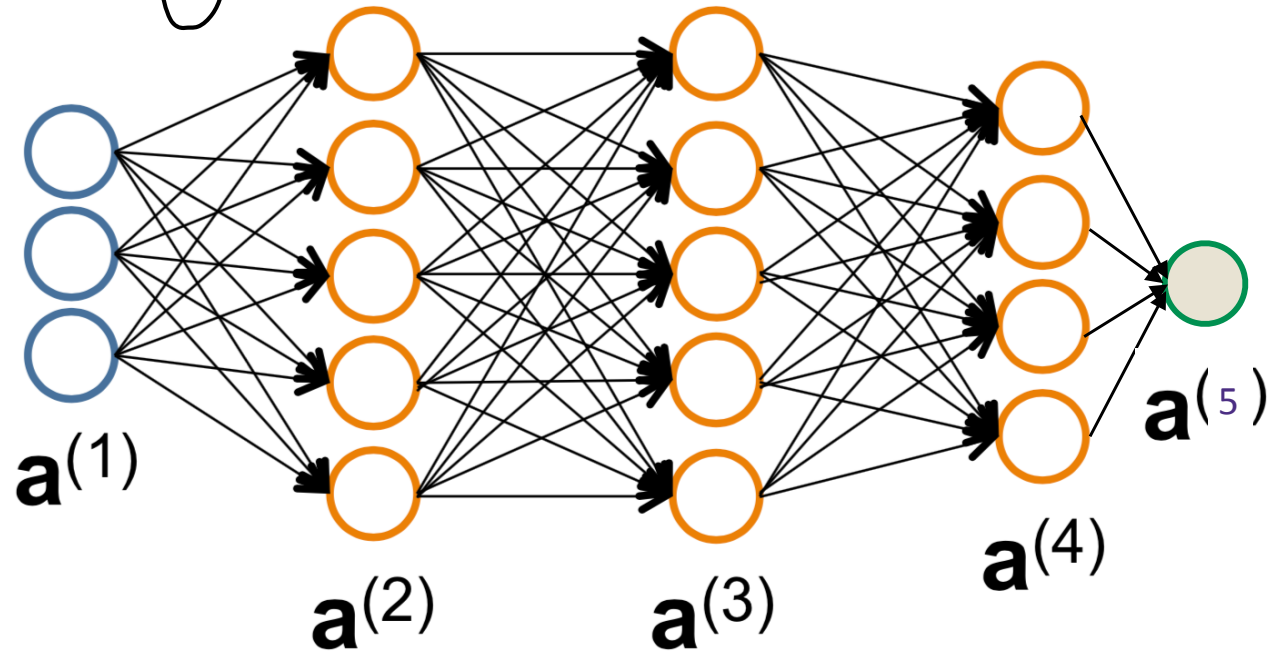
$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$\vdots$

$$\hat{y} = a^{(L+1)}$$



$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

Backprop

$x \in \mathbb{R}^d$, $\Theta^{(1)}, \dots, \Theta^{(L)}$: parameters to train
 $\Theta^{(1)}: \mathbb{R}^{m \times d}$, $\Theta^{(2)}, \dots, \Theta^{(L-1)} \in \mathbb{R}^{m \times m}$,
 $\Theta^{(L)}: \mathbb{R}^m$ vector

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$$\vdots$$
$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$
$$\hat{y} = a^{(L+1)}$$

Train by Stochastic Gradient Descent:

$$\Theta_{i,j}^{(l)} \leftarrow \Theta_{i,j}^{(l)} - \eta \frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}}$$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Backprop

Chain Rule

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$$\vdots$$
$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$
$$\hat{y} = a^{(L+1)}$$

$$\frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial \Theta_{i,j}^{(l)}} =: \delta_i^{(l+1)} \cdot a_j^{(l)}$$

Train by Stochastic Gradient Descent:

$$\Theta_{i,j}^{(l)} \leftarrow \Theta_{i,j}^{(l)} - \eta \frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}}$$

$$z_i^{(l+1)} = \sum_{j=1}^n \Theta_{i,j}^{(l)} \cdot a_j^{(l)}$$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Backprop

