Lecture #3 - Gradient Descent

Readings: Murphy 8-8.2.1
MIT Notes

> Standard ML paradigm is:
> Define Loss, then Optimize:

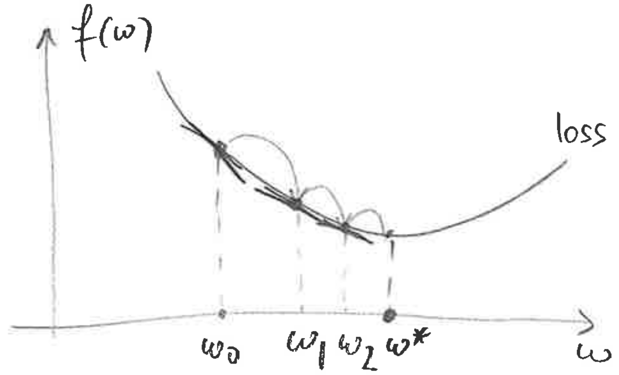
Example: $\boxed{\hat{\omega}_{LS} = \arg\min_{\omega} \| y - X\omega \|_2^2}$

And we had derived: $\boxed{\hat{\omega}_{LS} = (X^T \cdot X)^{-1} \cdot X^T \cdot y}$ as a closed form solution.

> But, for most losses used in practice, there is no closed-form solution!

> Key idea: iterative methods (very popular)

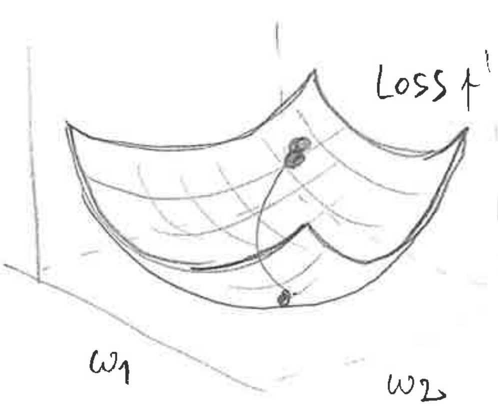
Gradient Descent in 1 dimension:



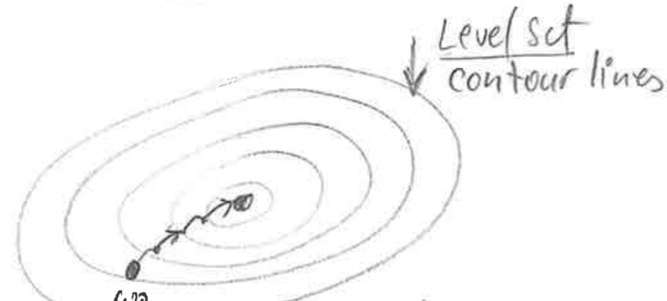
Step Direction: - gradient
Step Size: - proportional to |gradient|

use $\boxed{\eta} * |\text{gradient}|$

Gradient Descent in multiple dimensions:



→



Recall: For a function $f: \mathbb{R}^d \to \mathbb{R}$, a Level Set at value $\boxed{c}$ is the set of all points in the domain where f takes the same value

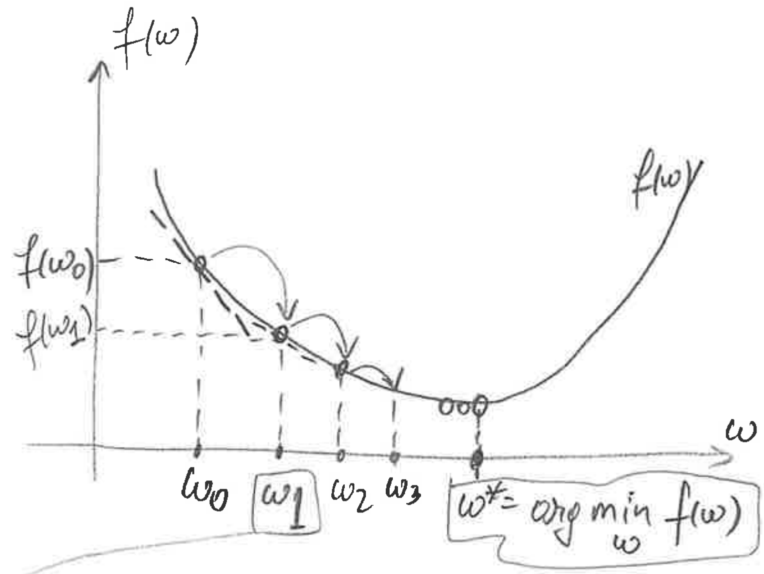
Algorithm – Gradient Descent (GD)

