

Memory vs compute

```
# generate some nonsense data for an example
X = np.random.randn(n,d)
y = np.random.randn(n)
```

```
# generate the random features
G = np.random.randn(p, d)*np.sqrt(.1)
b = np.random.rand(p)*2*np.pi
```

```
# construct HTH
HTH = np.zeros((p,p))
HTy = np.zeros(p)
for i in range(n):
    hi = np.dot(X[i,:], G.T)+b
    HTH += np.outer(hi, hi)
    HTy += y[i]*hi
    if i % 1000==0: print(i)
```

```
# construct HTH
HTH = np.zeros((p,p))
HTy = np.zeros(p)
block = p
for i in range(int(np.ceil(n/block))+1):
    Hi = np.dot(X[i*block:min(n,(i+1)*block),:], G.T)+b
    HTH += np.dot(Hi.T, Hi)
    HTy += np.dot(Hi.T, y[i*block:min(n,(i+1)*block)])
```

```
w = np.linalg.solve(HTH + lam*np.eye(p), HTy)
```

1 float in NumPy = 8 bytes
 $10^6 \approx 2^{20}$ bytes = 1 MB
 $10^9 \approx 2^{30}$ bytes = 1 GB

For each block compute the memory required in terms of n, p, d.

If $d \ll p \ll n$, what is the most memory efficient program (blue, green, red)?

If you have unlimited memory, what do you think is the fastest program?

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., Lasso, Logistic Regression do not have closed form solution for

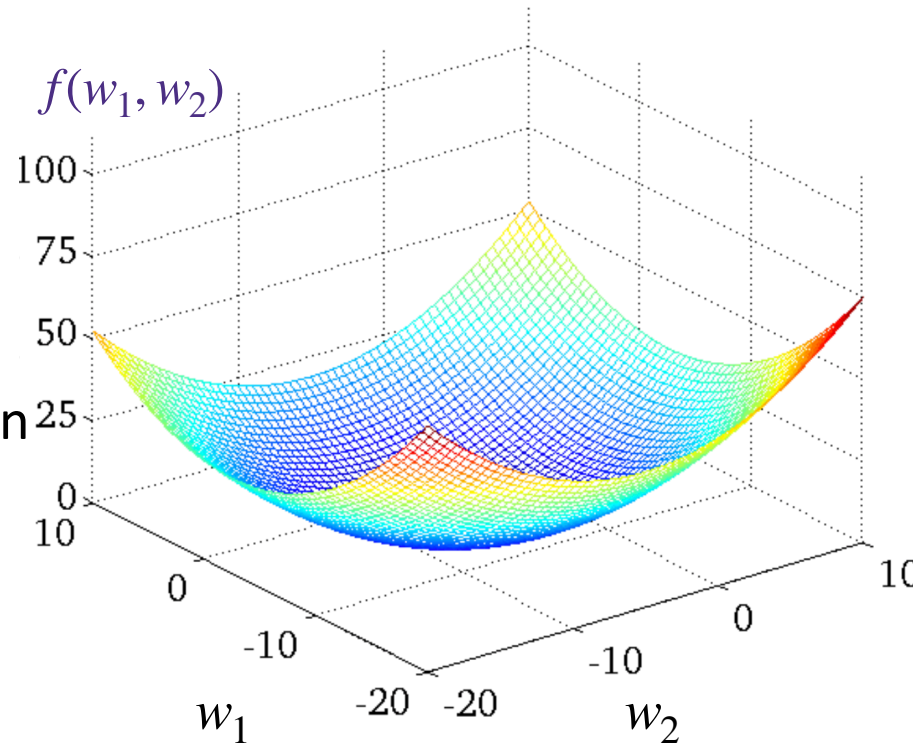
$$\nabla_{b,w} \mathcal{L}(b, w) = 0$$

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:**

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

- 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

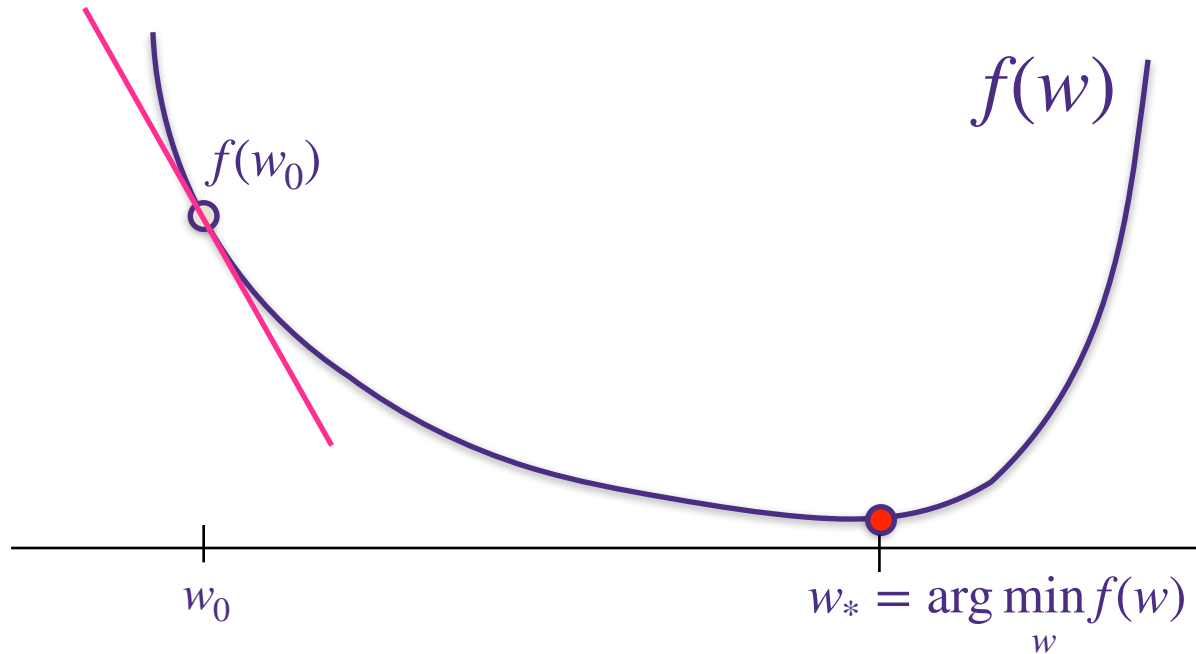
Let w_0 be an initial guess. How can we improve this solution?