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$$\vdots$$

$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$

$$\hat{y} = a^{(L+1)}$$

$$\frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial \Theta_{i,j}^{(l)}} =: \delta_i^{(l+1)} \cdot a_j^{(l)}$$

$$\delta_i^{(l)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l)}} = \sum_k \frac{\partial L(y, \hat{y})}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}}$$

$$z_k^{(l+1)} = \sum_{u=1}^n \Theta_{ku}^{(l)} \cdot g(z_u^{(l)})$$

$\delta_k^{(l+1)}$ $\Theta_{ku}^{(l)} \cdot g'(z_u^{(l)})$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Backprop

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$$\vdots$$

$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$

$$\hat{y} = a^{(L+1)}$$

$$\frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial \Theta_{i,j}^{(l)}} =: \delta_i^{(l+1)} \cdot a_j^{(l)}$$

$$\begin{aligned} \delta_i^{(l)} &= \frac{\partial L(y, \hat{y})}{\partial z_i^{(l)}} = \sum_k \frac{\partial L(y, \hat{y})}{\partial z_k^{(l+1)}} \cdot \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} \\ &= \sum_k \delta_k^{(l+1)} \cdot \Theta_{k,i}^{(l)} g'(z_i^{(l)}) \\ &= a_i^{(l)}(1 - a_i^{(l)}) \sum_k \delta_k^{(l+1)} \cdot \Theta_{k,i}^{(l)} \end{aligned}$$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Backprop

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

⋮

$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

⋮

$$\hat{y} = a^{(L+1)}$$

$$\frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial \Theta_{i,j}^{(l)}} =: \delta_i^{(l+1)} \cdot a_j^{(l)}$$

$$\delta_i^{(l)} = a_i^{(l)}(1 - a_i^{(l)}) \sum_k \delta_k^{(l+1)} \cdot \Theta_{k,i}^{(l)}$$

Handwritten notes and arrows illustrating the backpropagation of error terms:

$\delta^{(L+1)} \rightarrow \delta^{(L)} \rightarrow \delta^{(L-1)} \dots \delta^{(1)}$

Arrows point from the error terms in the equation above to the corresponding terms in the handwritten sequence.

Below the sequence, the partial derivatives are written:

$\frac{\partial L(y, \hat{y})}{\partial \Theta^{(L)}}, \dots, \frac{\partial L(y, \hat{y})}{\partial \Theta^{(1)}}$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Backprop

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$$\vdots$$

$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$

$$\hat{y} = a^{(L+1)}$$

$$\frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial \Theta_{i,j}^{(l)}} =: \delta_i^{(l+1)} \cdot a_j^{(l)}$$

$$\delta_i^{(l)} = a_i^{(l)}(1 - a_i^{(l)}) \sum_k \delta_k^{(l+1)} \cdot \Theta_{k,i}^{(l)}$$

$$\begin{aligned} \delta_i^{(L+1)} &= \frac{\partial L(y, \hat{y})}{\partial z_i^{(L+1)}} = \frac{\partial}{\partial z_i^{(L+1)}} [y \log(g(z^{(L+1)})) + (1 - y) \log(1 - g(z^{(L+1)}))] \\ &= \frac{y}{g(z^{(L+1)})} g'(z^{(L+1)}) - \frac{1 - y}{1 - g(z^{(L+1)})} g'(z^{(L+1)}) \\ &= y - g(z^{(L+1)}) = y - a^{(L+1)} \end{aligned}$$

$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Backprop

$$a^{(1)} = x$$

$$z^{(2)} = \Theta^{(1)} a^{(1)}$$

$$a^{(2)} = g(z^{(2)})$$

$$\vdots$$
$$a^{(l)} = g(z^{(l)})$$

$$z^{(l+1)} = \Theta^{(l)} a^{(l)}$$

$$a^{(l+1)} = g(z^{(l+1)})$$

$$\vdots$$
$$\hat{y} = a^{(L+1)}$$

Chain Rule

$$\frac{\partial L(y, \hat{y})}{\partial \Theta_{i,j}^{(l)}} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial \Theta_{i,j}^{(l)}} =: \delta_i^{(l+1)} \cdot a_j^{(l)}$$

$$\delta_i^{(l)} = a_i^{(l)}(1 - a_i^{(l)}) \sum_k \delta_k^{(l+1)} \cdot \Theta_{k,i}^{(l)}$$

$$\delta^{(L+1)} = y - a^{(L+1)}$$

Recursive Algorithm!

