Important Announcements and Notes (Oct 22)

- Old lecture notes for optimization II had some mistakes. Use the new version on the website

- Course feedback survey
  (Thank you for the kind comments ☺)
  - more examples
  - post course notes earlier
  - Increase pace (have pre-filled in parts)

Important Announcements and Notes (Oct 15)

- If $g(\vec{w})$ then $\frac{\partial g}{\partial w_j}(\vec{w})$ is a function of $\vec{w}$

  not just $\frac{\partial g}{\partial w_j}(w_j)$

- Go over example for $A^{-1}\vec{b}$

Gradient Descent

Can we always find a closed form expression for $w^* = \arg\min_{w \in \mathcal{W}} g(w)$?

Ans: No

Ex: $g(w) = w^2 + e^w$, $g'(w) = 2w + e^w$, $g''(w) = 2 + e^w \geq 0$

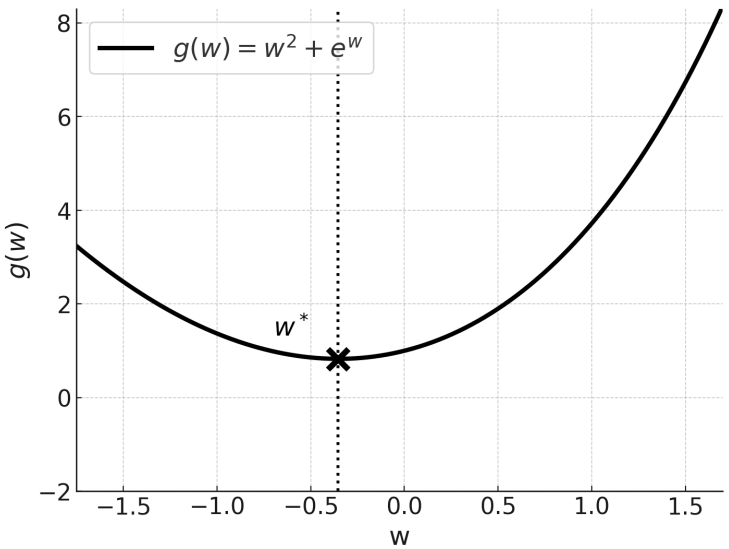
Convex

$w \in \mathbb{R} = \mathcal{W}$

$$g'(w) = 2w + e^w = 0$$

$$2w = -e^w$$

No closed form solution



Gradient descent will help us

Second-Order Gradient Descent (Newton-Raphson)

If $g(w)$ is a degree 5 polynomial or less. Then there exists a closed form solution for $g'(w) = 0$

Our approach: Lets approximate $g(w)$ with convex low order polynomial (i.e. second order polynomial)

In general, Taylor series at a point $w^{(0)}$ of $g(w)$ is:

$$g(w) = \sum_{n=0}^{\infty} \frac{g^{(n)}(w^{(0)})}{n!} (w - w^{(0)})^n$$

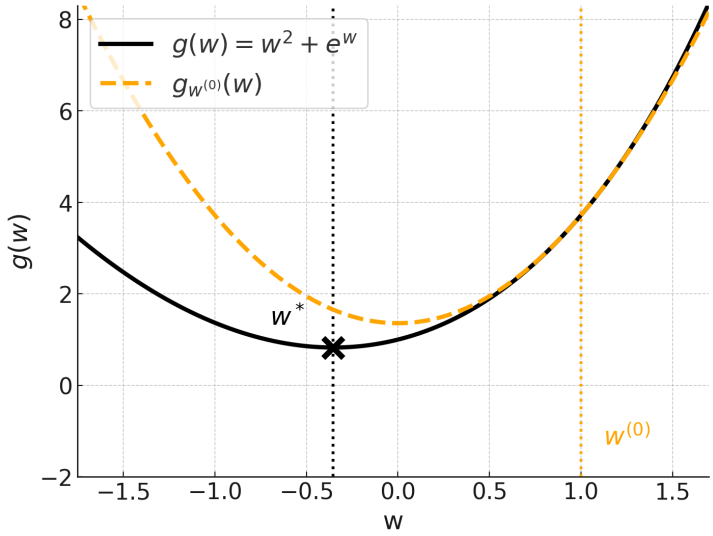
We use a second order approx: take the first 3 terms.

$$g(w) \approx g_{w^{(0)}}(w) = g(w^{(0)}) + g'(w^{(0)})(w - w^{(0)}) + \frac{g''(w^{(0)})}{2} (w - w^{(0)})^2$$

Ex: $g(w) = w^2 + e^w$

$w^{(0)} = 1$

$g_{w^{(0)}}(w) = g_1(w)$



minimizing $g_{w^{(0)}}(w) = g(w^{(0)}) + g'(w^{(0)})(w - w^{(0)}) + \frac{g''(w^{(0)})}{2}(w - w^{(0)})^2$