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 $f(w)$



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

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

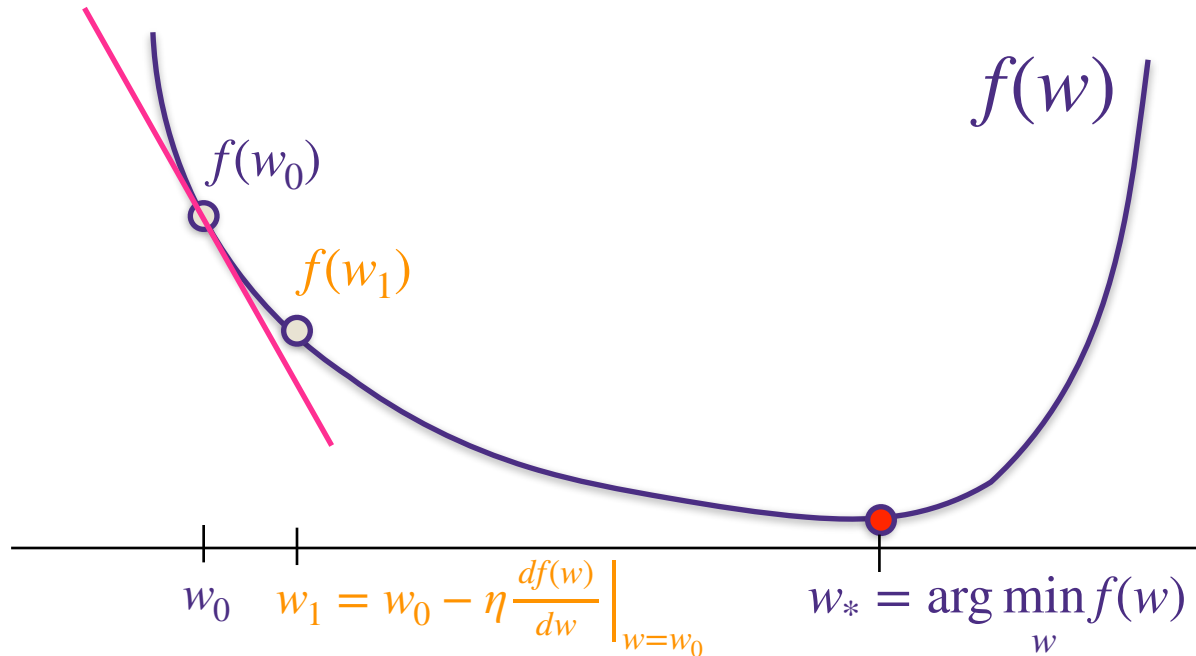
is very close to $f(w)$

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 $f(w_1) < f(w_0)$



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

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

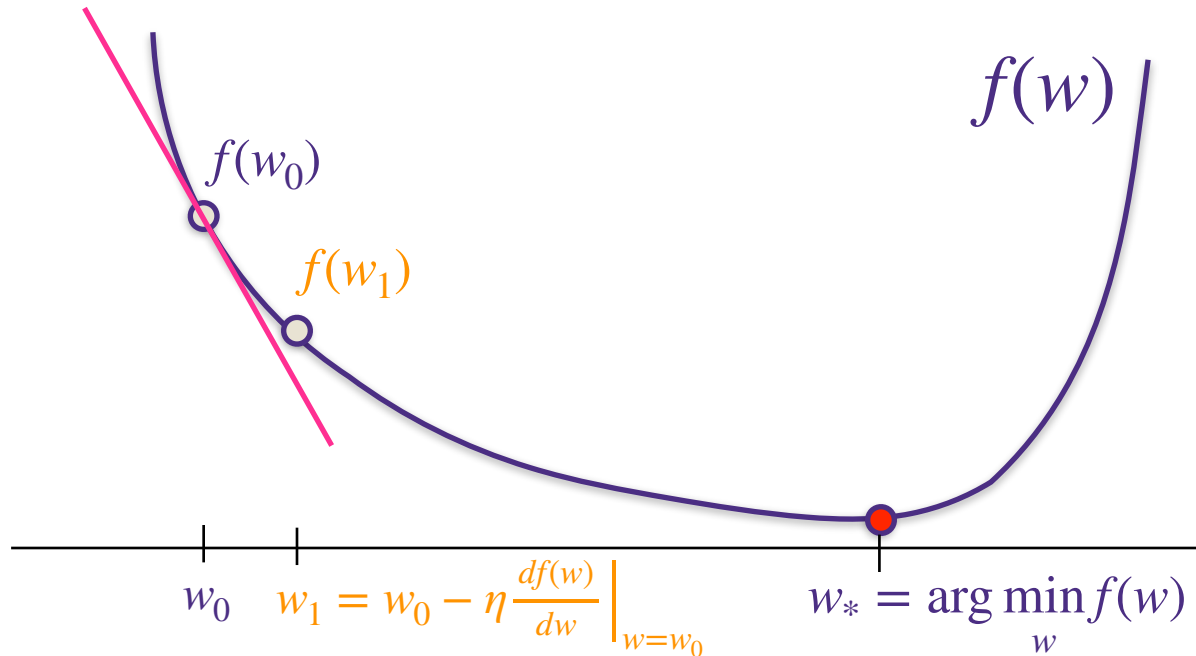
is very close to $f(w)$

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

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

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

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



Gradient descent

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

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

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

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

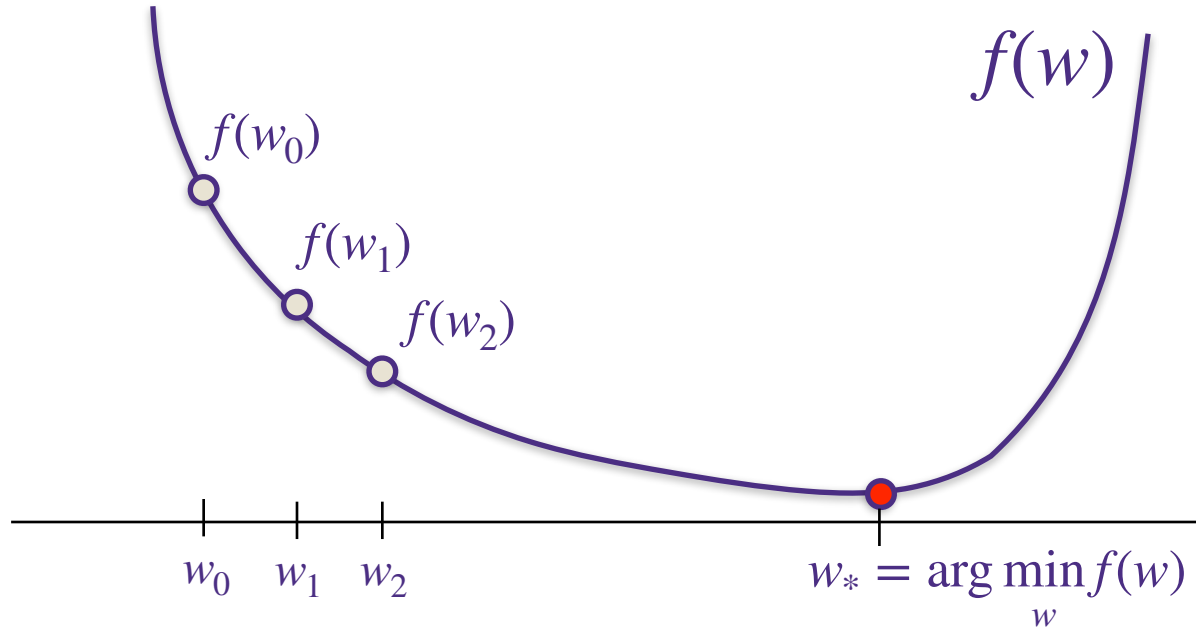
is very close to $f(w)$

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

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

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

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



Gradient descent

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

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

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

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

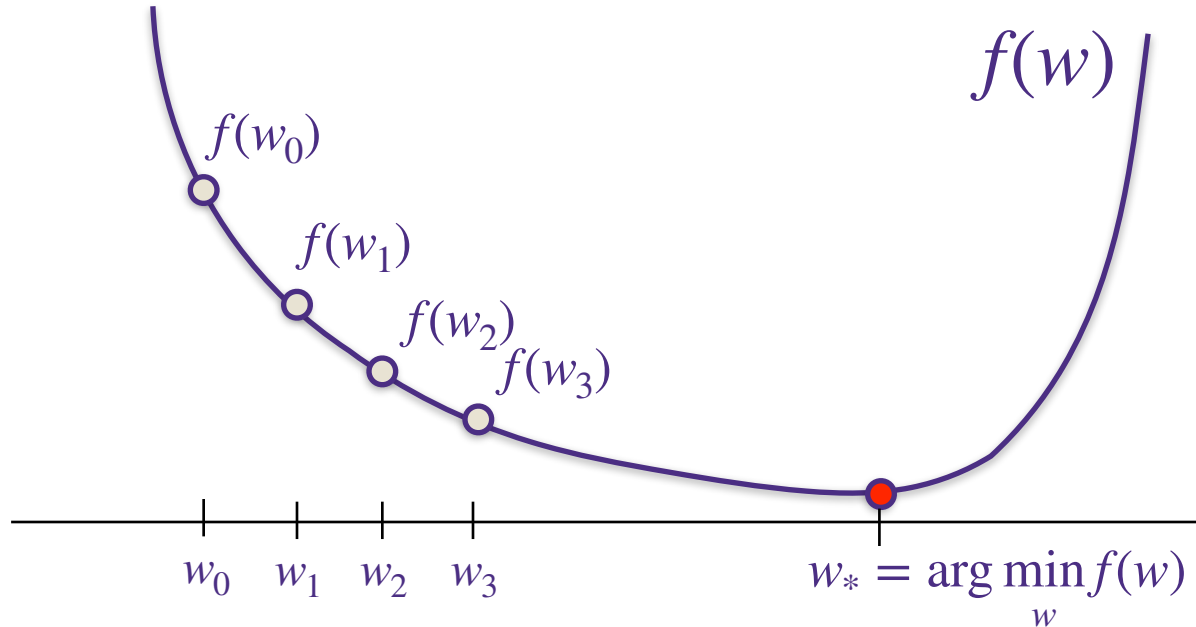
is very close to $f(w)$

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

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

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

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



Gradient descent

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

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

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

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

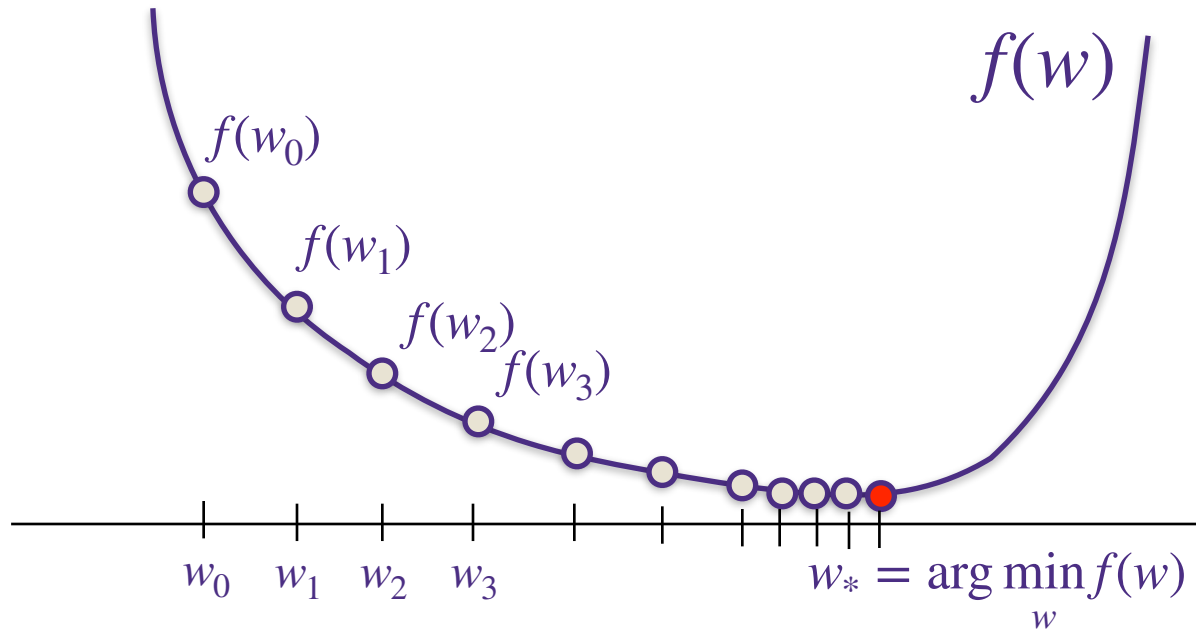
is very close to $f(w)$

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 $\frac{df(w)}{dw} \Big|_{w=w_k} \rightarrow 0$

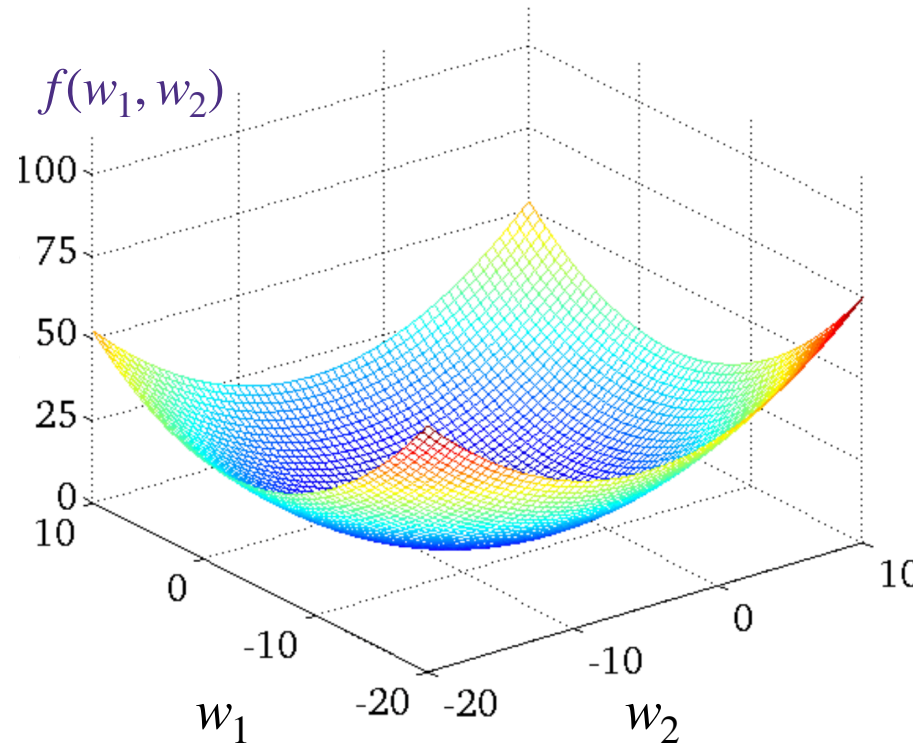
Running example: linear regression

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

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

- **Gradient descent:**

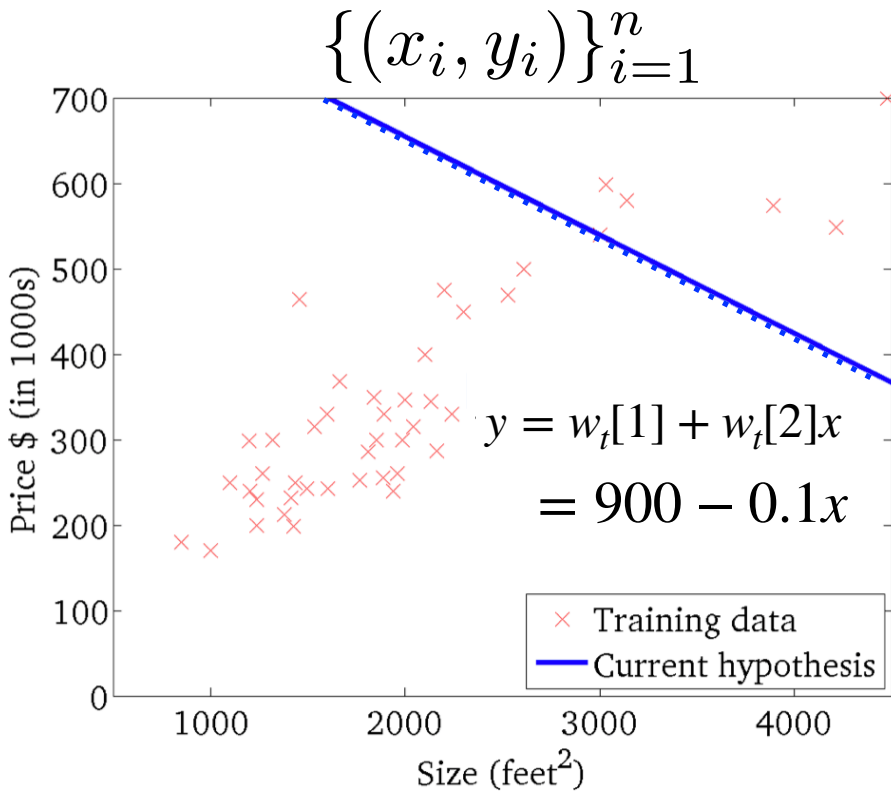
- Initialize: $w_0 = 0$
- For $t=0,1,2,\dots$
 - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



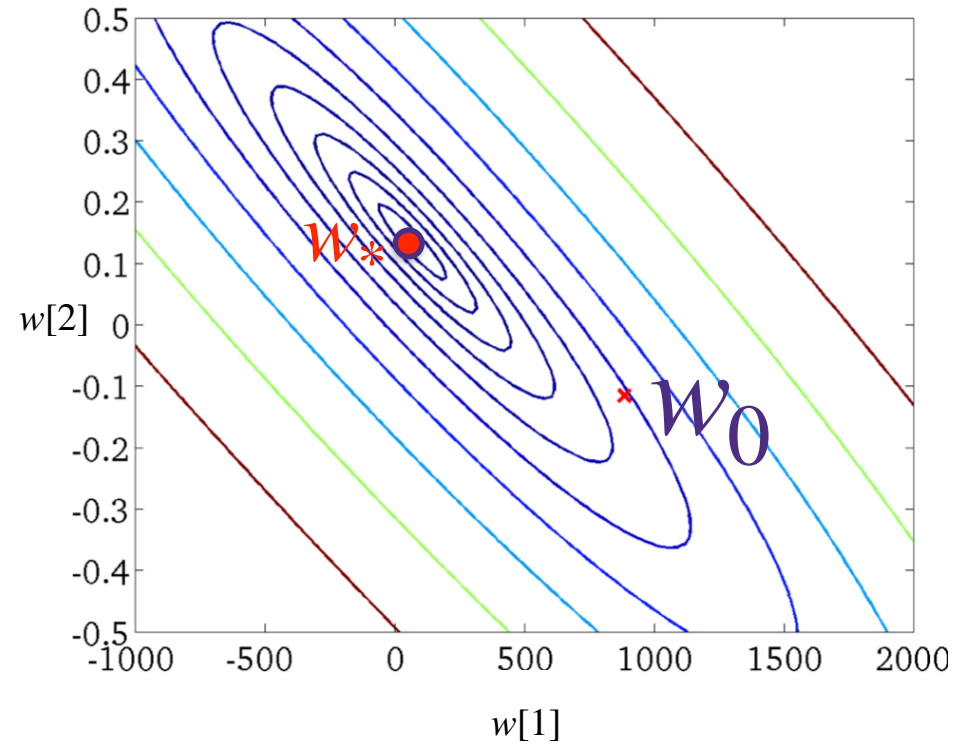
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor



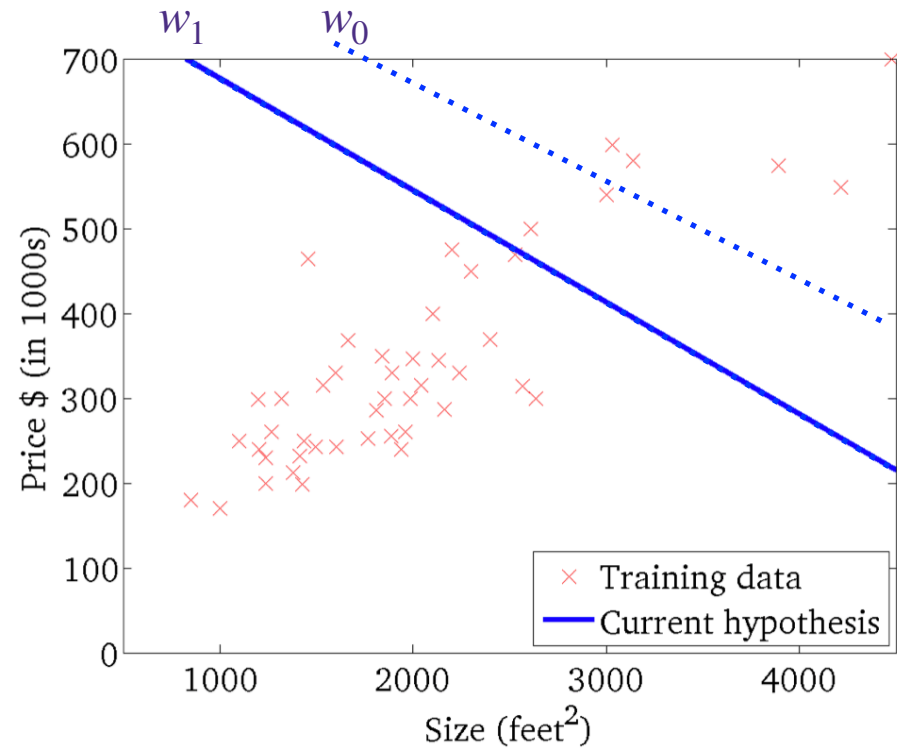
GD dynamics in the Parameter space

- Which direction will the GD move?

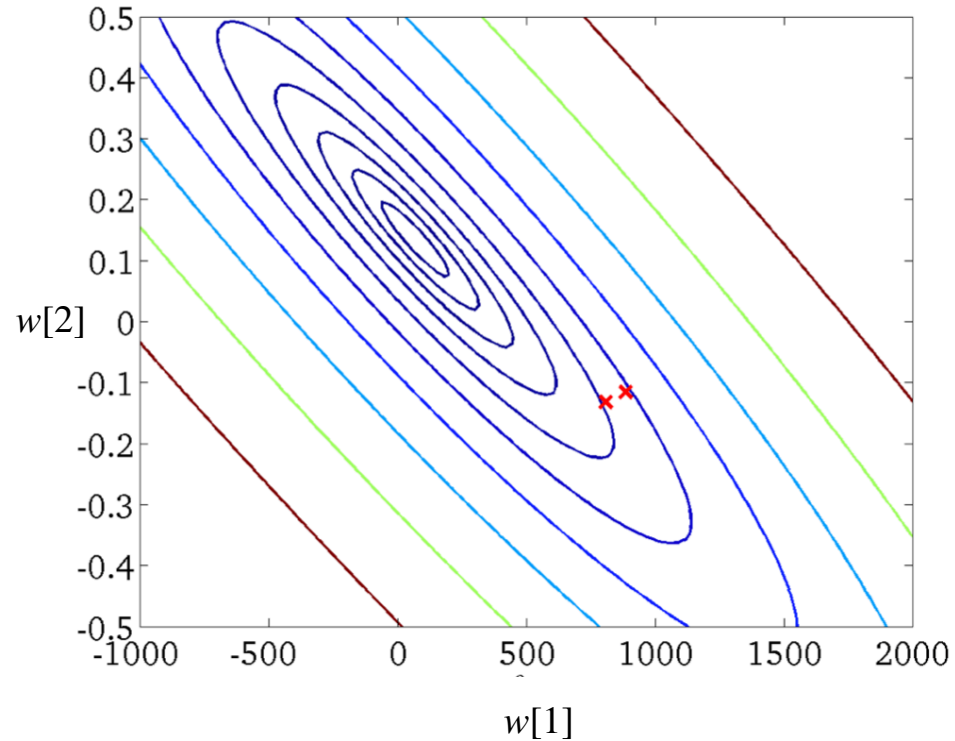
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

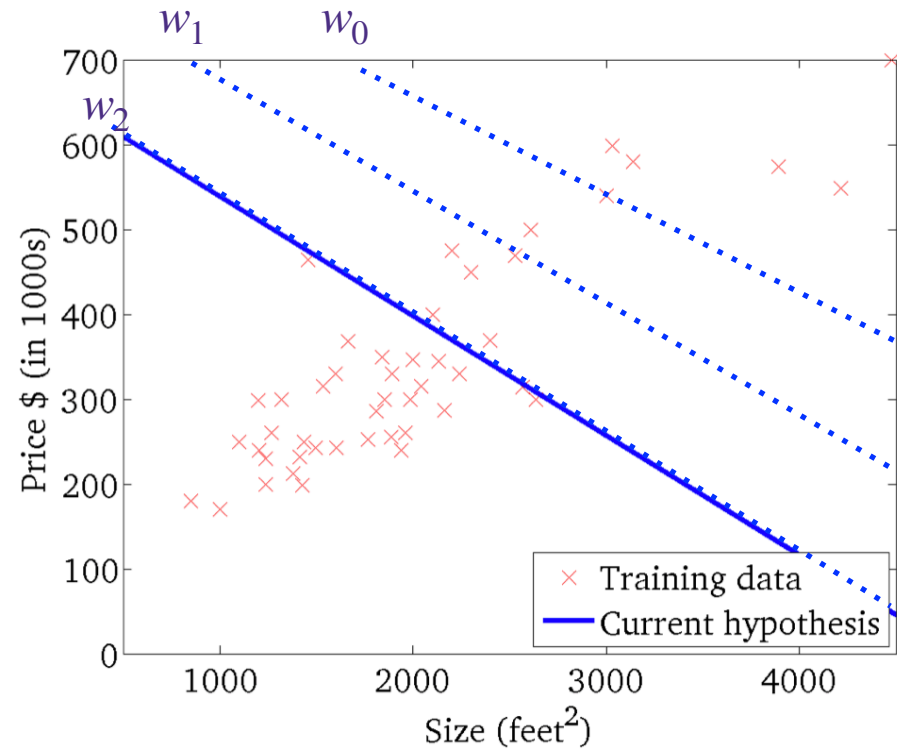


GD dynamics in the Parameter space

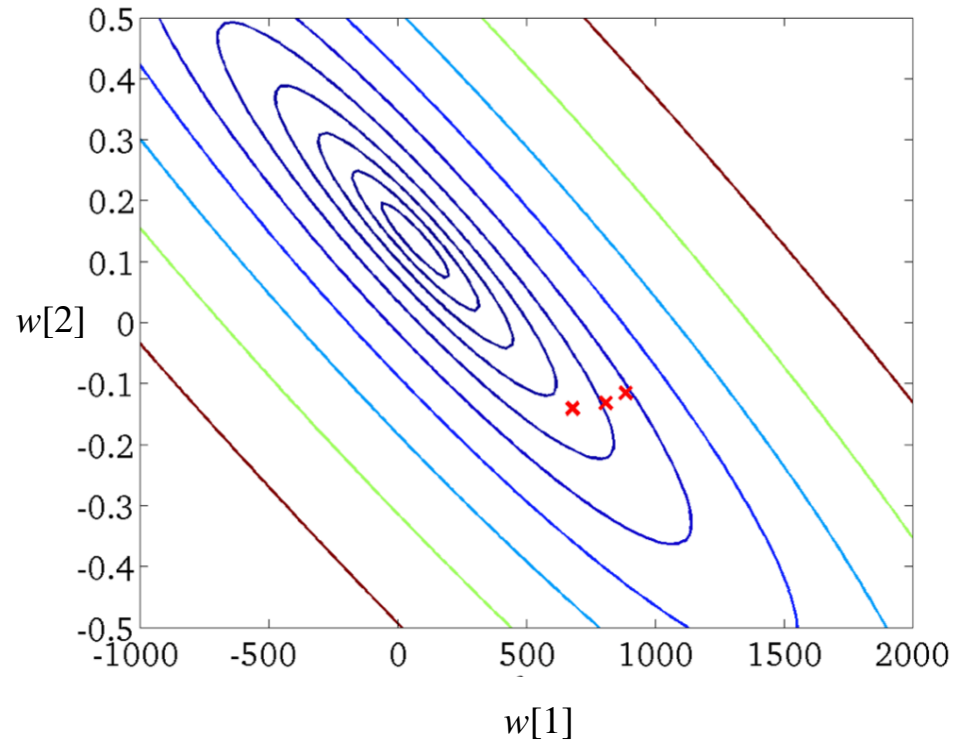
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

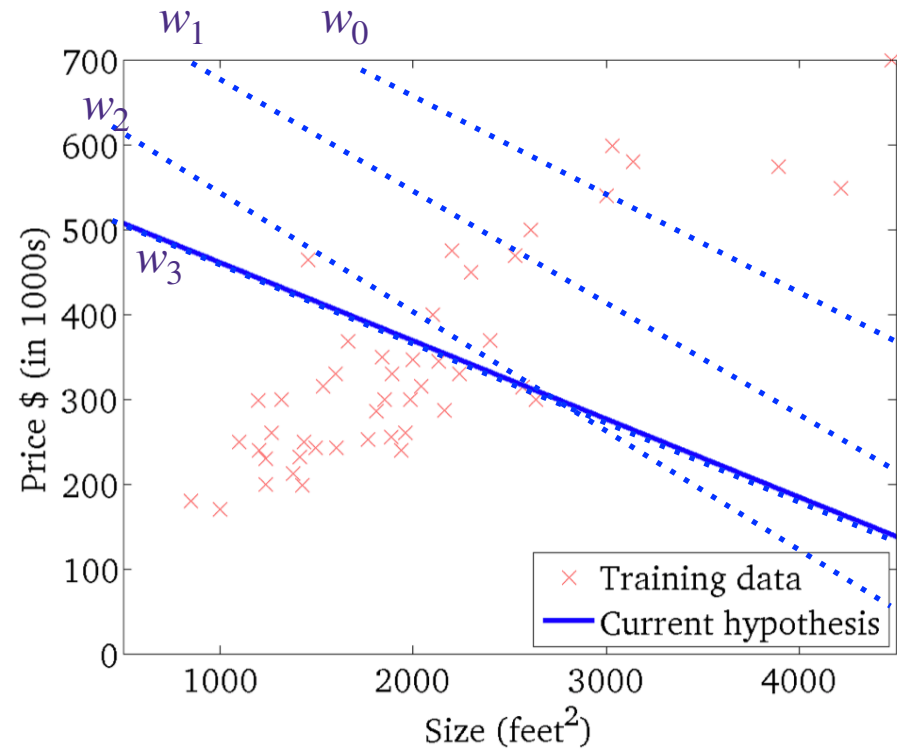


GD dynamics in the Parameter space

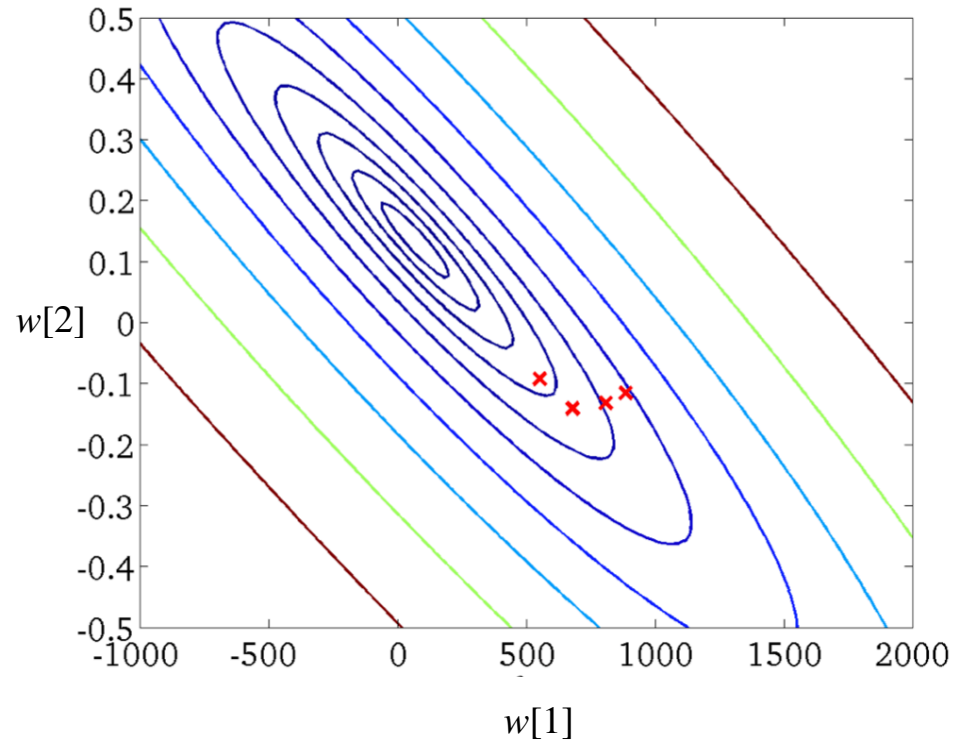
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