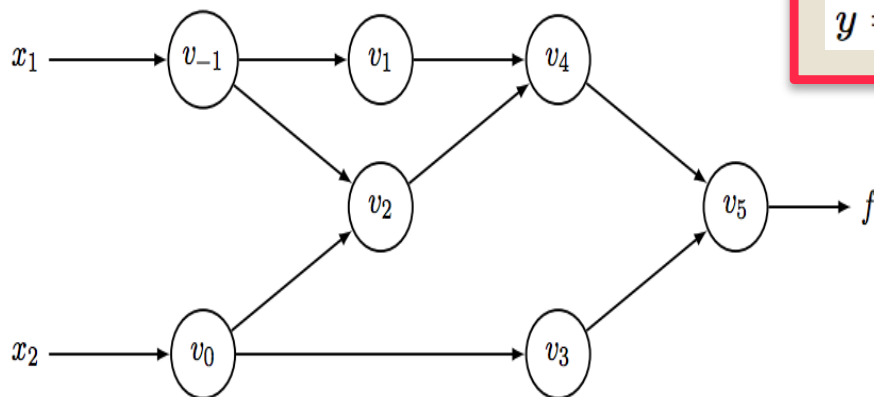
$$L(y, \hat{y}) = y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})$$

$$g(z) = \frac{1}{1 + e^{-z}} \quad \delta_i^{(l+1)} = \frac{\partial L(y, \hat{y})}{\partial z_i^{(l+1)}}$$

Auto-differentiation

Goal: compute gradient
takes $\odot$ (compute function)

Backprop for this simple network architecture is a special case of *reverse-mode auto-differentiation*:



$$y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$$

Forward Primal Trace

$v_{-1} = x_1$	$= 2$
$v_0 = x_2$	$= 5$
$v_1 = \ln v_{-1}$	$= \ln 2$
$v_2 = v_{-1} \times v_0$	$= 2 \times 5$
$v_3 = \sin v_0$	$= \sin 5$
$v_4 = v_1 + v_2$	$= 0.693 + 10$
$v_5 = v_4 - v_3$	$= 10.693 + 0.959$
$y = v_5$	$= 11.652$

Reverse Adjoint (Derivative) Trace

$\bar{x}_1 = \bar{v}_{-1}$	$= 5.5$
$\bar{x}_2 = \bar{v}_0$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} = \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0 = \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0} = \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} = \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0} = \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} = \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5 = \bar{y}$	$= 1$