for $t = 0, 1, 2, \dots$

$$\boxed{w_{t+1} = w_t - \eta \left.\frac{d f(w)}{dw}\right|_{w=w_t}}$$

Hyperparameters:

- initial point $\boxed{w_0}$
- step size $\boxed{\eta}$



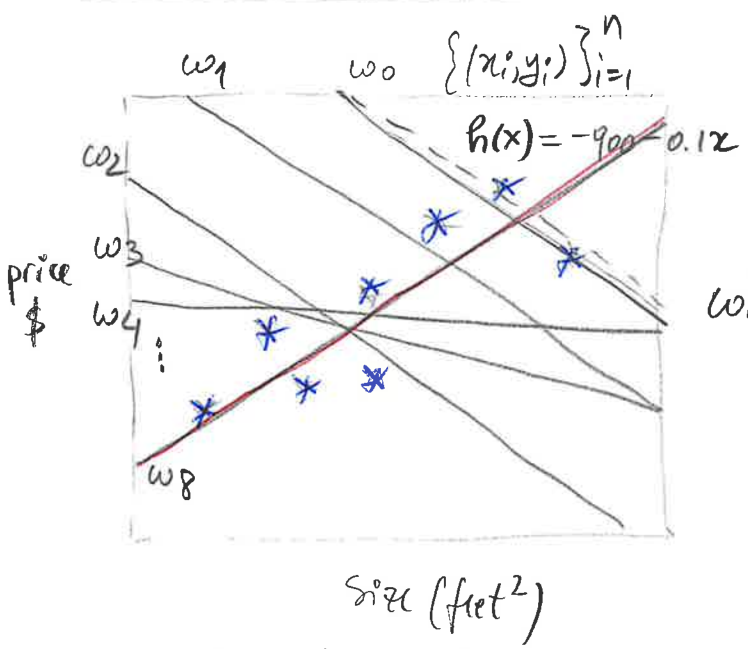
$$w_1 = w_0 - \eta \left.\frac{d f(w)}{dw}\right|_{w=w_0}$$

$$w_2 = w_1 - \eta \left.\frac{d f(w)}{dw}\right|_{w=w_1}$$

...

$\boxed{\text{Note}}$: As $t \to \infty$, $\left.\frac{d f(w)}{dw}\right|_{w=w_t} \to 0$

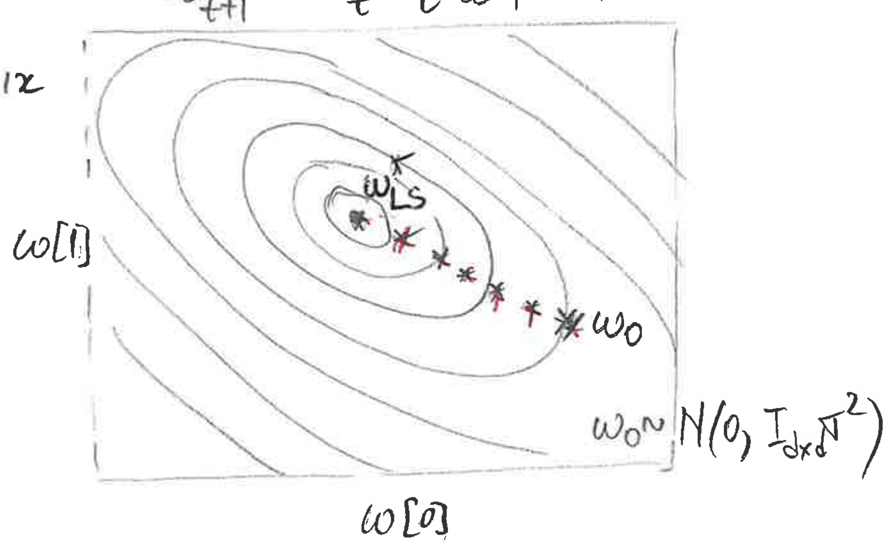
1-dimensional Linear Regression with 2 parameters



Evolution of predictor

$y = w[0] + w[1]x$

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



Gradient Descent dynamics in parameter space w. Ovals show "level set" of the objective function.