$g'_{w^{(0)}}(w) = g'(w^{(0)}) + g''(w^{(0)})(w - w^{(0)}) = 0$

$\Rightarrow g''(w^{(0)})\, w = g''(w^{(0)})\, w^{(0)} - g'(w^{(0)})$

$\Rightarrow w = w^{(0)} - \frac{g'(w^{(0)})}{g''(w^{(0)})}$

$w^{(1)} = w$

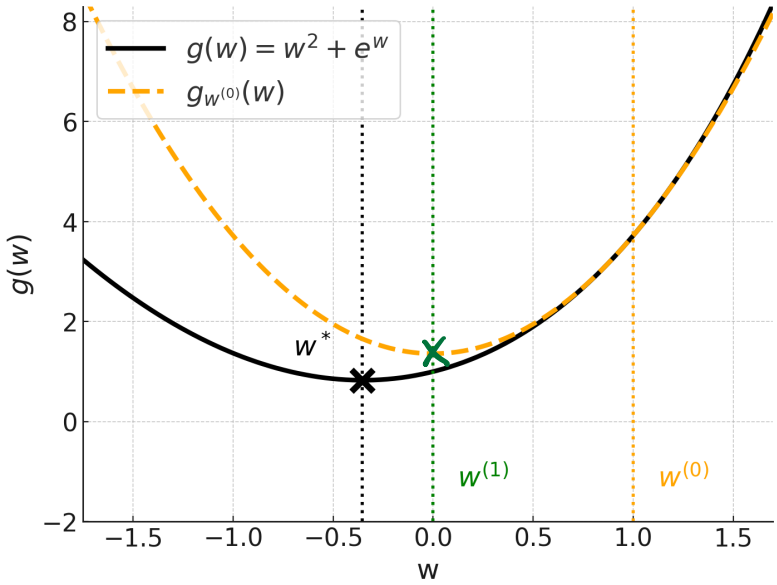
Ex: $w^{(0)} = 1$

$g'(1) = 2 \cdot 1 + e^1$

$g''(1) = 2 + e^1$

$w^{(1)} = w^{(0)} - \frac{g'(1)}{g''(1)}$

$= 1 - 1 = 0$

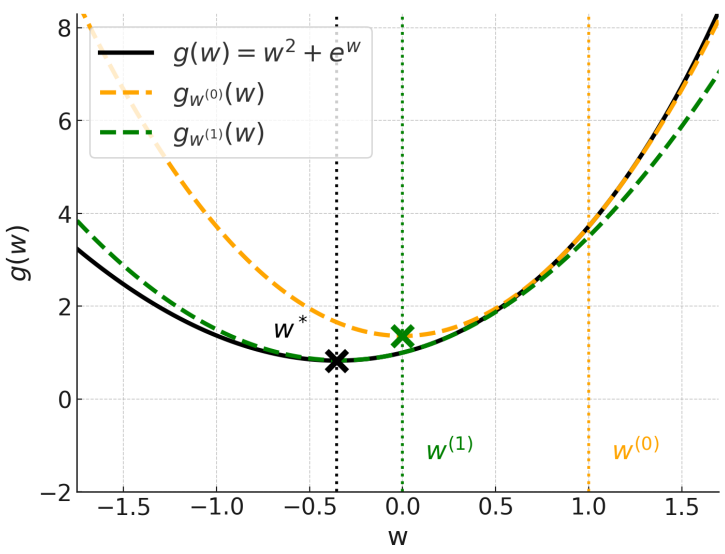


We can approximate $g(w)$ again but at $w^{(1)}$

$$g_{w^{(1)}}(w) = g(w^{(1)}) + g'(w^{(1)})(w - w^{(1)}) + \frac{g''(w^{(1)})}{2}(w - w^{(1)})^2$$

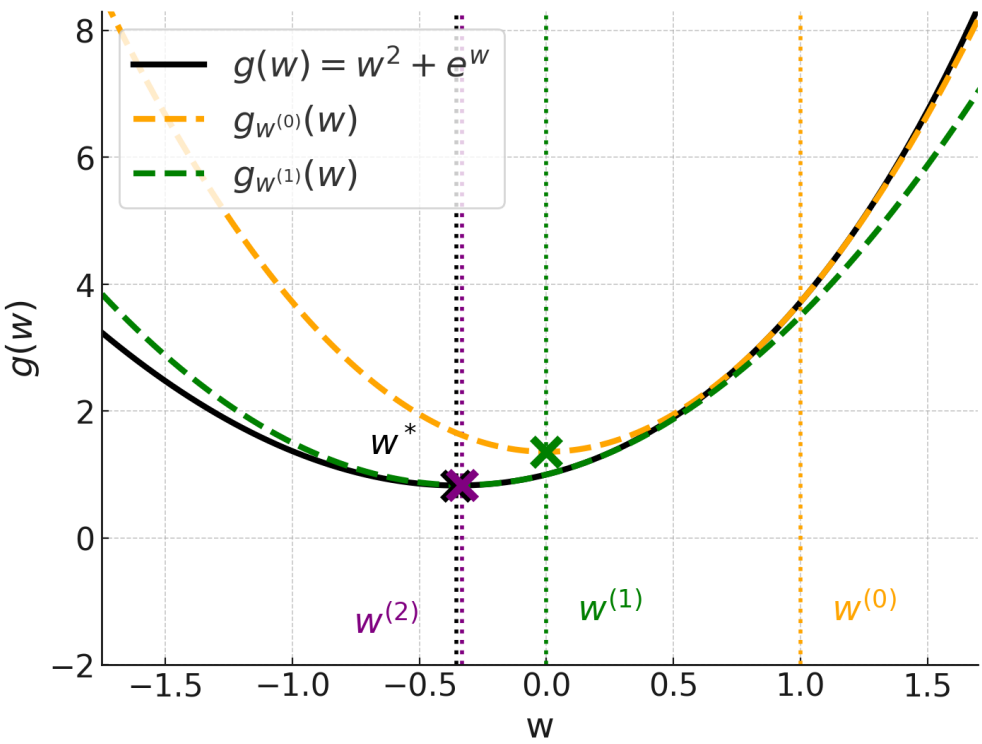
Ex: $w^{(1)} = 0$

$g_{w^{(1)}}(w)$



$$w^{(2)} = w^{(1)} - \frac{g'(w^{(1)})}{g''(w^{(1)})}$$

Ex: $w^{(2)} = -\frac{1}{3}$



In general $w^{(t+1)} = w^{(t)} - \frac{g'(w^{(t)})}{g''(w^{(t)})}$

$w^{(t+1)}$ approaches w^* as $t \to \infty$

(First-order) Gradient Descent

Sometimes it is hard to calculate $g''(w)$ especially in high dimensions

$$w^{(t+1)} = w^{(t)} - \eta^{(t)} g'(w^{(t)})$$

$\eta^{(t)}$ is the "step size" "Learning rate"