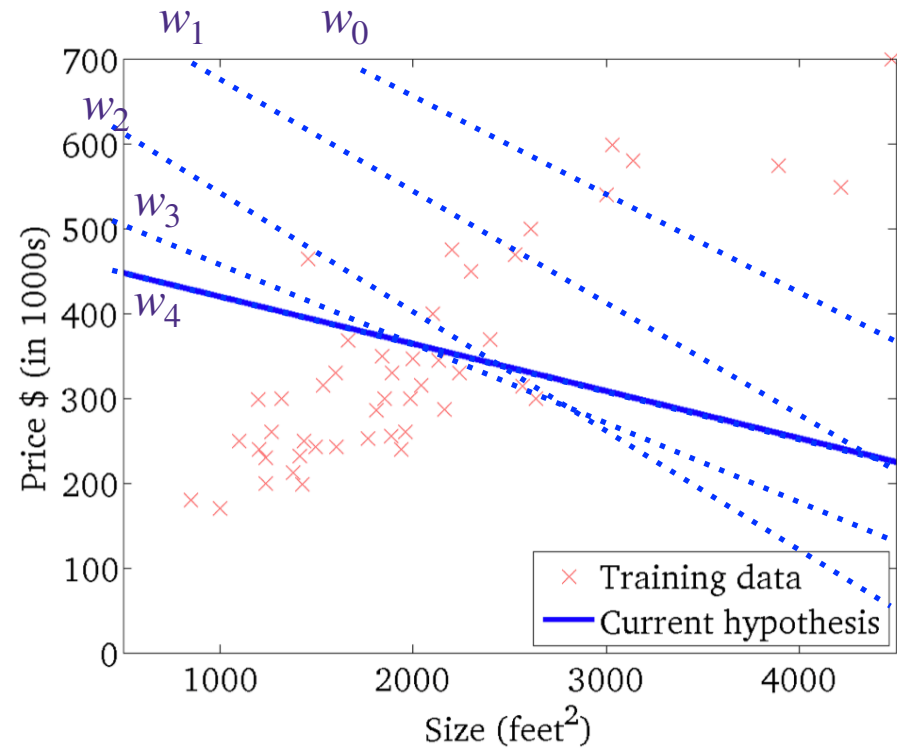


GD dynamics in the Parameter space

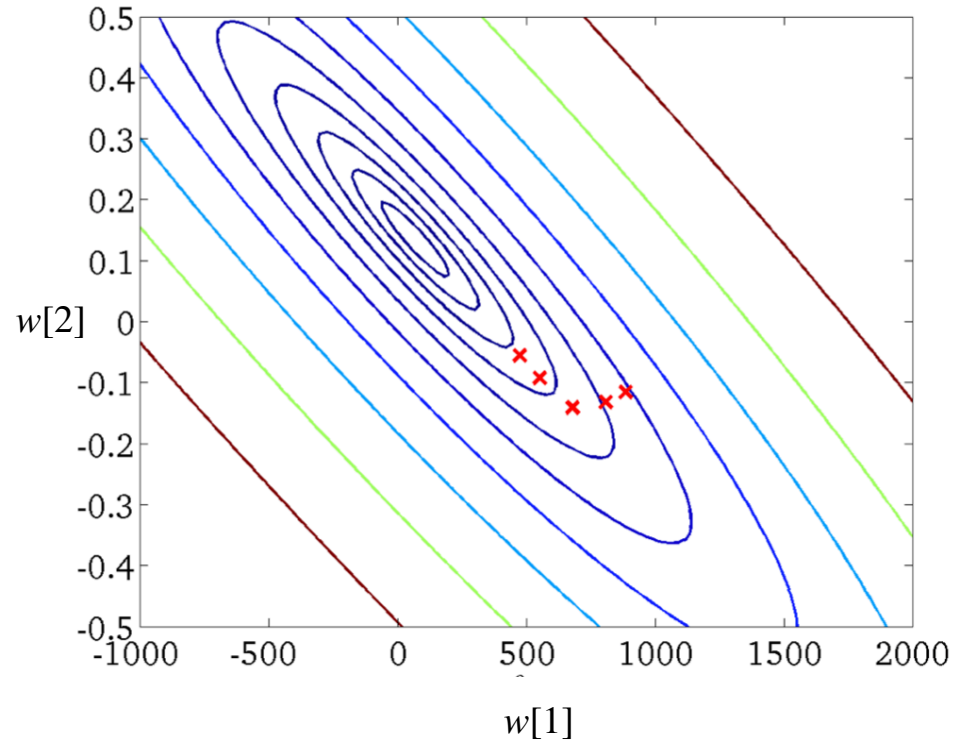
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

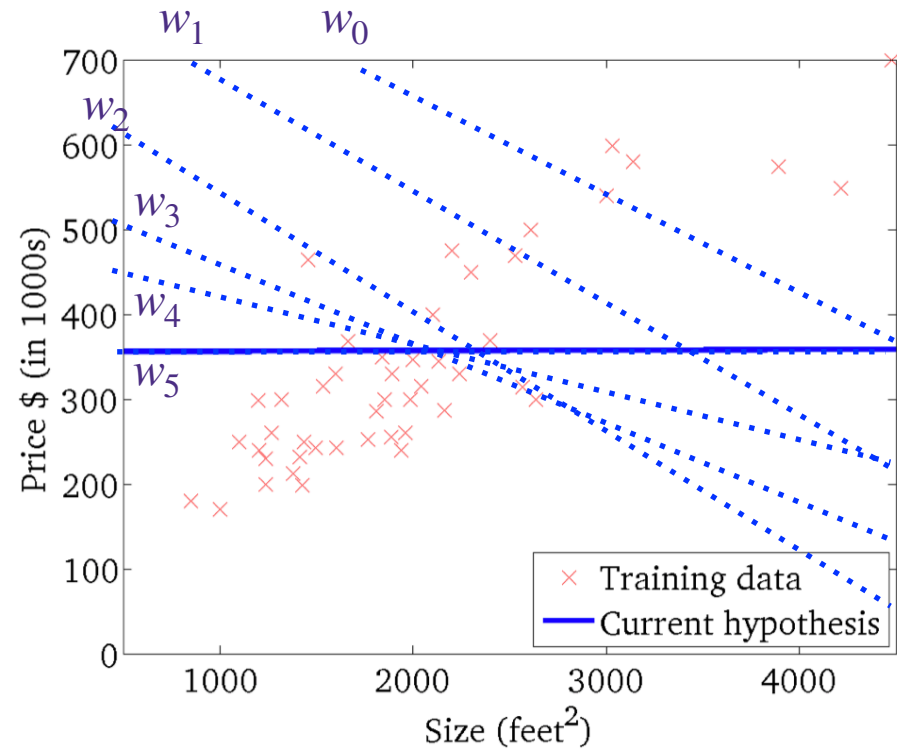


GD dynamics in the Parameter space

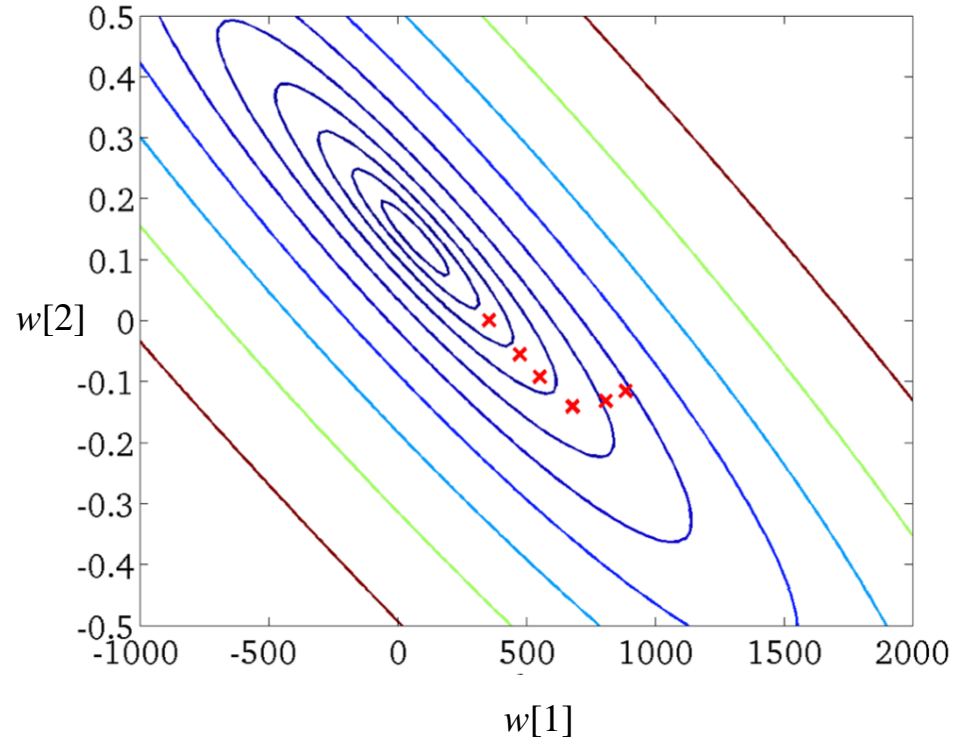
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