EXAMPLES:

1 Quadratic Functions

$$\hat{\omega} = \arg\min_{\omega} \underbrace{a\omega^2 + b\omega + c}_{f(\omega) \quad (\text{loss})}$$

Use:

$$\omega_{t+1} = \omega_t - \eta \cdot \left.\frac{d f(\omega)}{d\omega}\right|_{\omega=\omega_t}$$

$$\omega_0 \sim \mathcal{N}(0, I\sigma^2)$$

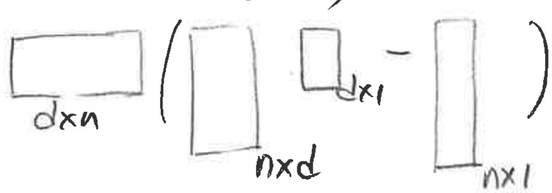
$$\left.\frac{d f(\omega)}{d\omega}\right|_{\omega=\omega_0} = 2a\omega_0 + b$$

$$\boxed{\omega_1 = \omega_0 - \eta(2a\omega_0 + b)}$$

2 Linear Regression

$$\hat{\omega} = \arg\min_{\omega} \underbrace{\frac{1}{2}\sum_i (y_i - x_i^T \omega)^2}_{f(\omega) \quad (\text{loss/cost})}$$

$$\nabla_{\omega} f(\omega_0) = X^T(X\omega_0 - y)$$ ← see lecture 02 notes.



$$\boxed{\omega_{t+1} = \omega_t - \eta \cdot X^T(X\omega_0 - y)}$$

3 Lasso

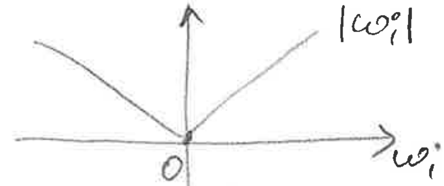
$$\hat{\omega} = \arg\min_{\omega} \underbrace{\frac{1}{2}\|y - X\omega\|_2^2 + \lambda \|\omega\|_1}_{f(\omega)\ (\text{loss})}$$

$\boxed{\|\omega\|_1 \triangleq \sum_{i=1}^{n} |\omega_i|}$ is not differentiable at 0

$$\boxed{\nabla_\omega f = X^T(X\omega - y) + \lambda \sum_{i=1}^{n} \text{sign}(\omega_i)}$$

(see notes of Lecture 02)

$$\boxed{\omega_{t+1} = \omega_t - \eta \cdot \nabla_\omega f|_{\omega=\omega_t}}$$



Derivative for each coordinate:

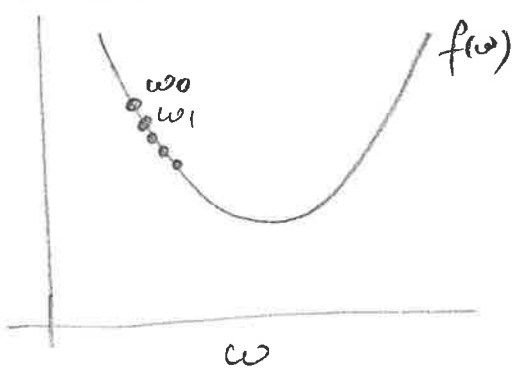
$$\frac{\partial |\omega_i|}{\partial \omega_i} = \begin{cases} +1, & \omega_i > 0 \\ [-1, 1], & \omega_i = 0 \\ -1, & \omega_i < 0 \end{cases}$$