if we knew $g''(w)$ we would set $\eta^{(t)} = \frac{1}{g''(w^{(t)})}$

Ex: $\eta^{(t)} = 1/g''(w^{(t)})$

$g(w) = w^2 + e^w$

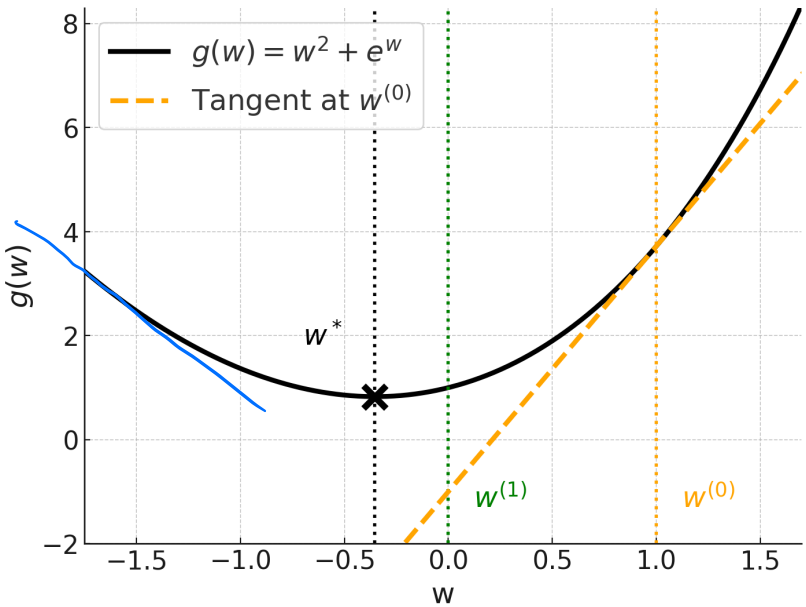
$g'(w) = 2w + e^w$

$g''(w) = 2 + e^w$

$w^{(0)} = 1, \eta^{(0)} = \frac{1}{2+e^1}$

$w^{(1)} = w^{(0)} - \eta^{(0)}(g'(w^{(0)})) = 0$

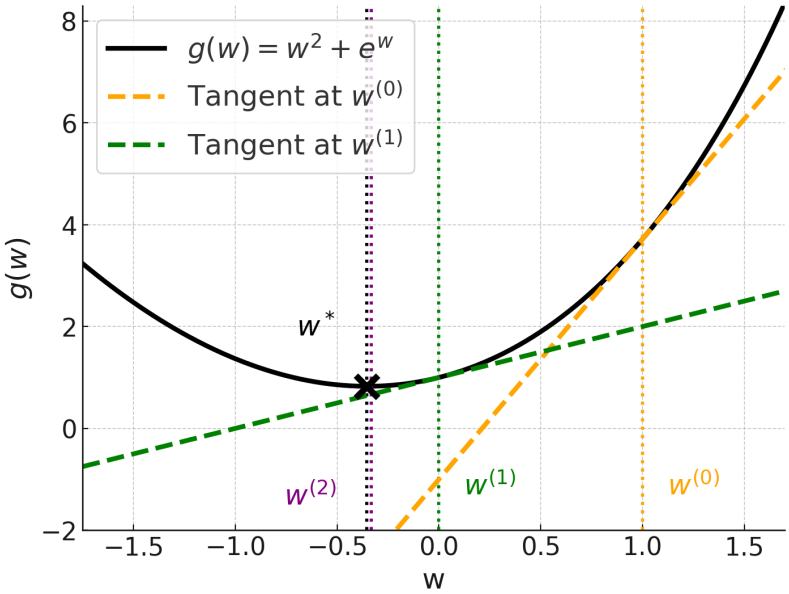
$= 1 - 1 = 0$



g' gives the direction of ascent for g

$$\eta^{(1)} = \frac{1}{2+e^0} = \frac{1}{3}$$

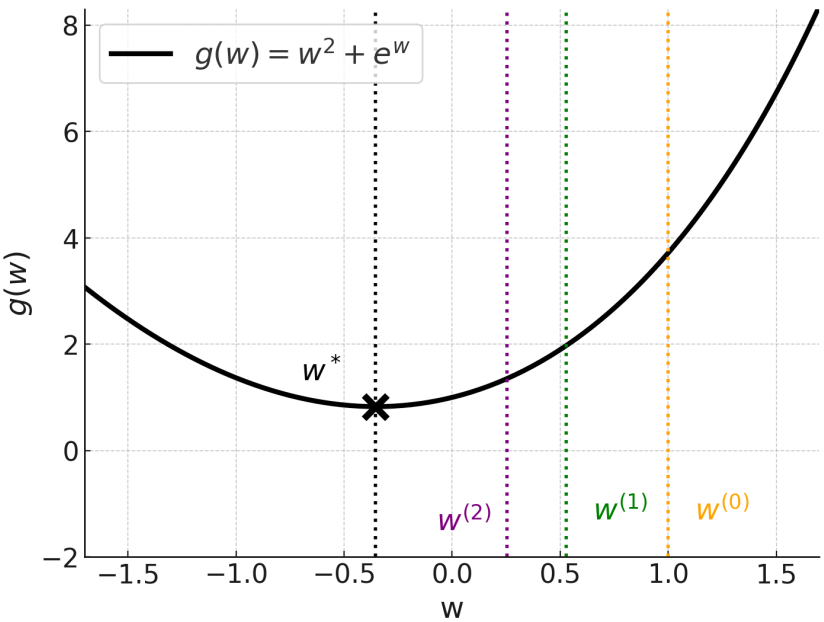
$$w^{(2)} = w^{(1)} - \eta^{(1)}\left(g'(w^{(1)})\right)$$



Ex: $\eta^{(t)} = \frac{1}{10} < \frac{1}{2+e}$

Small $\eta^{(t)}$

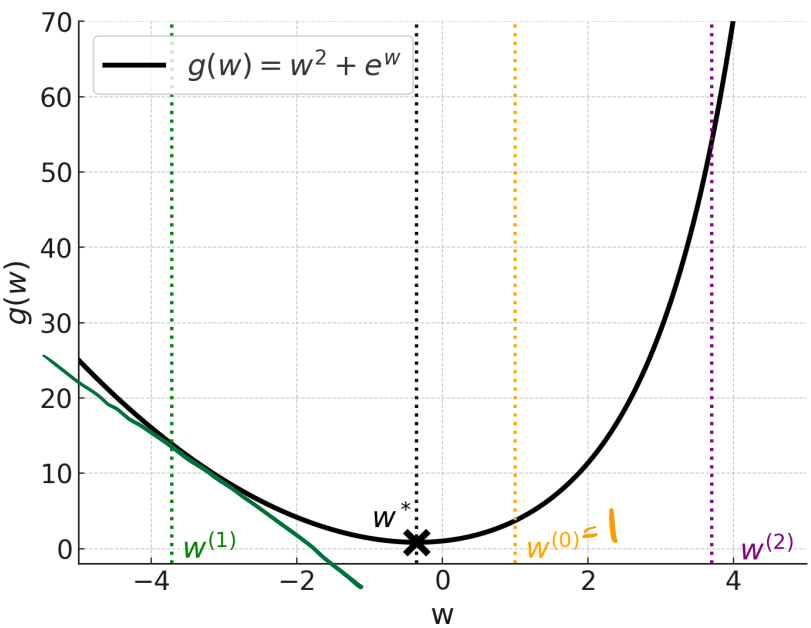
Slowly reaches w^*



Ex: $\eta^{(t)} = 1 > \frac{1}{2+e}$

Large $\eta^{(t)}$

might never reach w^* (Diverge)