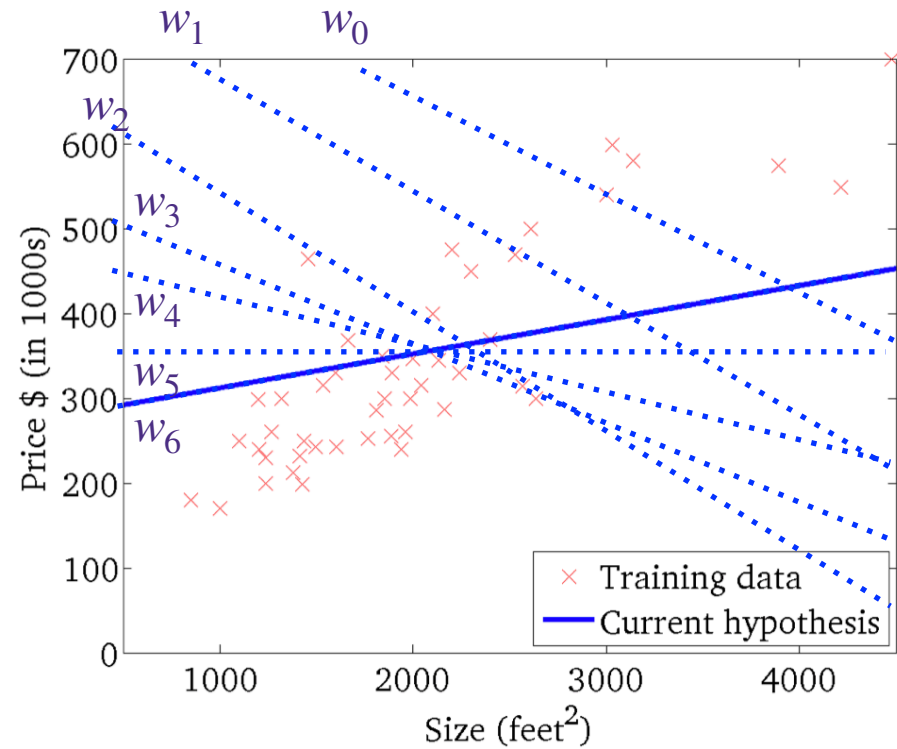


GD dynamics in the Parameter space

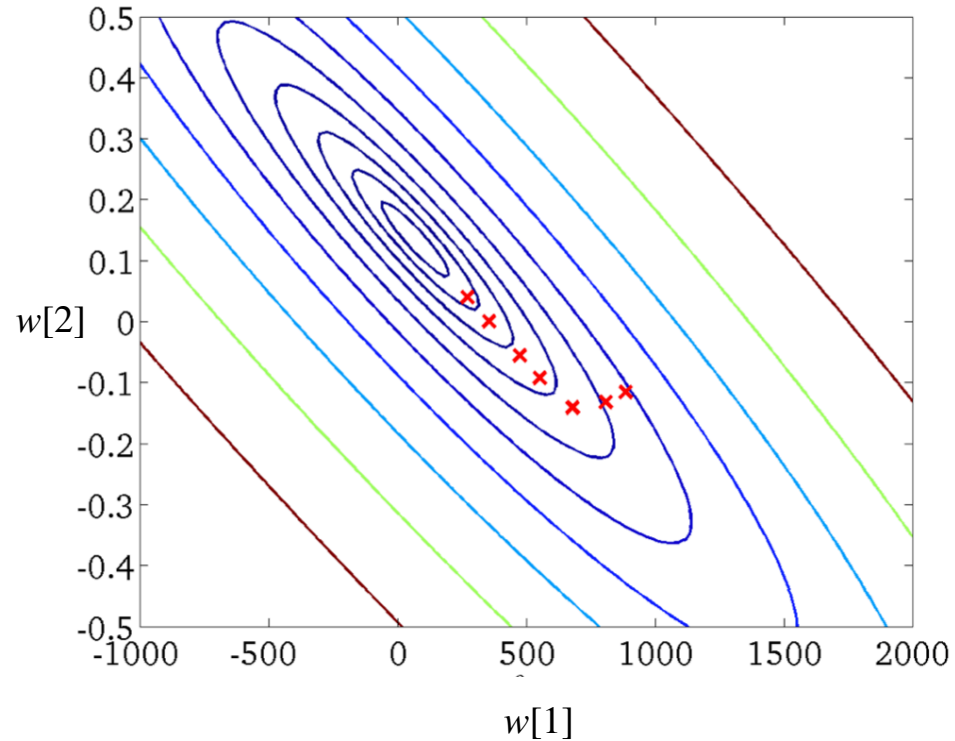
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

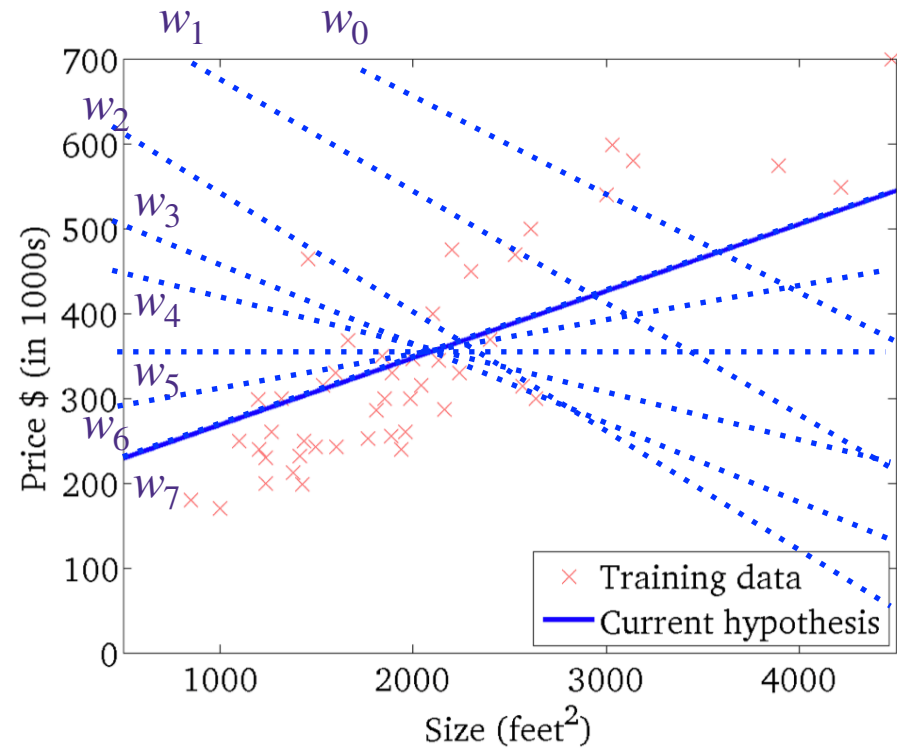


GD dynamics in the Parameter space

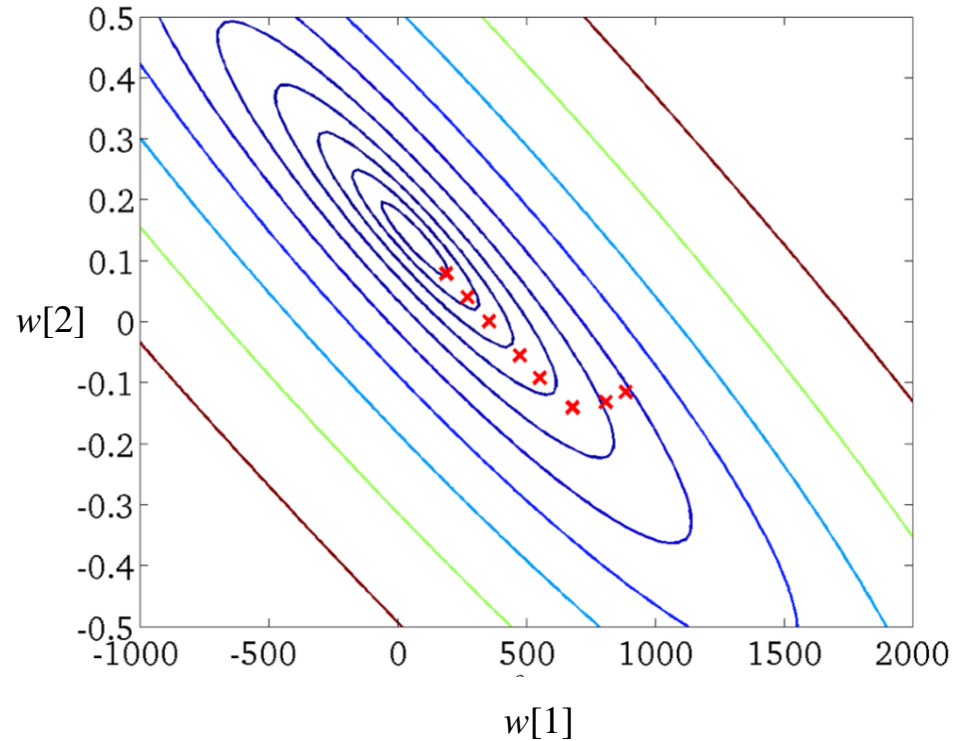
- $w_0 = (900, -0.1)$

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

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



Evolution of the predictor

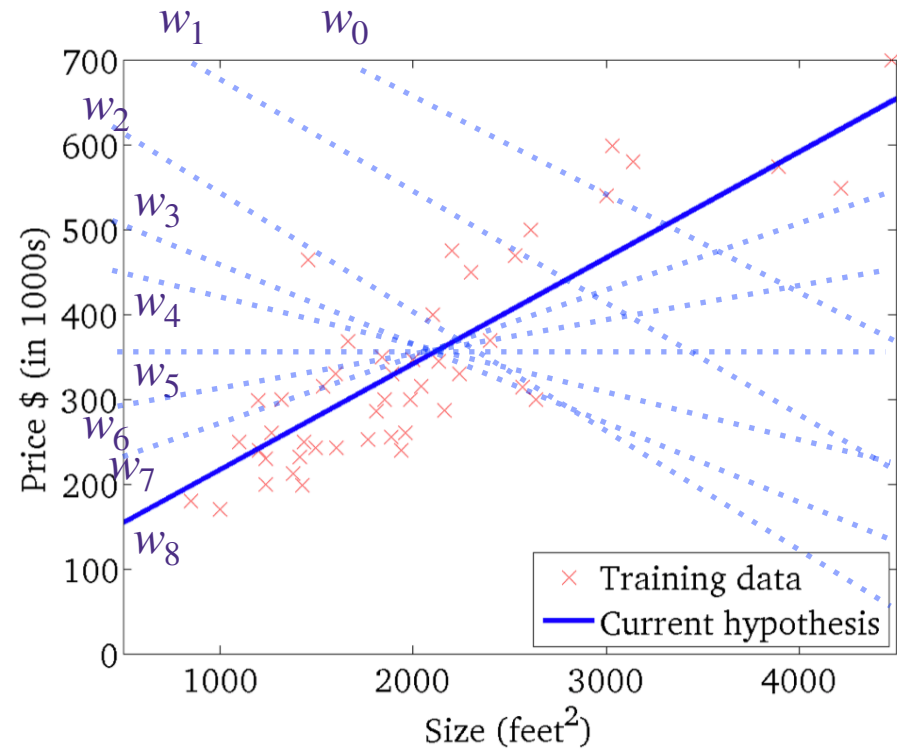


GD dynamics in the Parameter space

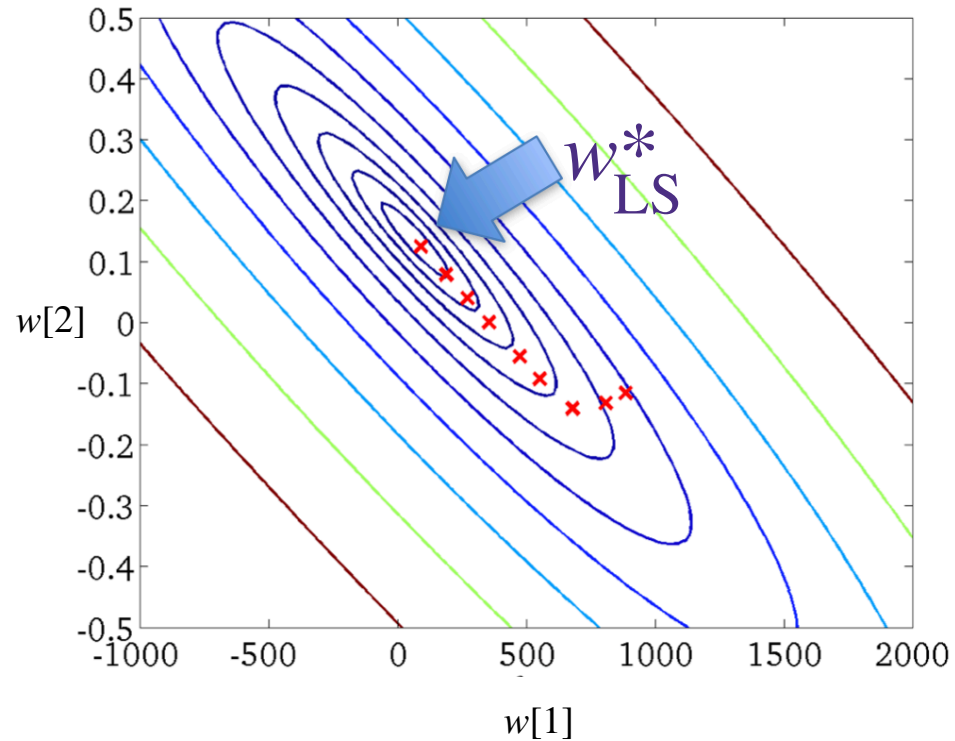
- $w_0 = (900, -0.1)$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

How do you choose step size?

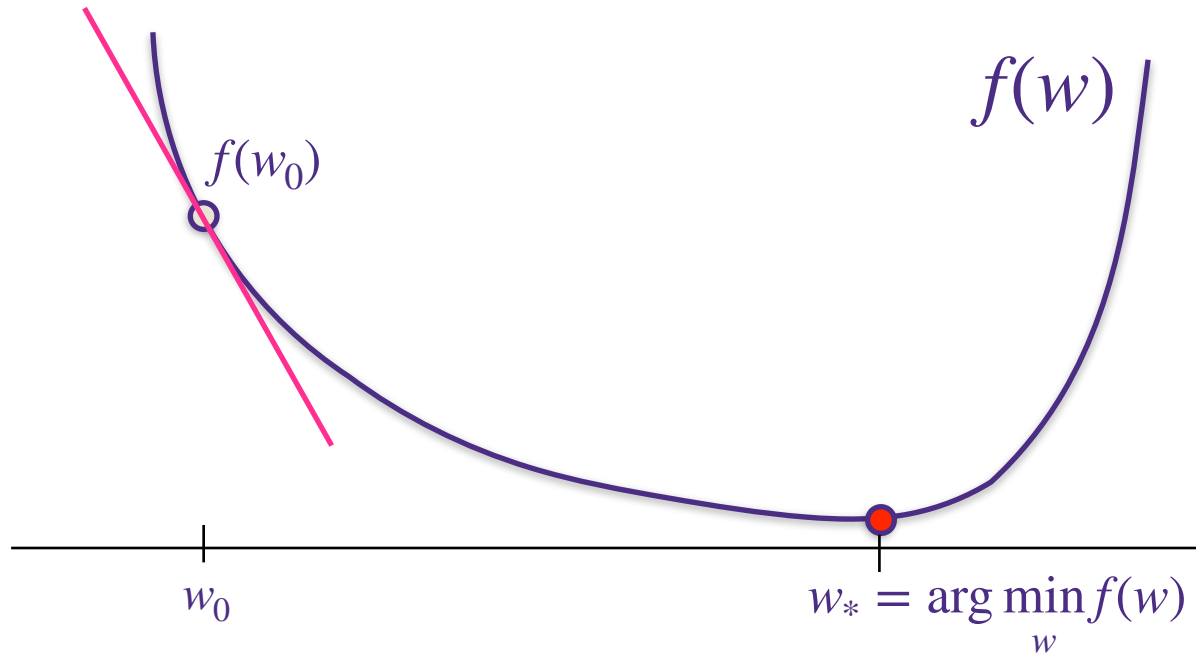
Let w_0 be an initial guess. How can we improve this solution?