Auto-differentiation

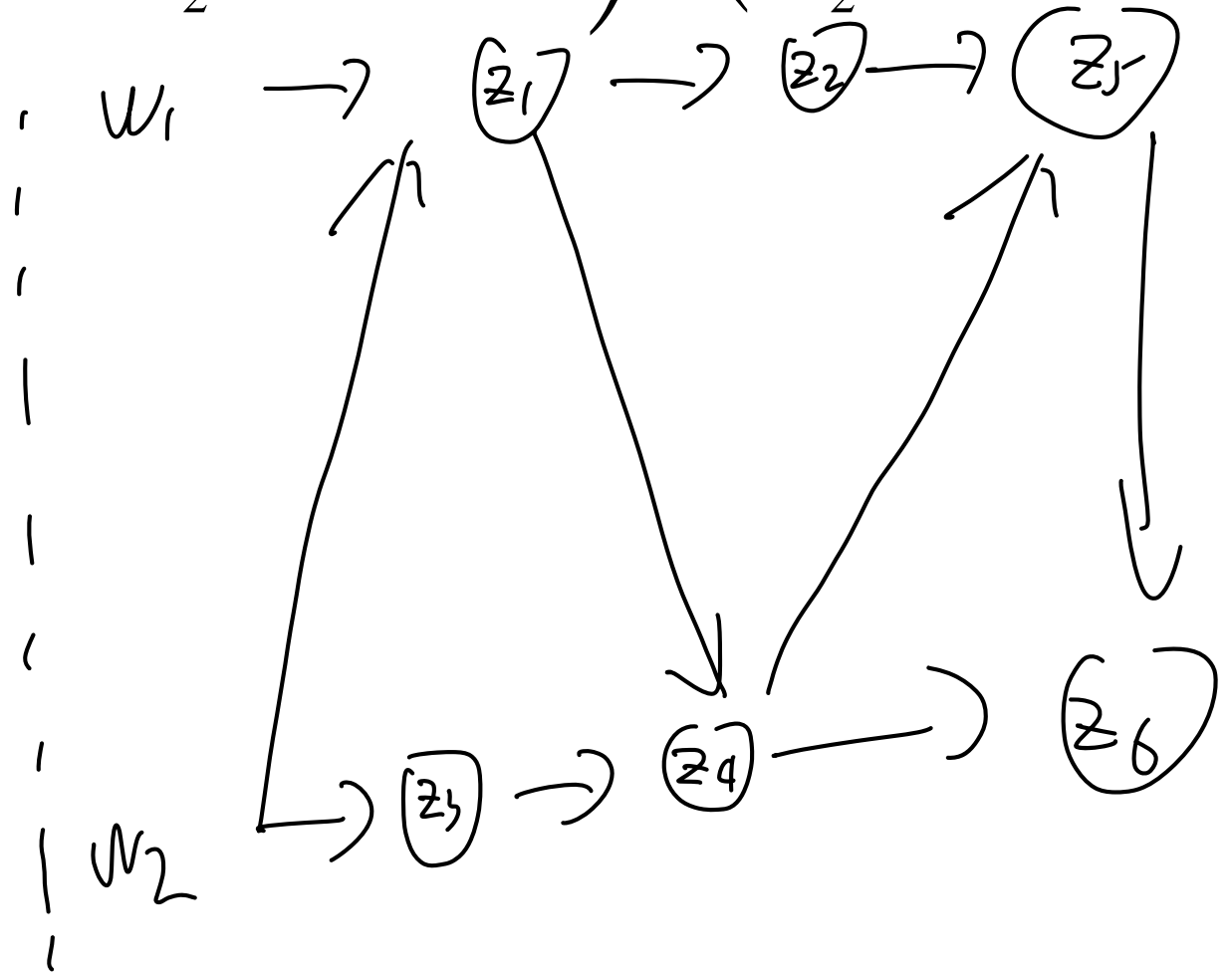
- Given a function, computes its partial derivatives
- Compute all of the partial derivatives of a function with (nearly) same computation runtime [Griewank '89, Baur and Strassen '83]
- Backbone of (applied) machine learning: Pytorch, Tensorflow, ...

Example of Computation Graph

$$f(w_1, w_2) = \left(\sin\left(\frac{2\pi w_1}{w_2}\right) + \frac{3w_1}{w_2} - \exp(2w_2) \right) \cdot \left(\frac{3w_1}{w_2} - \exp(2w_2) \right)$$

Input: $z_0 = (w_1, w_2)$

1. $z_1 = \frac{w_1}{w_2}$
2. $z_2 = \sin(2\pi z_1)$
3. $z_3 = \exp(2w_2)$
4. $z_4 = 3z_1 - z_3$
5. $z_5 = z_2 + z_4$
6. $z_6 = z_5 - z_5$



Computation Model

- Given access to a set of differentiable real functions $h \in \mathcal{H}$
- Use functions in $\mathcal{H}$ to create intermediate variables.
- Evaluation trace:
 - All intermediate variables will be scalars; each corresponds to a node.
 - Input $z_0 = w \in \mathbb{R}^d$. $[z_0]_1 = w_1, [z_0]_2 = w_2, \dots, [z_0]_d = w_d$
 - Step 1: $z_1 = h_1$ (a subset of variables in w)
 -
 - Step t: $z_t = h_t$ (a subset of variables in $z_1, \dots, z_{t-1}, w$)
 - ...
 - Step T: $z_T = h_T$ (a subset of variables in $z_1, \dots, z_{T-1}, w$)
 - **Return:** z_T
($h_1, \dots, h_T \in \mathcal{H}$)