- Small step size is safe (you will probably converge to w^*) however it might take long
- Large step size can get you to w^* fast, but sometimes it might diverge

Multivariate Gradient Descent

$\vec{w} \in \mathcal{W} = \mathbb{R}^d$, $d > 1$, $g(\vec{w})$

Objective:

$$\vec{w}^* = (w_1^*, \ldots, w_d^*)^T = \underset{\vec{w} \in \mathcal{W}}{\operatorname{argmin}}\, g(\vec{w})$$

gradient descent update rule:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta^{(t)} \nabla g(\vec{w}^{(t)})$$

where $\nabla g(\vec{w}^{(t)}) = \left(\frac{\partial g}{\partial w_1}(\vec{w}), \ldots, \frac{\partial g}{\partial w_d}(\vec{w}) \right)^T \in \mathbb{R}^d$

Ex: $W = \mathbb{R}^2$, $g(\vec{w}) = g(w_1, w_2) = w_1^2 + e^{w_1} + w_2^2 + e^{w_2}$

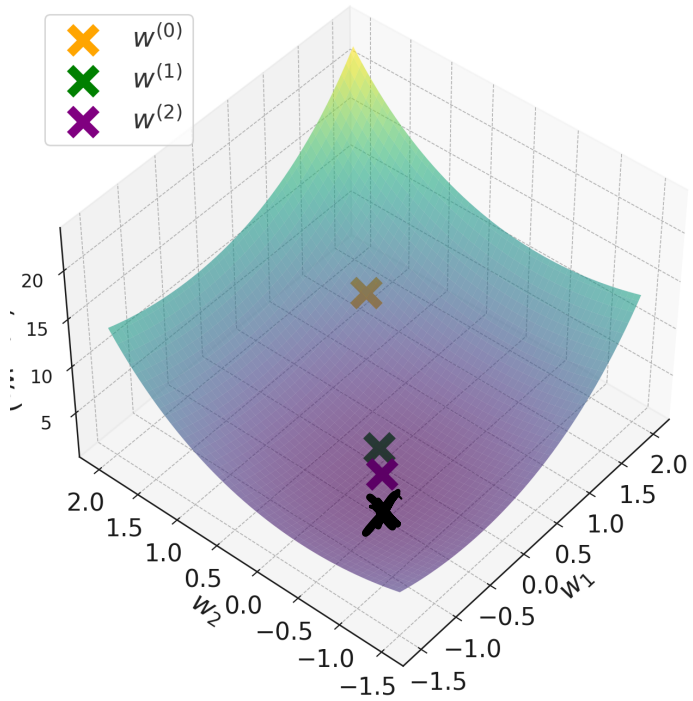
$$\frac{\partial g}{\partial w_1}(\vec{w}) = 2w_1 + e^{w_1}, \quad \frac{\partial g}{\partial w_2}(\vec{w}) = 2w_2 + e^{w_2}$$

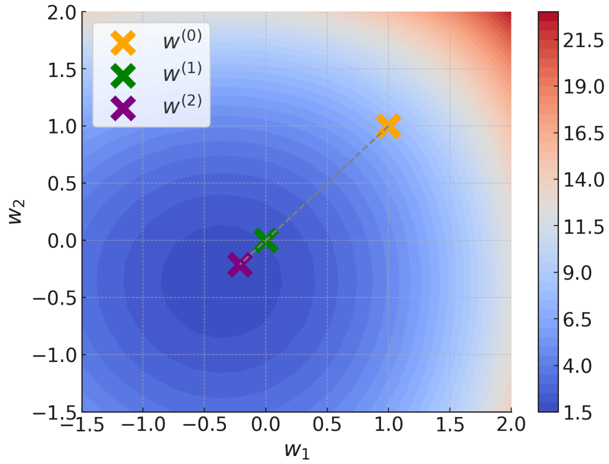
$$\vec{w}^{(t+1)} = \begin{pmatrix} w_1^{(t+1)} \\ w_2^{(t+1)} \end{pmatrix} = \begin{pmatrix} w_1^{(t)} \\ w_2^{(t)} \end{pmatrix} - \eta^{(t)} \begin{pmatrix} 2w_1^{(t)} + e^{w_1^{(t)}} \\ 2w_2^{(t)} + e^{w_2^{(t)}} \end{pmatrix}$$

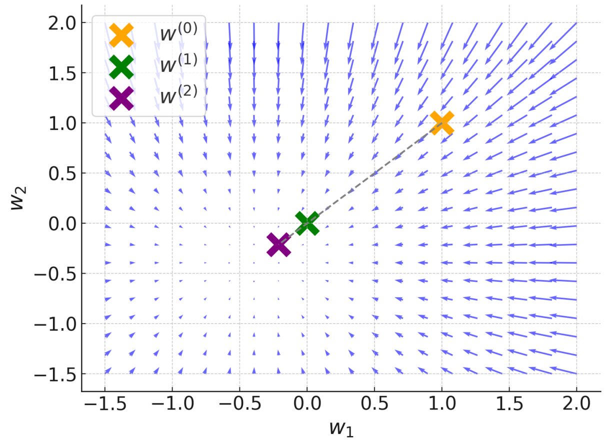
$$\vec{w}^{(0)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \eta^{(0)} = \frac{1}{2+e}$$

$$\vec{w}^{(1)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \frac{1}{2+e} \begin{pmatrix} 2+e \\ 2+e \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

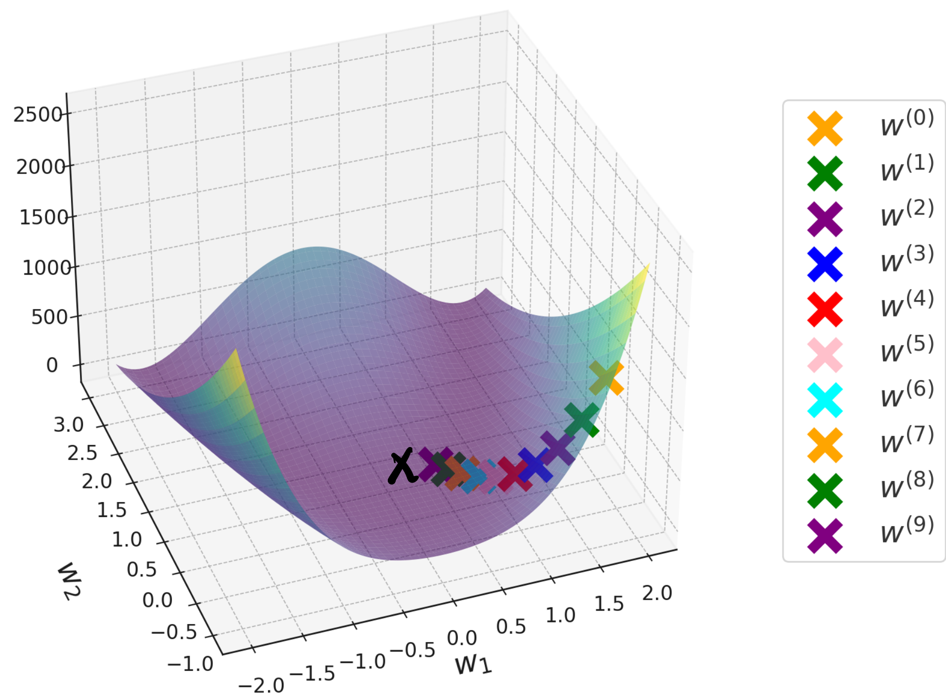
$$\eta^{(1)} = \frac{1}{2+e} \qquad \vec{w}^{(2)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \frac{1}{2+e} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