$$f(\omega) = a\omega^2 + b\omega + c + |\omega|$$

$$\frac{df}{d\omega} = 2a\omega + b + \text{sign}(\omega) = 0$$

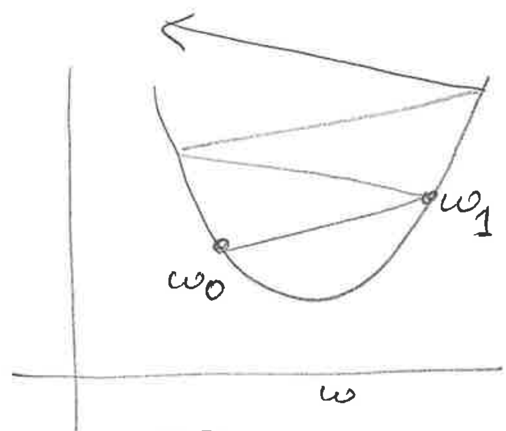
- can find if ω is 1-dimensional
- $\omega \in \mathbb{R}^d$ $\quad$ $\text{sign}(\omega) \to 2^d$ possibilities

How to choose a step size?



Step size $\boxed{\eta}$ is too small

=> slow convergence.

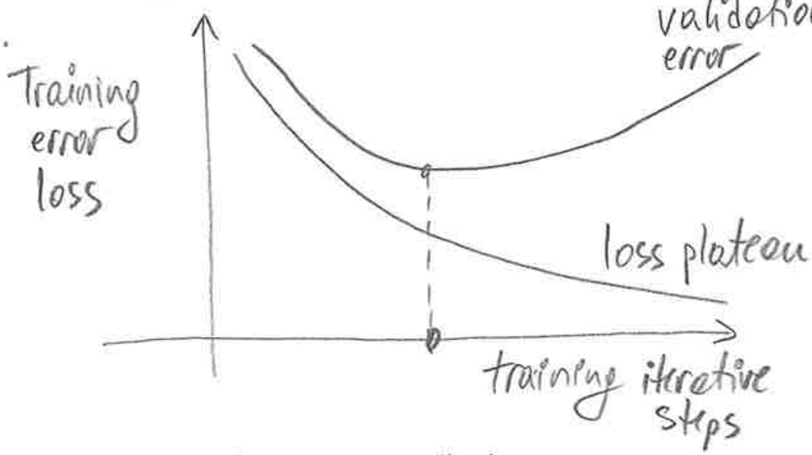


If $\boxed{\eta}$ is too large.

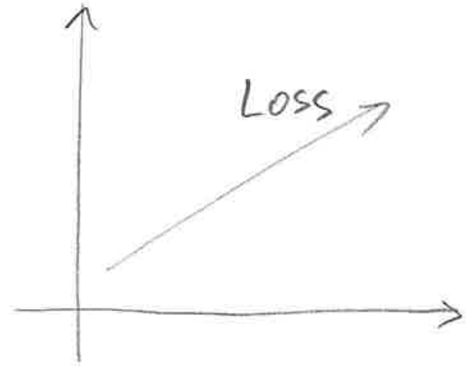
=> Diverging

> In practice: guess and check !?

Visualize loss error over training.



Train loss effectively minimized, but use $\boxed{\text{early stopping}}$ to prevent overfitting.



Loss increasing is a sign that step size $\boxed{\eta}$ might be too large!

Based on Géron's Book: (pp. 145)

- Main problem with Batch Gradient Descent is:
  - uses the whole training set to compute the gradients at every step, which makes it very slow when training set is large!
- At the opposite extreme: Stochastic Gradient Descent – picks a random instance in the training set at every step and computes the gradients based only on that single instance!
- As a compromise: mini-Batch Gradient Descent.

NOTES

- The actual implementations of Batch GD & Stochastic GD algorithms are provided at pp. 143 & 146 in Geron's Book.
- These algorithms are also available in Scikit-Learn as the SGDRegressor class.

- When using GD, you should ensure that all features have similar scale (eg, using Scikit-Learn's StandardScaler class), or else it will take longer to converge! [Geron, pp. 141]