Computation Model

- Every $h \in \mathcal{H}$ is one of the following:
 - Type 1: An affine transformation of the inputs

$$z_4 = z_1 - z_3$$

- Type 2: A product of variables, to some power

$$z_6 = z_4 \cdot z_5$$

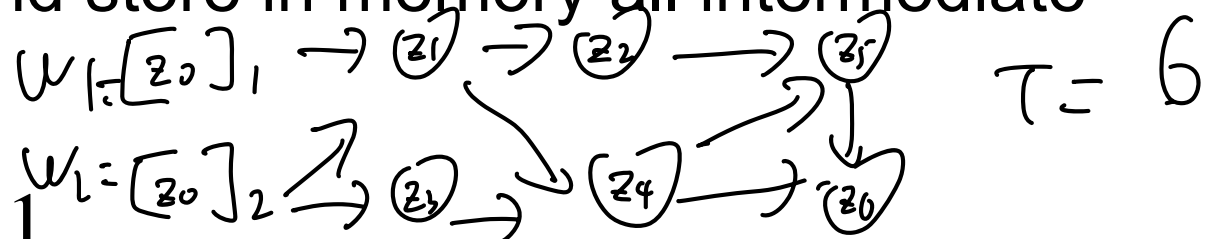
- Type 3: A fixed set of one dimensional differentiable functions: $\sin(\cdot)$, $\cos(\cdot)$, $\exp(\cdot)$, $\log(\cdot)$, ... ReLU ✓
 - We assume we can easily compute the derivatives for each of this functions.
 - Type 3 can be approximated by Type 1 and Type 2, using polynomials.

Reverse Mode of Automatic Differentiation

$f(w) = z_6$, goal: $\frac{dz_6}{dw}$

Goal: Compute partial derivatives of $f(w)$, i.e., df/dw .

- Step 1: compute $f(w)$ and store in memory all intermediate variables $z_1, \dots, z_T$



- Step 2: Initialize: $\frac{dz_T}{dz_T} = 1$.

(1) $\frac{dz_6}{dz_6} = 1$

- Step 3: For $t = T, T-1, \dots, 0$

(2) $\frac{dz_6}{dz_5} = \frac{dz_6}{dz_6} \cdot \frac{\partial z_6}{\partial z_5} = z_4$

(3) $\frac{dz_6}{dz_4} = \frac{dz_6}{dz_6} \cdot \frac{\partial z_6}{\partial z_4} + \frac{dz_6}{dz_5} \cdot \frac{\partial z_5}{\partial z_4}$

$$\frac{dz_T}{dz_t} = \sum_{c \text{ is a child of } t} \frac{dz_T}{dz_c} \cdot \frac{\partial z_c}{\partial z_t}$$

(Child: a node z_t directly points to)

(4) $\frac{dz_6}{dz_3} = \frac{dz_6}{dz_4} \cdot \frac{\partial z_4}{\partial z_3} = z_5 + 1$

(5) $\frac{dz_6}{dz_2} = \frac{dz_6}{dz_5} \cdot \frac{\partial z_5}{\partial z_2}$

(6) $\frac{dz_6}{dz_1} = \frac{dz_6}{dz_2} \cdot \frac{\partial z_2}{\partial z_1} + \frac{dz_6}{dz_4} \cdot \frac{\partial z_4}{\partial z_1}$

- Step 4: Return $\frac{dz_T}{dz_0} = \frac{df}{dw}$

last: $\frac{dz_6}{dz_0} = \frac{dz_6}{dz_1} \cdot \frac{\partial z_1}{\partial z_0} + \frac{dz_6}{dz_2} \cdot \frac{\partial z_2}{\partial z_0}$

Time Complexity

Theorem (Baur and Strassen '83, Griewak '89): Assume every h is specified as in our computational model. For $h(\cdot)$ of type 3, assume we can compute the derivative $h'(z)$ in time as the same order of computing $h(z)$. Let T denote the time to compute $f(w)$. Then the reverse mode computes df/dw in time $O(T)$.