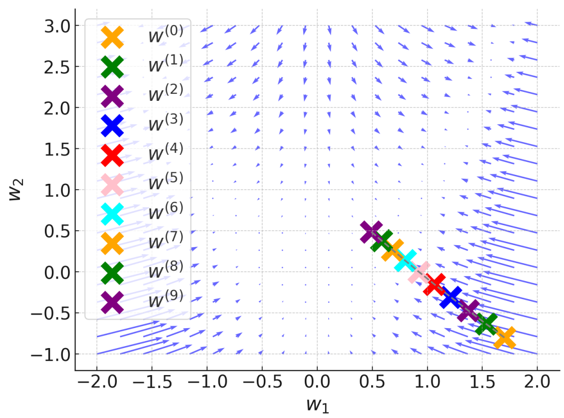


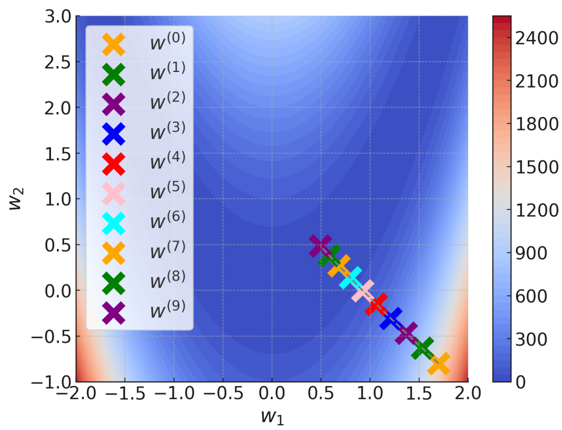




Ex: $g(w_1, w_2) = (1 - w_1)^2 + 100(w_2 - w_1^2)^2$ $\qquad \eta^{(t)} = \frac{1}{6}$







Selecting the step size

Common (and simple) options ($t \in \mathbb{N}$)

1. constant: $\eta^{(t)} = \eta \in (0, \infty)$

2. Inverse decaying: $\eta^{(t)} = \frac{\eta}{1+\lambda t} \approx \frac{1}{t}$ where $\eta, \lambda \in (0, \infty)$

3. Exponential decaying: $\eta^{(t)} = \eta \frac{1}{e^{\lambda t}}$ where $\eta, \lambda \in (0, \infty)$

4. Normalized gradient: $\eta^{(t)} = \frac{\eta}{\varepsilon + \|\nabla g(\vec{w}^{(t)})\|}$ where $\eta, \varepsilon \in (0, \infty)$

ε is small

(ex: $\varepsilon = 10^{-8}$)

$$\|\nabla g(\vec{w}^{(t)})\| = \sqrt{\sum_{j=1}^{d} \left(\frac{\partial g}{\partial w_j}(\vec{w}^{(t)})\right)^2}$$

Gradient Descent with $\hat{L}(f)$

$$D = ((\vec{x}_1, y), \ldots, (\vec{x}_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n$$

$\mathcal{X} = \mathbb{R}^{d+1}$, $\mathcal{Y} = \mathbb{R}$ regression

$$\mathcal{F} = \{ f \mid f : \mathbb{R}^{d+1} \to \mathbb{R} \text{ and } f(\vec{x}) = \vec{x}^T \vec{w},\ \vec{w} \in \mathbb{R}^{d+1} \}$$

$$\ell(\hat{y}, y) = (\hat{y} - y)^2 \quad \text{squared loss}$$

$$\text{objective: } \hat{\vec{w}} = \operatorname{argmin}_{\vec{w} \in \mathbb{R}^{d+1}} \hat{L}(\vec{w})$$

$$\hat{L}(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i)^2$$

(Batch) Gradient Descent (BGD)

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} - \eta^{(t)} \nabla \hat{L}(\vec{w}^{(t)})$$

$$\nabla \hat{L}(\vec{w}^{(t)}) = \left(\frac{\partial \hat{L}}{\partial w_0}(\vec{w}^{(t)}), \ldots, \frac{\partial \hat{L}}{\partial w_d}(\vec{w}^{(t)}) \right)^T \in \mathbb{R}^{d+1}$$

$$= \left(\frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) x_{i0}, \ldots, \frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) x_{id} \right)^T$$

$$= \frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) \underbrace{(x_{i0}, \ldots, x_{id})^T}_{\vec{x}_i}$$

$$= \frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) \vec{x}_i$$

Algorithm: BGD Linear Regression Learner
(with a constant size)

---

input: $D = ((\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n))$, η, T number of "epochs"

$\vec{w} \leftarrow$ random vector in $\mathbb{R}^{d+1}$

for $t = 1, \ldots, T$

$$\nabla \hat{L}(\vec{w}) \leftarrow \frac{2}{n} \sum_{i=1}^{n} (\vec{x}_i^T \vec{w} - y_i) \vec{x}_i$$

$$\vec{w} = \vec{w} - \eta \nabla \hat{L}(\vec{w})$$

return $\hat{f}(\vec{x}) = \vec{x}^T \vec{w}$

Computation of Closed form Solution
vs. BGD

number of computational steps: number of elementary operations (adding, subtracting, multiplying

Big "O" notation:

