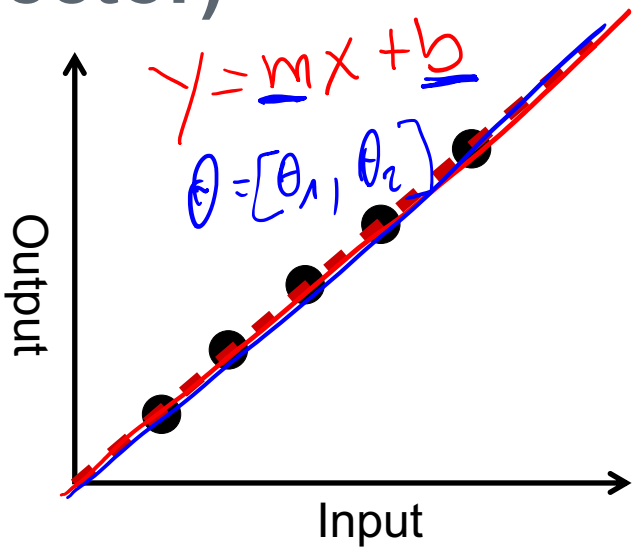
Optimization and Gradient Descent

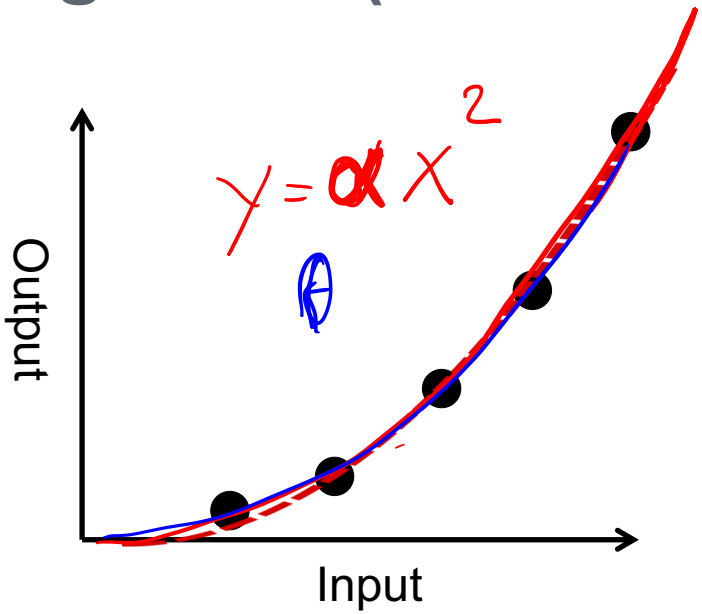
Heni Ben Amor, Ph.D.
Assistant Professor
Arizona State University

Ira A. Fulton Schools of Engineering
Arizona State University

Recap: Regression

- Learn continuous relationship of input to output
- Can be seen as learning a function
- Extract functional relationship
- Maps input vector to single real (or vector)





Optimization for Machine Learning

- Given an ML model with parameters
- Find model parameters that minimize loss

$$\underset{\theta \in \mathbb{R}^D}{\operatorname{argmin}} \mathcal{L}(\theta_t; \mathbf{X}; \mathbf{Y})$$

training dat

input outputs

loss of training data wrt. θ

- In other words: change model parameters until they best fit your data
- This is a numerical optimization problem

Numerical Optimization

- Iterative optimization
- Use in absence of closed-form solutions
- Repeatedly try out multiple solutions until a sufficiently good solution is found.