graph of $\vee$ node $Q \in \text{edge}$

$$O(V + E)$$

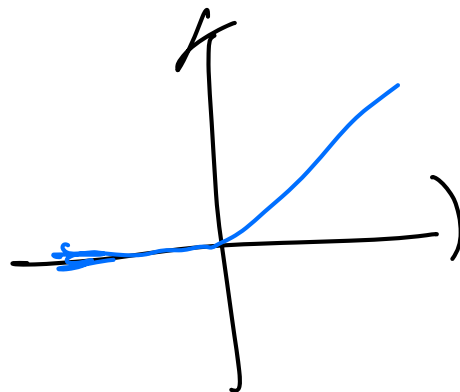
$$f(w)$$

$$\nabla f(w)$$

Clarke Differential

$$\text{ReLU}(x) = \max\{x, 0\}$$

Not differentiable



w

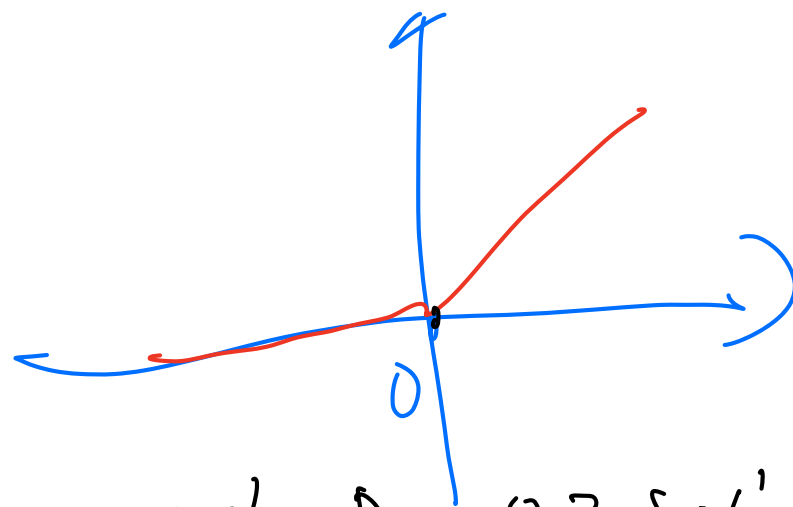
Subdifferential and Subgradient

Definition: Given $f: \mathbb{R}^d \rightarrow \mathbb{R}$, for every x , the subdifferential set is defined as

$\partial_s f(x) \triangleq \{s \in \mathbb{R}^d : \forall x' \in \mathbb{R}^d, f(x') \geq f(x) + s^\top (x' - x)\}$. The elements in the subdifferential set are subgradients.

$$GD: X_{t+1} \leftarrow X_t - \eta \nabla f(X_t)$$

subgradient descent:
pick $g_t \in \partial_s f(x_t)$
 $X_{t+1} \leftarrow X_t - \eta g_t$



$$\partial_s f(0) = \{s : \forall x', \text{ReLU}(x') \geq s \cdot x'\}$$

$$\begin{aligned} & \text{if } x' > 0, \quad \max\{0, x'\} \geq s \cdot x' \\ & \text{if } x' \leq 0, \quad s \leq 1 \Rightarrow [0, 1] \\ & \quad \quad \quad s' \geq 0 \end{aligned}$$

Subdifferential and Subgradient

Definition: Given $f : \mathbb{R}^d \rightarrow \mathbb{R}$, for every x , the subdifferential set is defined as

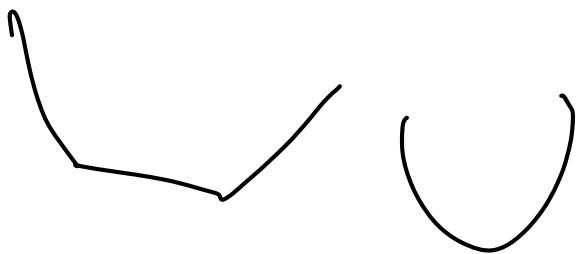
$\partial_s f(x) \triangleq \{s \in \mathbb{R}^d : \forall x' \in \mathbb{R}^d, f(x') \geq f(x) + s^\top (x' - x)\}$. The elements in the subdifferential set are subgradients.