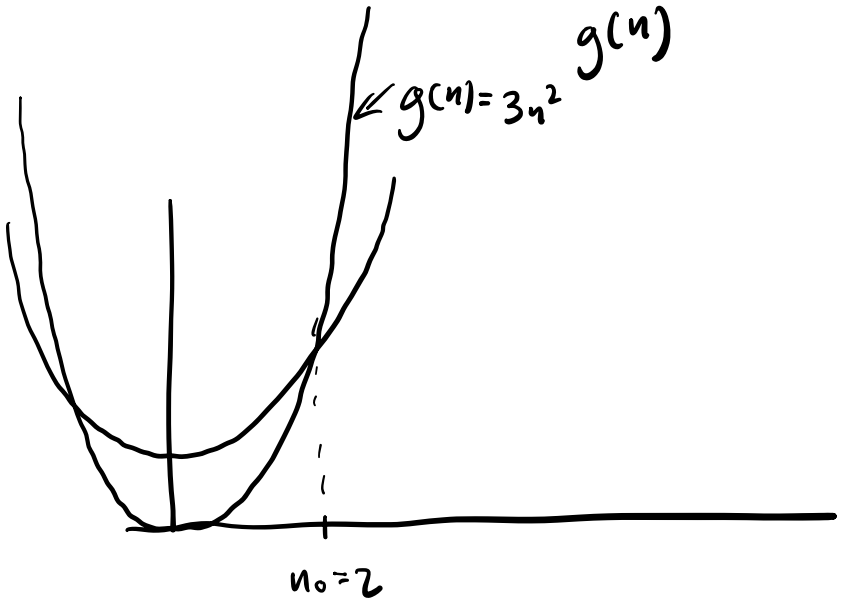
$f(n) = O(g(n))$ $f(n) \leq C g(n)$ for $n \geq n_0$

$2 \cdot 4 + 3 \leq 3 \cdot 4$

$2n^2 + 3 \leq 3n^2$

in $n \geq 2$

Ex: $f(n) = 2n^2 + 3 = O(n^2)$, with $g(n)$



Closed form: $\hat{\vec{w}} = A^{-1}\vec{b}$ where $A = \sum_{i=1}^{n} \vec{x}_i \vec{x}_i^T$, $\vec{b} = \sum_{i=1}^{n} \vec{x}_i y_i$

$O(d^2)$

Computation A: $O(nd^2)$

$\vec{b}$: $O(nd)$

A^{-1}: $O(d^3)$

$A^{-1}\vec{b}$: $O(d^2)$

Total computation: $O(nd^2 + nd + d^3 + d^2)$

$= O(nd^2 + d^3)$

BBD:

computing $\nabla \hat{L}(\vec{w})$: $O(nd)$

$\vec{w}$: $O(d)$

Total compute: $O(ndT)$

Comparing: $O(nd^2+d^3)$ and $O(ndT)$

if $T<d$: $O(ndT) < O(nd^2) \leq O(nd^2+d^3)$

BGD is more computationally efficient if $T<d$

Can we do even better (in terms of compute)?

Mini-Batch Gradient Descent (MBGD)

$$D = \Big((\vec{x}_1, y_1), \dots, (\vec{x}_b, y_b),$$
$$(\vec{x}_{b+1}, y_{b+1}), \dots, (\vec{x}_{2b}, \dots, y_{2b}),$$
$$\vdots$$
$$(\vec{x}_{(M-1)b+1}, y_{(M-1)b+1}), \dots, (\vec{x}_{\underset{\uparrow}{Mb}}, y_n) \Big)$$

b: mini-batch size

$M = \frac{n}{b}$: number of mini batches

Ex: $n = 8, b = 2, M = \frac{8}{2} = 4$

$$D = \Big(\underbrace{(\vec{x}_1, y_1), (\vec{x}_2, y_2)}_{\text{MB 1}}, \underbrace{(\vec{x}_3, y_3), (x_4, y_4)}_{\text{MB 2}}, \underbrace{(\vec{x}_5, y_5), (\vec{x}_6, y_6)}_{\text{MB 3}}, \underbrace{(\vec{x}_7, y_7), (\vec{x}_8, y_8)}_{\text{MB 4}} \Big)$$

$\frac{n}{b}$ is not always an integer

$M = \text{floor}\left(\frac{n}{b}\right)$ → round down to nearest int

We don't use datapoints at Mb

so we don't use $n - Mb \leq b$ datapoints

$$\hat{L}_m(\vec{w}) = \frac{1}{b} \sum_{i=(m-1)b+1}^{mb} (\vec{x}_i^T \vec{w} - y_i)^2 \qquad m \in \{1, \ldots, M\}$$

$\hat{L}_m(\vec{w})$ is a sample mean estimate of $L(\vec{w})$ with b data points

$\hat{L}_m(\vec{w})$ is a worse estimate than $\hat{L}(\vec{w})$

Algorithm: MBGD Linear Regression Learner (with a constant size)

---

input: $D = ((\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n))$, η, T, b

$\vec{w} \leftarrow$ random vector in $\mathbb{R}^{d+1}$

$M \leftarrow \text{floor}\left(\frac{n}{b}\right)$

for $t = 1, \ldots, T$

Randomly shuffle D

for $m = 1, \ldots, M$

$$\nabla \hat{L}(\vec{w}) \leftarrow \frac{2}{b} \sum_{i=(m-1)b+1}^{mb} (\vec{x}_i^T \vec{w} - y_i) \vec{x}_i$$

$$\vec{w} = \vec{w} - \eta \nabla \hat{L}(\vec{w})$$

return $\hat{f}(\vec{x}) = \vec{x}^T \vec{w}$

Advantage is MT is now the number of gradient steps

Setting $b=n \Rightarrow M=1$ gives back BGD

$b=1 \Rightarrow M=n$ gives "Stochastic GD (SGD)"