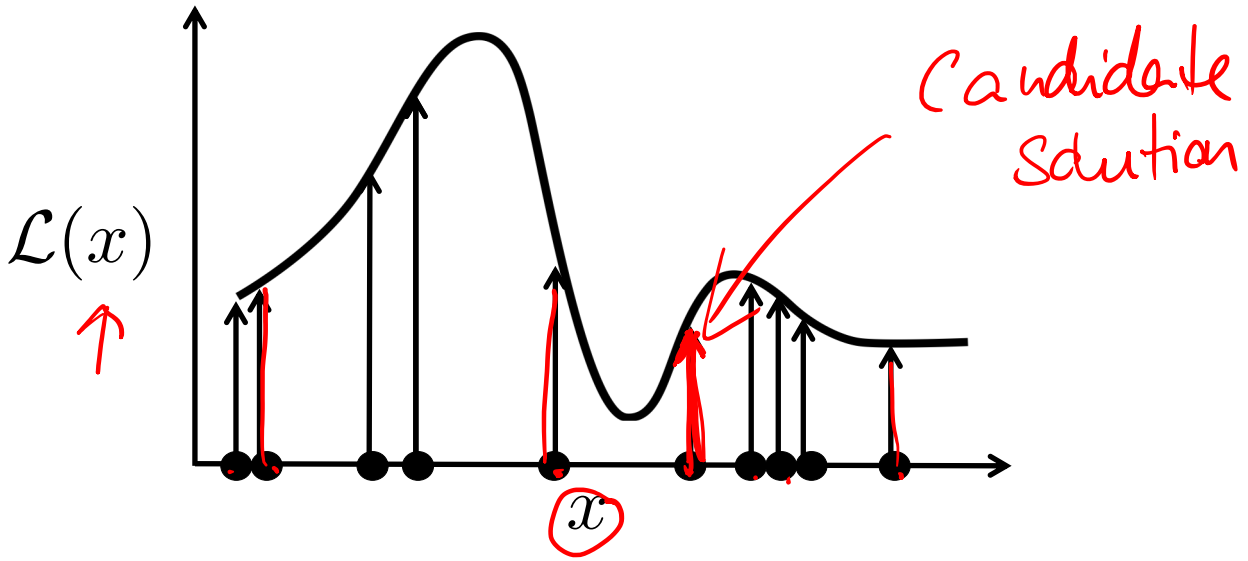
objective func = loss function

$$\underset{\mathbf{x} \in \mathbb{R}^D}{\operatorname{argmin}} \mathcal{L}(\mathbf{x})$$

$$\mathcal{L} : \mathbb{R}^D \rightarrow \mathbb{R}$$

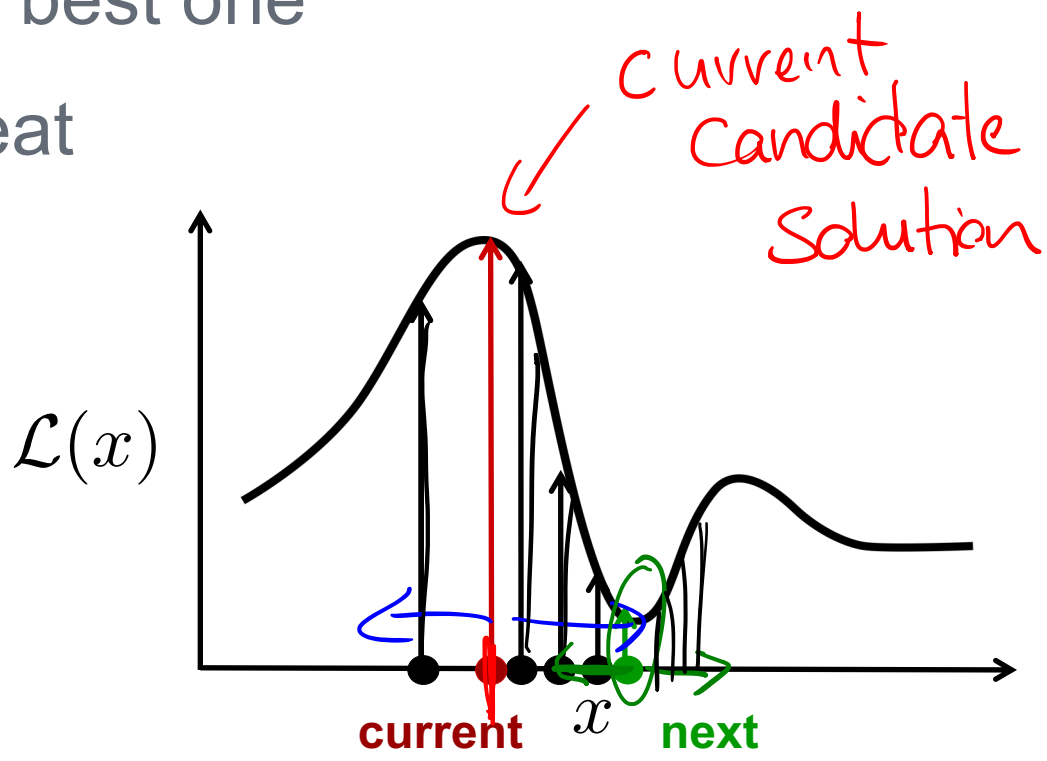
Naive Approach

- Random search in search space
- Sample random $\mathbf{x} \in \mathbb{R}^D$ – dimensionality $x \in \mathbb{R}$
- Evaluate $\mathcal{L}(\mathbf{x})$
- Take the one with the smallest value