if f is differentiable at x

$$\partial_s f(x) = \{\nabla f(x)\}$$

if f is convex

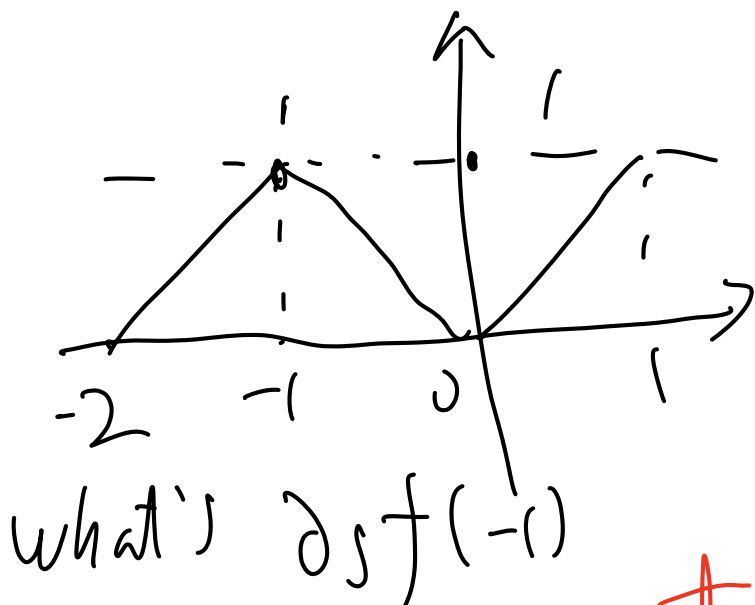
$\partial_s f(x)$ is always non-empty



Subdifferential is not enough

Definition: Given $f: \mathbb{R}^d \rightarrow \mathbb{R}$, for every x , the subdifferential set is defined as

$\partial_s f(x) \triangleq \{s \in \mathbb{R}^d : \forall x' \in \mathbb{R}^d, f(x') \geq f(x) + s^\top (x' - x)\}$. The elements in the subdifferential set are subgradients.



empty set

We need s s.t. $\forall x'$
 $f(x') \geq 1 + s^\top (x' - (-1))$

choose $x' = -2$

$$0 \geq 1 + s((-2) - (-1))$$

$$\Rightarrow s \geq 1$$

choose $x' = 1$

$$1 \geq 1 + s(1 - (-1))$$

$$\Rightarrow s \leq 0$$

Clarke Differential

$$x_{t+1} \leftarrow x_t - \eta g_t$$

where pick $g_t \in \partial f(x_t)$

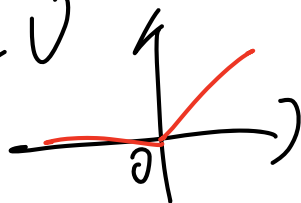
Definition: Given $f: \mathbb{R}^d \rightarrow \mathbb{R}$, for every x , the Clark differential is defined as

$$\partial f(x) \triangleq \text{conv}(\{s \in \mathbb{R}^d : \exists \{x_i\}_{i=1}^{\infty} \rightarrow x, \{\nabla f(x_i)\}_{i=1}^{\infty} \rightarrow s\}).$$

The elements in the subdifferential set are subgradients.

$$\text{conv}(S) = \left\{ V : V = \sum_{i=1}^n \lambda_i u_i, u_i \in S, \lambda_i \geq 0, \sum_{i=1}^n \lambda_i = 1 \right\}$$

1. example ReLU



$$\{x_i\}: -1, -\frac{1}{2}, -\frac{1}{3}, \dots \rightarrow 0, \nabla f(x_i) = 0$$

$$\{x_i\}: 1, \frac{1}{2}, \dots \rightarrow 0, \nabla f(x_i) = 1$$

$$S = \{1, 0\} \Rightarrow \text{conv}(S) = [0, 1]$$



2.

$$\{x_i\}: -2, -\frac{3}{2}, \dots \rightarrow -1, \nabla f(x_i) = 1$$

$$\{x_i\}: 0, -\frac{1}{2}, \dots \rightarrow -1, \nabla f(x_i) = -1$$

$$\begin{aligned} & \text{conv}(\{-1, 1\}) \\ &= \{ -\lambda_1 + \lambda_2 \mid \lambda_1, \lambda_2 \geq 0, \lambda_1 + \lambda_2 = 1 \} \\ &= [-1, 1] \end{aligned}$$

When does Clarke differential exists

Definition (Locally Lipschitz): $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is locally Lipschitz if $\forall x \in \mathbb{R}^d$, there exists a neighborhood S of x , such that f is Lipschitz in S .

$$\forall x' \in S, \|x' - x\| \text{ small, } L > 0 \\ |f(x') - f(x)| \leq L \cdot \|x' - x\|$$

ReLU, Leaky-ReLU are locally Lip

Thm: If f is locally Lip $\Rightarrow$ Clarke differential exist

$\Rightarrow$ If f is convex, $\partial_s f = \partial f$

$\Rightarrow$ if f is differentiable $\Rightarrow \partial f = \{\nabla f\}$