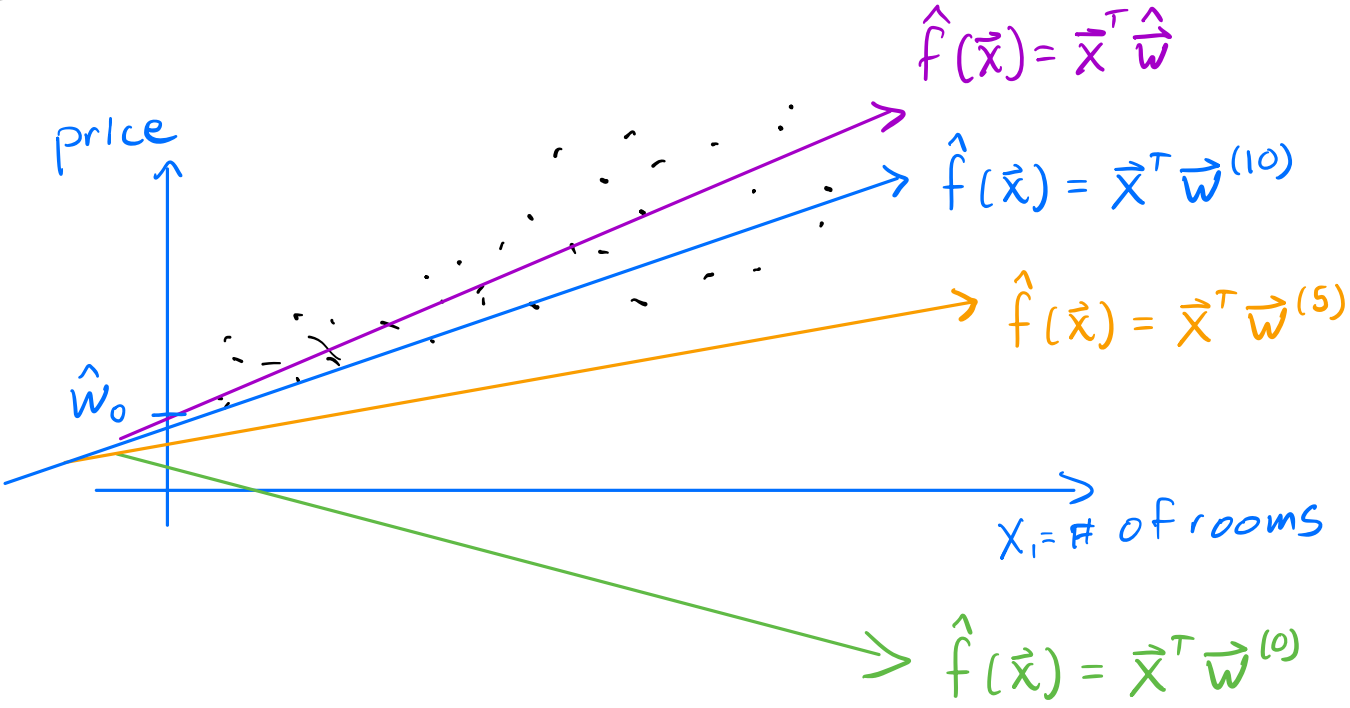
Computation:

MBGD: $O(dbMT)$

BGD: $O(dnT)$

In Practice for same T MBGD usually find a better $\vec{w}$ (case: $b<n$)

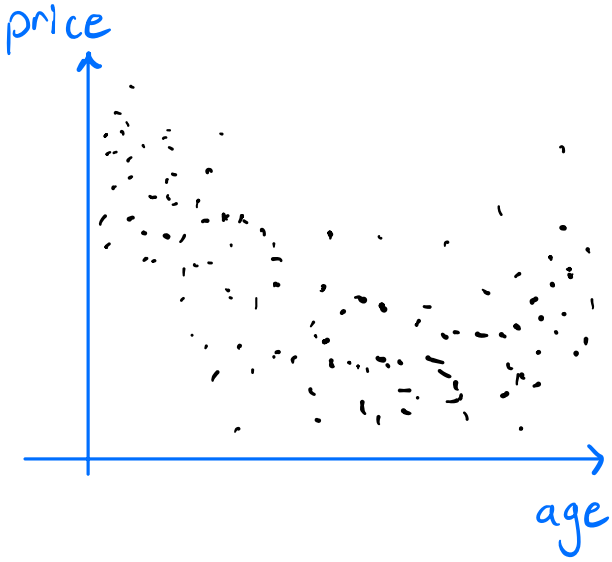
Ex:

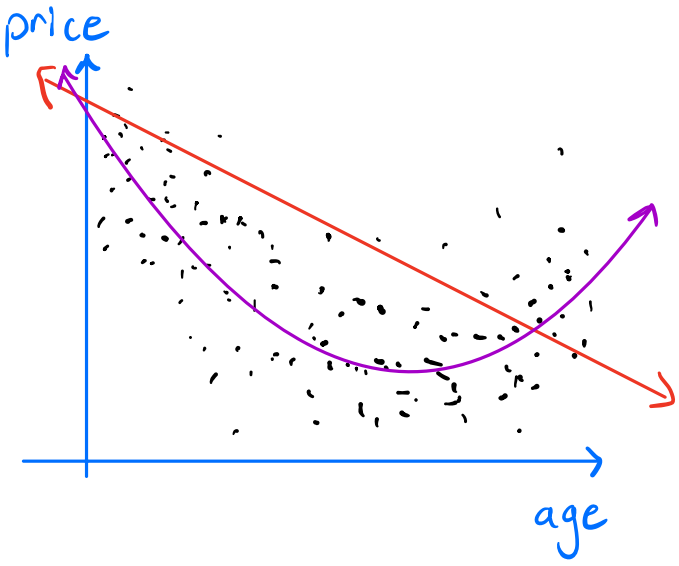


Should we always use a linear function class $\mathcal{F}$?

Polynomial Regression

Ex: Predicting car price based on age of the car





Let $\mathcal{F}$ contain polynomial functions instead of just linear functions

Ex: $d=1$, $\mathcal{X} = \mathbb{R}^{d+1} = \mathbb{R}^2$, $\mathcal{Y} = \mathbb{R}$

linear function: $\vec{x}^T w = (1, x_1)(w_0, w_1)^T = w_0 + x_1 w_1$ $\quad \vec{w} \in \mathbb{R}^2$

$$\mathcal{F}_1 = \{ f \mid f : \mathbb{R}^2 \to \mathbb{R} \text{ and } f(\vec{x}) = \vec{x}^T \vec{w} = w_0 + x_1 w_1, \ \vec{w} \in \mathbb{R}^2 \}$$

degree 2 polynomial: $w_0 + x_1 w_1 + x_1^2 w_2$ $\quad \vec{w} \in \mathbb{R}^3$

$= \phi_2(\vec{x})^T \vec{w}$ $\quad$ linear in $\phi_2(\vec{x})$

"feature map"

where $\phi_2(\vec{x}) = (x_0 = 1, x_1, x_1^2)^T \in \mathbb{R}^3$

$$\mathcal{F}_2 = \{ f \mid f : \mathbb{R}^2 \to \mathbb{R} \text{ and } f(\vec{x}) = \phi_2(\vec{x})^T \vec{w}, \ \vec{w} \in \mathbb{R}^3 \}, \quad \mathcal{F}_1 \subset \mathcal{F}_2$$

degree 3 polynomial: $w_0 + x_1 w_1 + x_1^2 w_2 + x_1^3 w_3 \qquad \vec{w} \in \mathbb{R}^4$