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)

Gradient descent for **Ridge** 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 Ridge we have

$$\hat{w}_{\text{Ridge}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2} \|\mathbf{y} - \mathbf{X}w\|_2^2 + \frac{\lambda}{2} \|w\|_2^2}_{f(w)}$$

$$\nabla f(w_t) =$$

$$w_{t+1} =$$

Gradient descent for **Ridge** 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 Ridge we have

$$\hat{w}_{\text{Ridge}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2} \|\mathbf{y} - \mathbf{X}w\|_2^2 + \frac{\lambda}{2} \|w\|_2^2}_{f(w)}$$

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

$$w_{t+1} = (1 - \lambda)w_t + \eta \mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t)$$

Gradient descent for **Lasso** 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 Lasso we have

$$\hat{w}_{\text{Lasso}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2} \|\mathbf{y} - \mathbf{X}w\|_2^2 + \lambda \|w\|_1}_{f(w)}$$

$$\nabla f(w_t) =$$

$$w_{t+1} =$$

Gradient descent for **Lasso** 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 Lasso we have

$$\hat{w}_{\text{Lasso}} = \arg \min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2} \|\mathbf{y} - \mathbf{X}w\|_2^2 + \lambda \|w\|_1}_{f(w)}$$

$$\nabla f(w_t) = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) + \lambda \text{sign}(w_t)$$

$$w_{t+1} = w_t + \eta \mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) - \lambda \text{sign}(w_t)$$

Gradient Descent Demo

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:

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

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:

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

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

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 $\{1, \dots, n\}$

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

Stochastic Gradient Descent

Theorem

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

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

If $\|w_1 - w_0\|_2^2 \leq R$ and $\sup_w \max_i \|\nabla \ell_i(w)\|_2 \leq G$ then

$$\mathbb{E}[\ell(\bar{w}) - \ell(w_*)] \leq \frac{R}{2T\eta} + \frac{\eta G}{2} \leq \sqrt{\frac{RG}{T}} \quad \eta = \sqrt{\frac{R}{GT}}$$

$$\bar{w} = \frac{1}{T} \sum_{t=1}^T w_t$$

(In practice use last iterate)

Stochastic Gradient Descent

Proof

$$\mathbb{E}[\|w_{t+1} - w_*\|_2^2] = \mathbb{E}[\|w_t - \eta \nabla \ell_{I_t}(w_t) - w_*\|_2^2]$$

Stochastic Gradient Descent

Proof

$$\begin{aligned}\mathbb{E}[\|w_{t+1} - w_*\|_2^2] &= \mathbb{E}[\|w_t - \eta \nabla \ell_{I_t}(w_t) - w_*\|_2^2] \\ &= \mathbb{E}[\|w_t - w_*\|_2^2] - 2\eta \mathbb{E}[\nabla \ell_{I_t}(w_t)^T (w_t - w_*)] + \eta^2 \mathbb{E}[\|\nabla \ell_{I_t}(w_t)\|_2^2] \\ &\leq \mathbb{E}[\|w_t - w_*\|_2^2] - 2\eta \mathbb{E}[\ell(w_t) - \ell(w_*)] + \eta^2 G\end{aligned}$$

$$\begin{aligned}\mathbb{E}[\nabla \ell_{I_t}(w_t)^T (w_t - w_*)] &= \mathbb{E}[\mathbb{E}[\nabla \ell_{I_t}(w_t)^T (w_t - w_*) | I_1, w_1, \dots, I_{t-1}, w_{t-1}]] \\ &= \mathbb{E}[\nabla \ell(w_t)^T (w_t - w_*)] \\ &\geq \mathbb{E}[\ell(w_t) - \ell(w_*)]\end{aligned}$$

$$\begin{aligned}\sum_{t=1}^T \mathbb{E}[\ell(w_t) - \ell(w_*)] &\leq \frac{1}{2\eta} (\mathbb{E}[\|w_1 - w_*\|_2^2] - \mathbb{E}[\|w_{T+1} - w_*\|_2^2] + T\eta^2 G) \\ &\leq \frac{R}{2\eta} + \frac{T\eta G}{2}\end{aligned}$$

Stochastic Gradient Descent

Proof

Jensen's inequality:

For any random $Z \in \mathbb{R}^d$ and convex function $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$, $\phi(\mathbb{E}[Z]) \leq \mathbb{E}[\phi(Z)]$

$$\mathbb{E}[\ell(\bar{w}) - \ell(w_*)] \leq \frac{1}{T} \sum_{t=1}^T \mathbb{E}[\ell(w_t) - \ell(w_*)]$$

$$\bar{w} = \frac{1}{T} \sum_{t=1}^T w_t$$

Mini-batch SGD

- 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

- 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

Questions?

Convexity

- When is an optimization (or learning) easy/fast to solve?

Recap: Ridge vs. Lasso

- **Ridge**

$$\text{minimize}_w \sum_{i=1}^n (w^T x_i - y_i)^2 + \lambda \|w\|_2^2$$

- Very fast:
 - Closed form solution if used with linear models
 - Even with other loss functions, optimization is fast for squared ℓ_2 regularization, because $\|w\|_2^2$ is **convex and smooth**

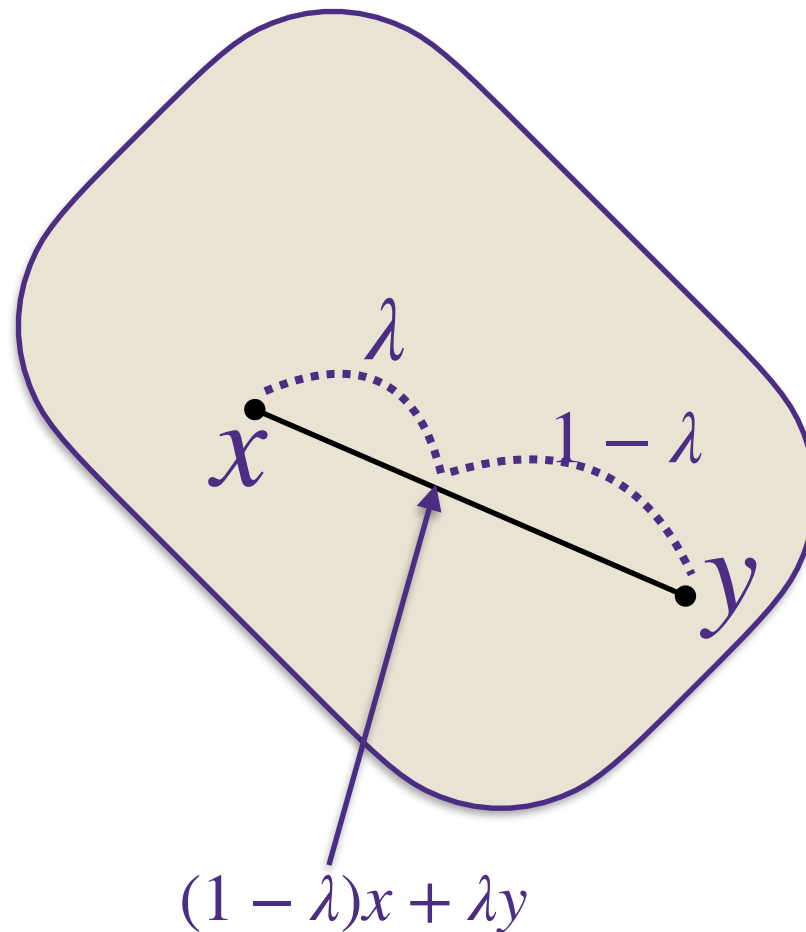
- **Lasso**

$$\text{minimize}_w \sum_{i=1}^n (w^T x_i - y_i)^2 + \lambda \|w\|_1$$

- Slower than Ridge:
 - Requires iterative optimization algorithm like sub-gradient descent
 - In particular, it is slower because $\|w\|_1$ is **convex but non-smooth**

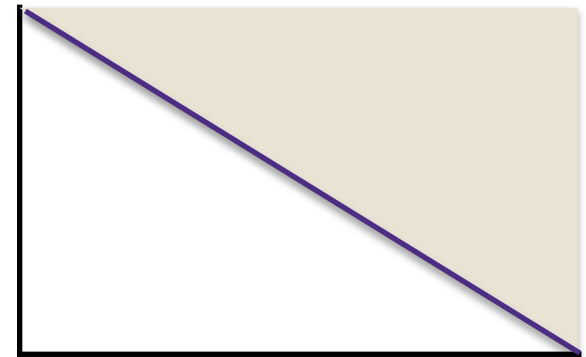
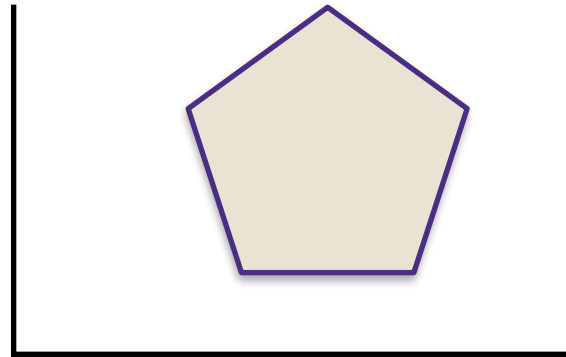
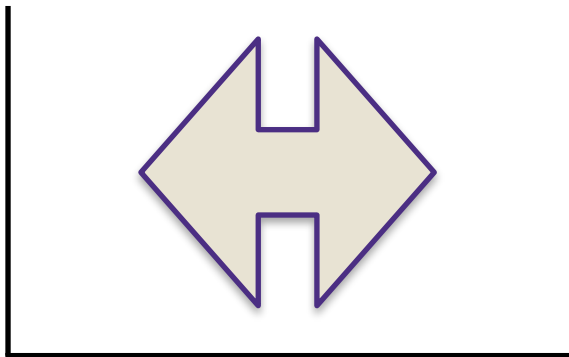
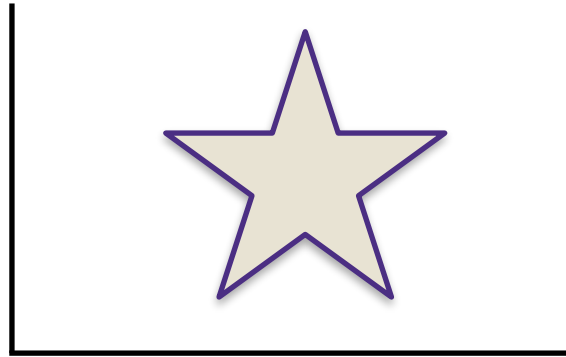
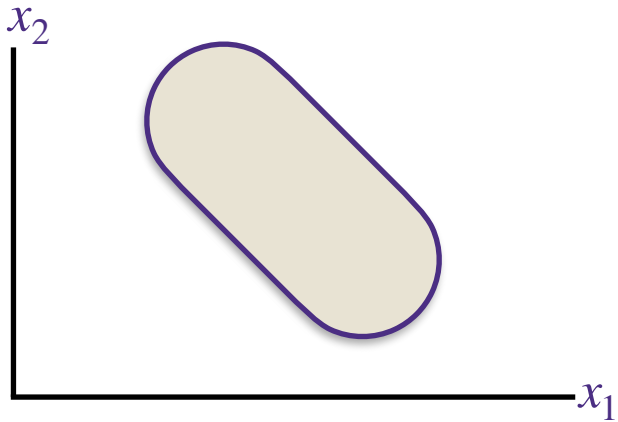
What is a convex set?

A set $K \subset \mathbb{R}^d$ is convex if $(1 - \lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0, 1]$



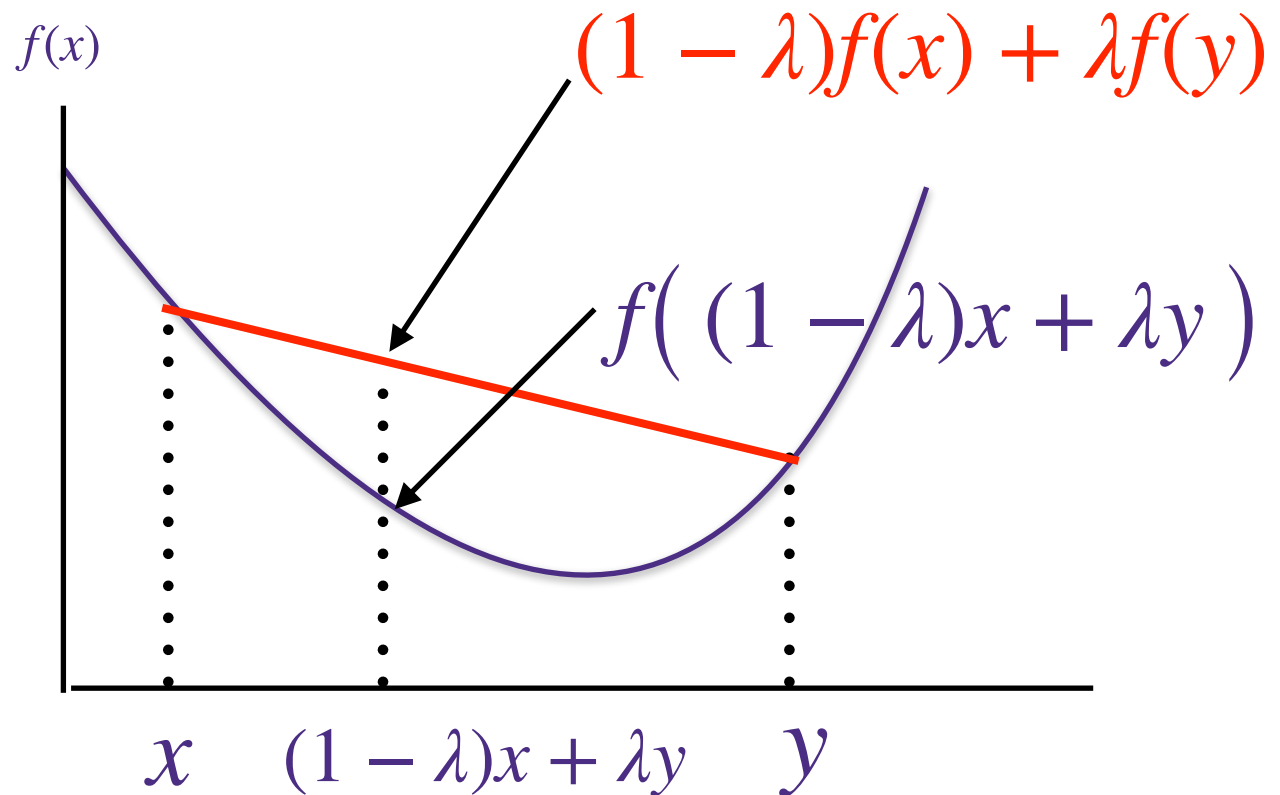
What is a convex set?

A set $K \subset \mathbb{R}^d$ is convex if $(1 - \lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0, 1]$



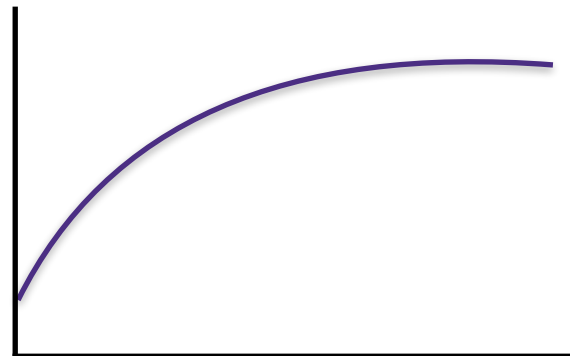
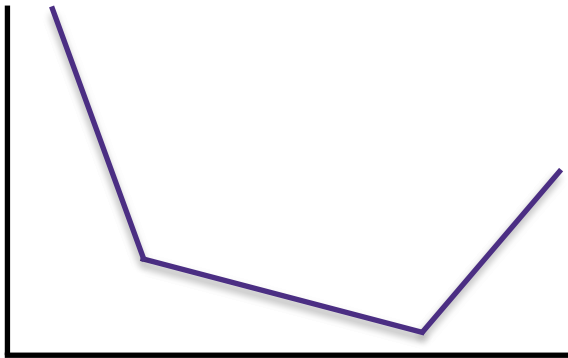
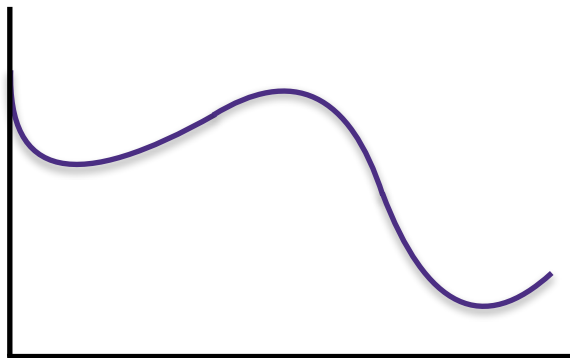
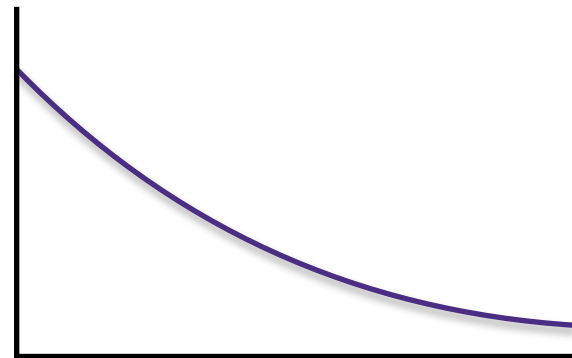
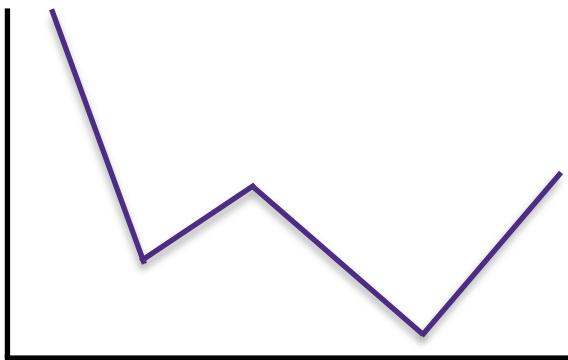
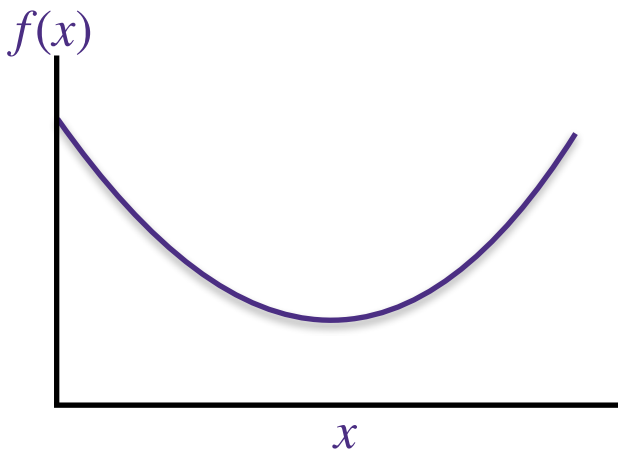
What is a convex function?

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if $f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$



What is a convex function?

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if $f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$



Convex functions and convex sets?

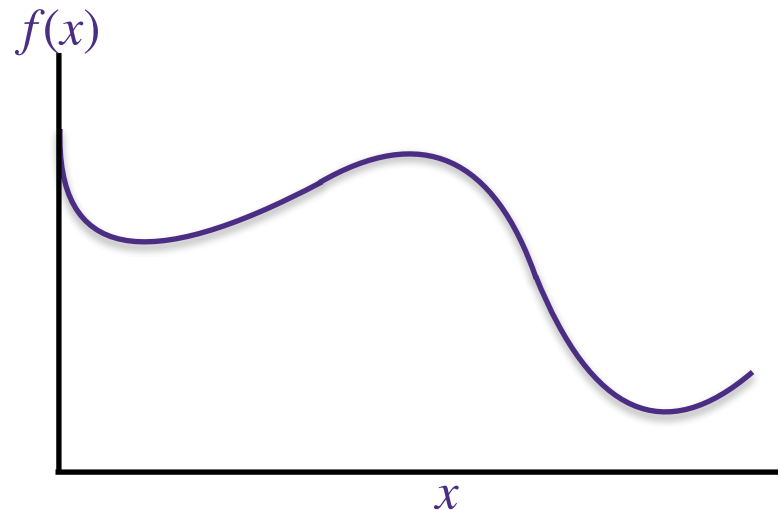
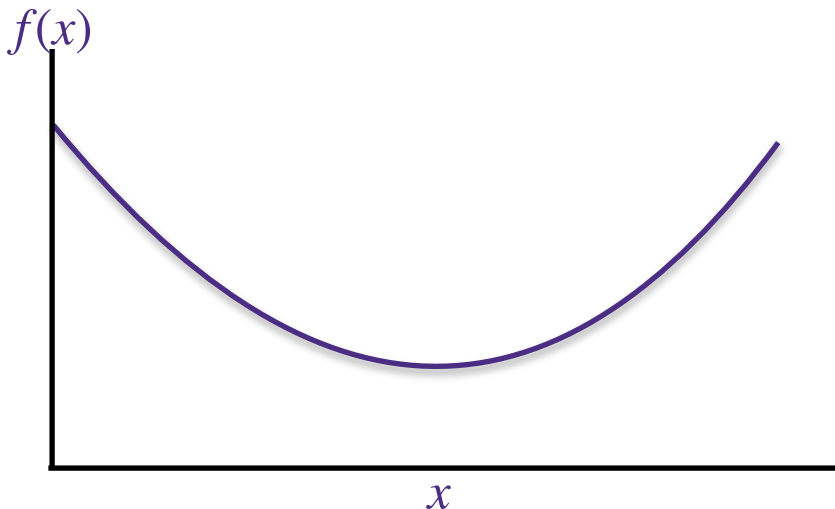
A set $K \subset \mathbb{R}^d$ is convex if $(1 - \lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0, 1]$

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if $f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if the set $\{(x, t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

Graph of f is defined as $\{(x, t) : f(x) = t\}$

Epigraph of f is defined as $\{(x, t) : f(x) \leq t\}$

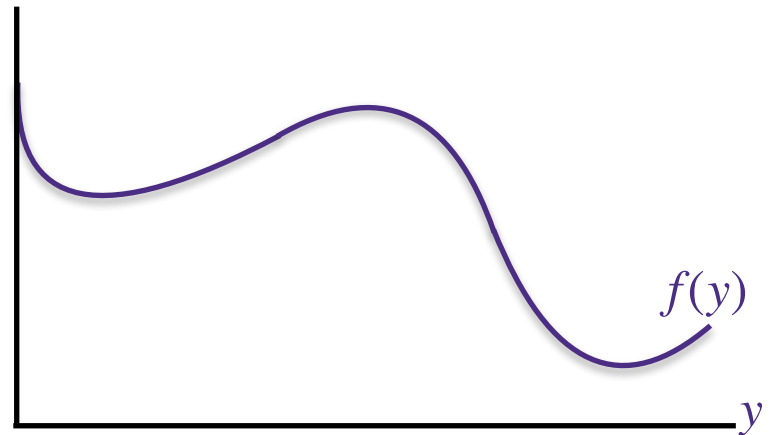
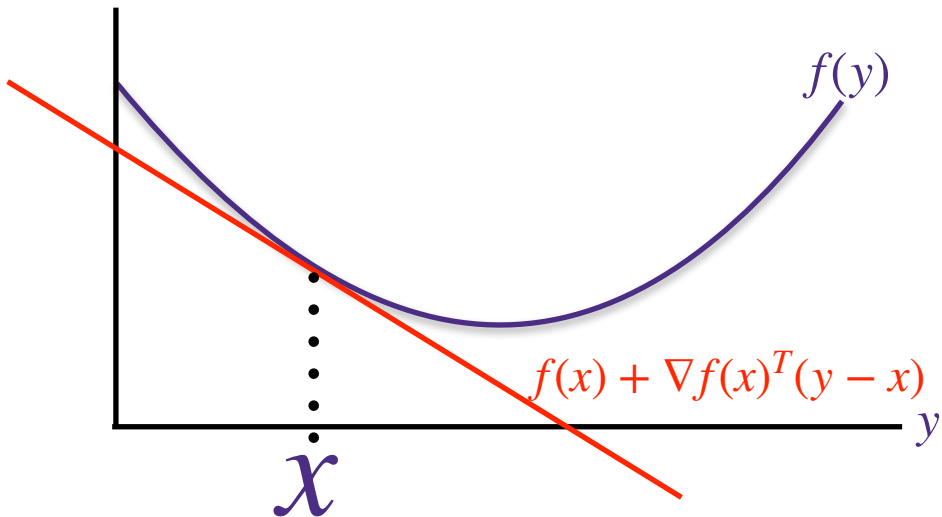