$= \phi_3(\vec{x})^T \vec{w}$ linear in $\phi_3(\vec{x})$

where $\phi_3(\vec{x}) = (1, x_1, x_1^2, x_1^3)^T \in \mathbb{R}^4$

$\mathcal{F}_3 = \{f \mid f: \mathbb{R}^2 \to \mathbb{R} \text{ and } f(\vec{x}) = \phi_3(\vec{x})^T \vec{w}, \vec{w} \in \mathbb{R}^4\}$

$\mathcal{F}_1 \subset \mathcal{F}_2 \subset \mathcal{F}_3$

Ex: $d=2$, $\mathcal{X} = \mathbb{R}^{d+1} = \mathbb{R}^3$, $\mathcal{Y} = \mathbb{R}$

linear function: $\vec{x}^T w = (\overset{x_0}{1}, x_1, x_2)(w_0, w_1, w_2)^T$

$= w_0 + x_1 w_1 + x_2 w_2 \qquad \vec{w} \in \mathbb{R}^3$

$\mathcal{F}_1 = \{f \mid f: \mathbb{R}^3 \to \mathbb{R} \text{ and } f(\vec{x}) = \vec{x}^T \vec{w}, \vec{w} \in \mathbb{R}^3\}$

degree 2 polynomial: $w_0 + x_1 w_1 + x_2 w_2 + x_1^2 w_3 + x_2^2 w_4 + x_1 x_2 w_5$

$= \phi_2(\vec{x})^T \vec{w}, \quad \vec{w} \in \mathbb{R}^6$ linear in $\phi_2(\vec{x})$

where $\phi_2(\vec{x}) = (x_0 = 1, x_1, x_2, x_1^2, x_2^2, x_1 x_2)^T \in \mathbb{R}^6$

$\mathcal{F}_2 = \{f \mid f: \mathbb{R}^3 \to \mathbb{R} \text{ and } f(\vec{x}) = \phi_2(\vec{x})^T \vec{w}, \vec{w} \in \mathbb{R}^6\}$

$\mathcal{F}_1 \subset \mathcal{F}_2$

In general $\phi_p : \mathcal{X} \to \mathcal{Z}$ is a degree p polynomial "feature map"

For $\mathcal{X} = \mathbb{R}^{d+1}$, $\mathcal{Z} = \mathbb{R}^{\bar{p}}$ where $\bar{p} = \binom{(d+1)+p-1}{p} = \binom{d+p}{p}$

Ex: $d=2, p=2 \Rightarrow \bar{p} = \binom{2+2}{2} = \binom{4}{2} = \frac{4 \cdot 3}{2 \cdot 1} = 6$

Ex: $d=2, p=3 \Rightarrow \bar{p} = \binom{2+3}{2} = \binom{5}{2} = \frac{5 \cdot 4}{2} = 10$

The function class becomes:

$$\tilde{F}_p = \{ f \mid f : \mathbb{R}^{d+1} \to \mathbb{R} \text{ and } f(\vec{x}) = \phi_p(\vec{x})^T \vec{w}, \vec{w} \in \mathbb{R}^{\bar{p}} \}$$

$$\tilde{F}_1 \subset \tilde{F}_2 \subset \dots \subset \tilde{F}_p$$

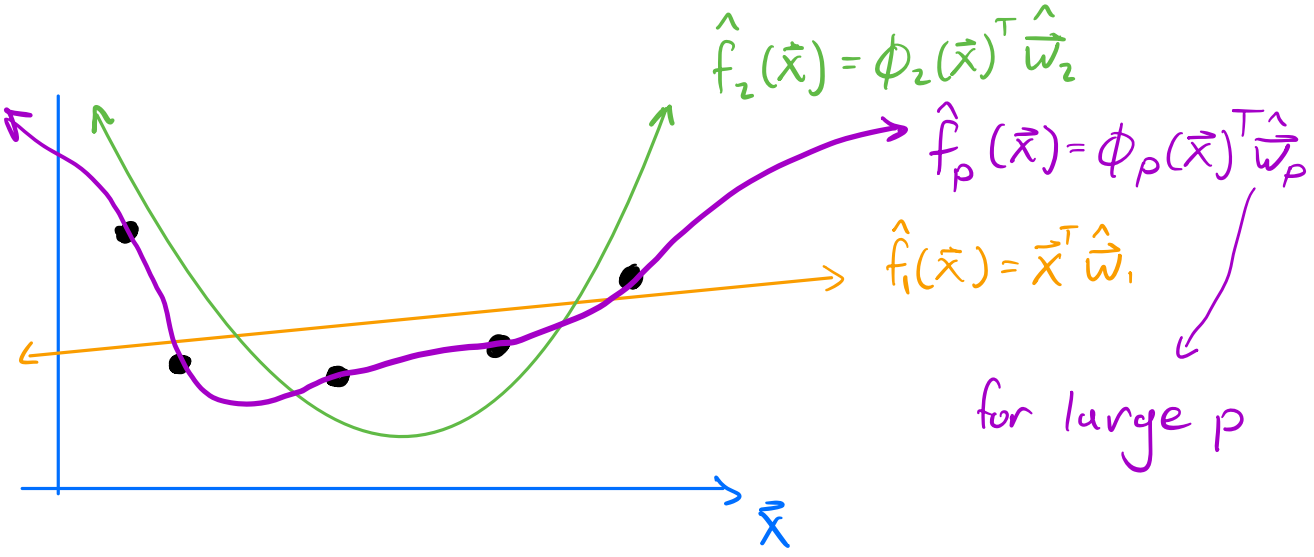
You can then solve for $\hat{\vec{w}}_p = \underset{\vec{w} \in \mathbb{R}^{\bar{p}}}{\arg\min} \hat{L}_p(\vec{w})$

where $\hat{L}_p(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} \ell(\phi_p(\vec{x}_i)^T \vec{w}, y_i) = \hat{L}(\hat{f}_p)$

using the closed form solution $= \phi_p(\vec{x}_i)^T \vec{w}$

or gradient descent as before

Ex:



$$\hat{L}(\hat{f}_1) \geq \hat{L}(\hat{f}_2) \geq \cdots \geq \hat{L}(\hat{f}_p) \approx 0$$

Why not set p as large as possible?