More definitions of convexity

A set $K \subset \mathbb{R}^d$ is convex if $(1 - \lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0, 1]$

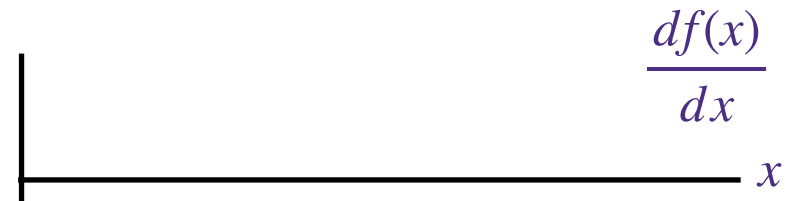
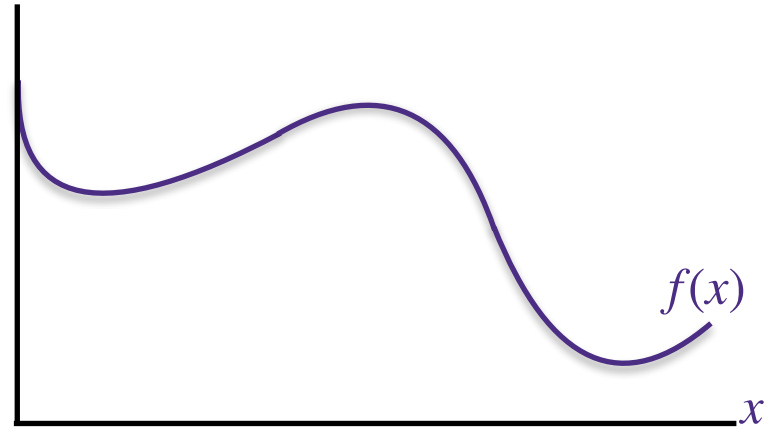
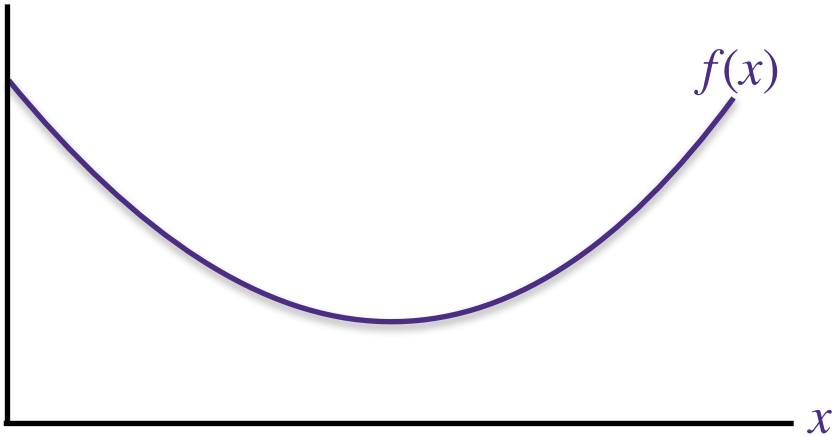
A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if the set $\{(x, t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that is differentiable everywhere is convex if $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in \text{dom}(f)$



More definitions of convexity

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in \text{dom}(f)$



More definitions of convexity

A set $K \subset \mathbb{R}^d$ is convex if $(1 - \lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0, 1]$

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if $f((1 - \lambda)x + \lambda y) \leq (1 - \lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if the set $\{(x, t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that is differentiable everywhere is convex if $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in \text{dom}(f)$

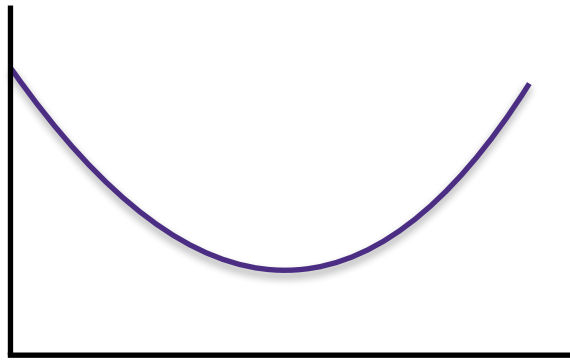
A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in \text{dom}(f)$

Why do we care about convexity?

Convex functions

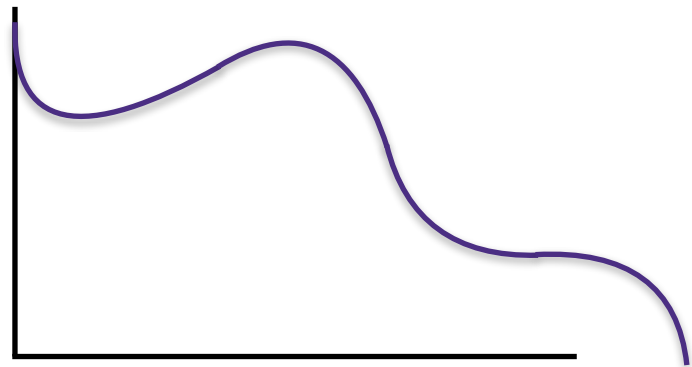
- All local minima are global minima
- Efficient to optimize (e.g., gradient descent)

Convex Function



We only need to find a point with $\nabla f(x) = 0$, which for convex functions implies that it is a local minima and a global minima

Non-convex Function



For non-convex functions, a stationary point with $\nabla f(x) = 0$ could be a local minima, a local maxima, or a saddle point

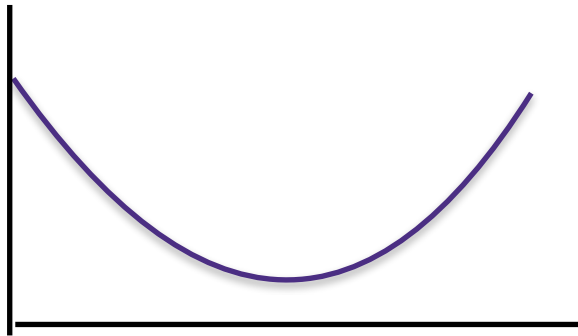
Gradient Descent on $\min_w f(w)$

Initialize: $w_0 = 0$

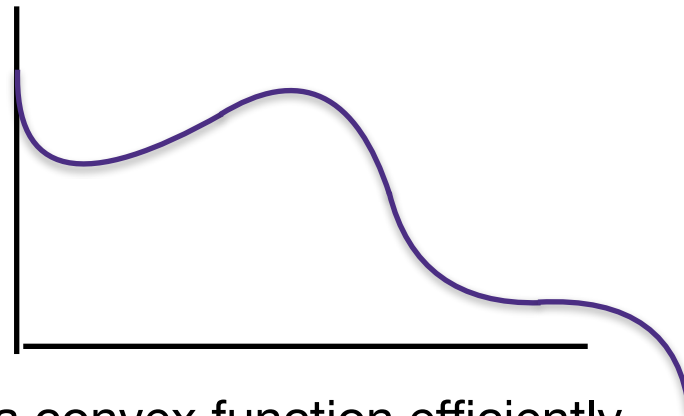
for $t = 1, 2, \dots$

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

Convex Function



Non-convex Function



- Strength: Can find global minima of a convex function efficiently
- Weakness: Can only be applied to smooth functions
 - i.e., functions that is differentiable everywhere,
 - otherwise $\nabla f(x)$ is not defined and gradient descent cannot be applied

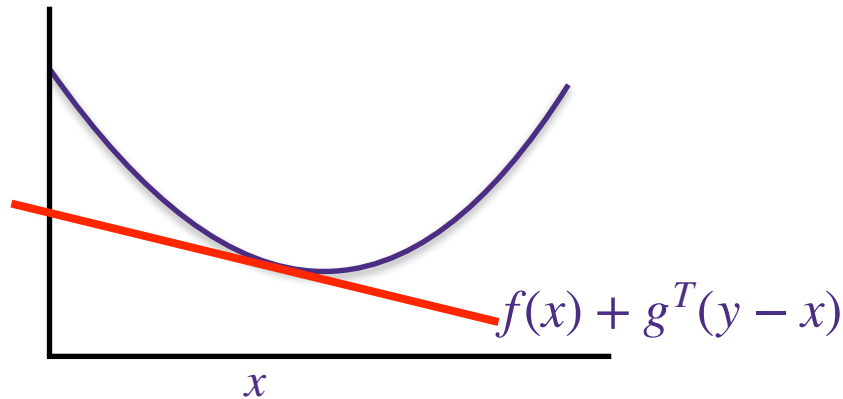
Sub-Gradient

Definition: a function is **non-smooth** if it is not differentiable everywhere

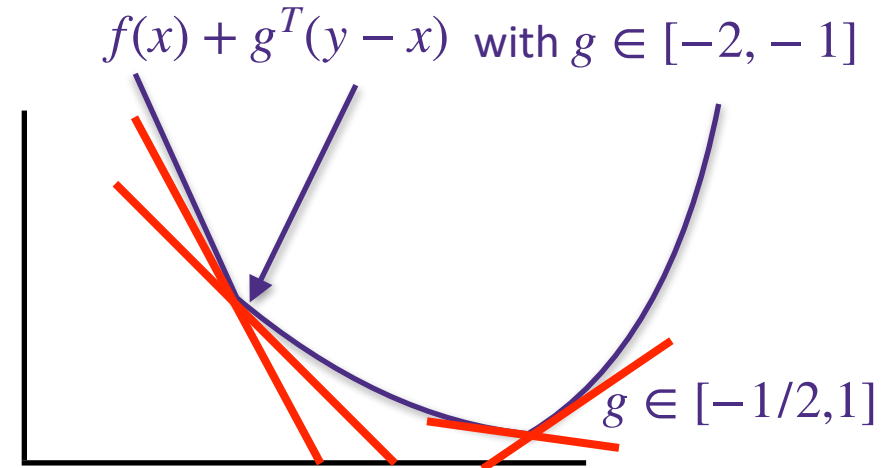
Definition: a vector $g \in \mathbb{R}^d$ is a **sub-gradient** at x if it satisfies

$$f(y) \geq f(x) + g^T(y - x) \text{ for all } y \in \mathbb{R}^d$$

Smooth Convex Function



Non-smooth Convex Function



- for smooth convex functions,

- gradient is the unique sub-gradient, and
- the global minimum is achieved at points where gradient is zero

- for non-smooth convex functions,

- the minimum is achieved at points where sub-gradient set includes the zero vector

Sub-Gradient Descent for non-smooth functions

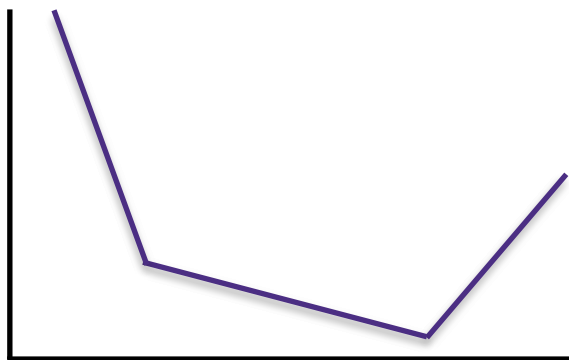
Initialize: $w_0 = 0$

for $t = 1, 2, \dots$

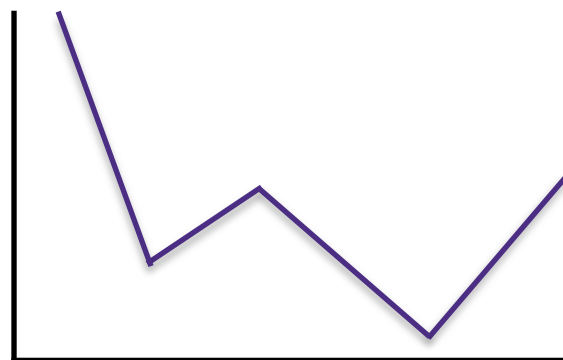
Find any g_t such that $f(y) \geq f(w_t) + g_t^\top (y - w_t)$

$$w_{t+1} \leftarrow w_t - \eta_t g_t$$

Convex Function



Non-convex Function



- Strength: finds global minima for **non-smooth convex functions**
- Weakness: it is slower than gradient descent on convex smooth functions, because the gradient do not get smaller near the global minima
 - Instead of last iterate w_t , we use the best one we saw in all iterates
 - The stepsize needs to decrease with t

Optimization

- You can always run gradient descent whether f is convex or not. But you only have guarantees if f is convex
- Many bells and whistles can be added onto gradient descent such as momentum and dimension-specific step-sizes (Nesterov, Adagrad, ADAM, etc.)

Questions?