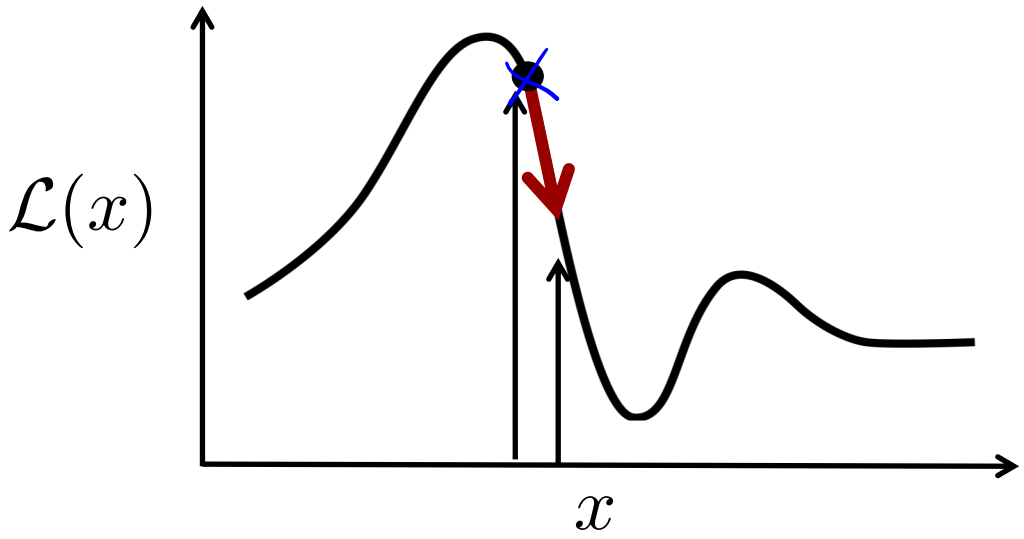
Hill-Climbing

- **A form of stochastic optimization**
  - Sample around current solution
  - Take best one
  - Repeat



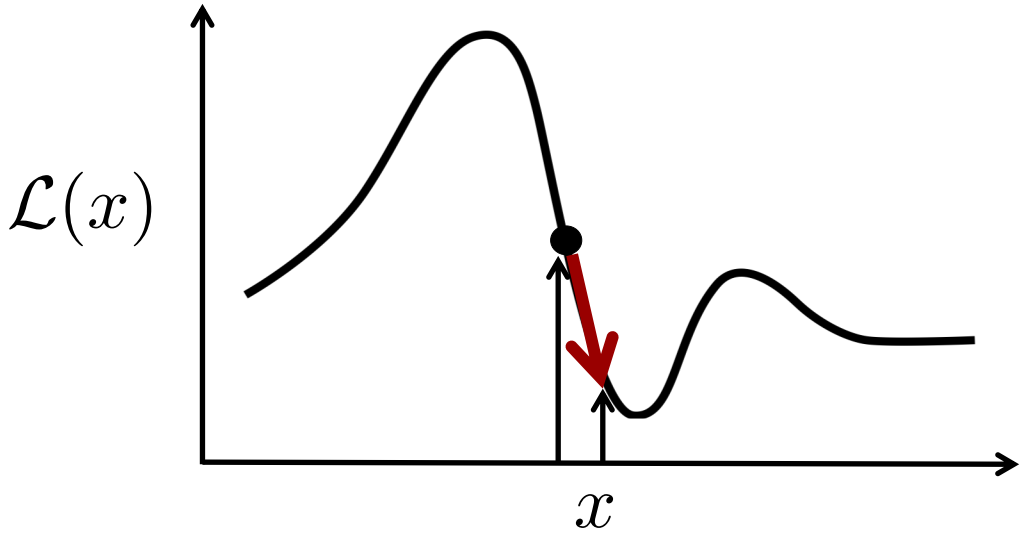
Gradient Descent

- We want to minimize the number of samples
- Make clever sampling steps
- Calculate the direction of the steepest descent
- Using the derivative



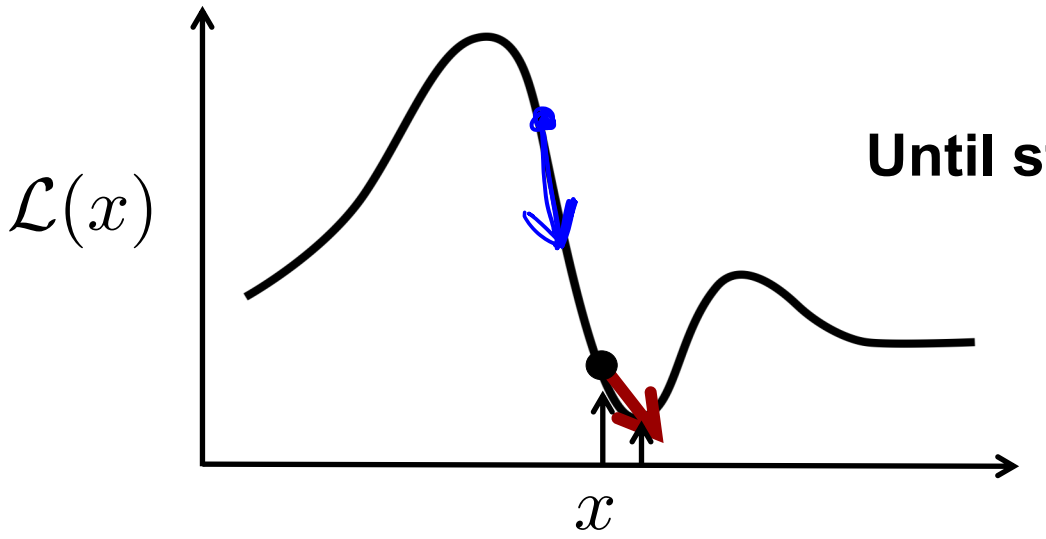
Gradient Descent

- We want to minimize the number of samples
- Make clever sampling steps
- Calculate the direction of the steepest descent
- Using the derivative



Gradient Descent

- We want to minimize the number of samples
- Make clever sampling steps
- Calculate the direction of the steepest descent
- Using the derivative



Until stopping criterion

Formalizing Gradient Descent

- **Gradient descent update rule:**

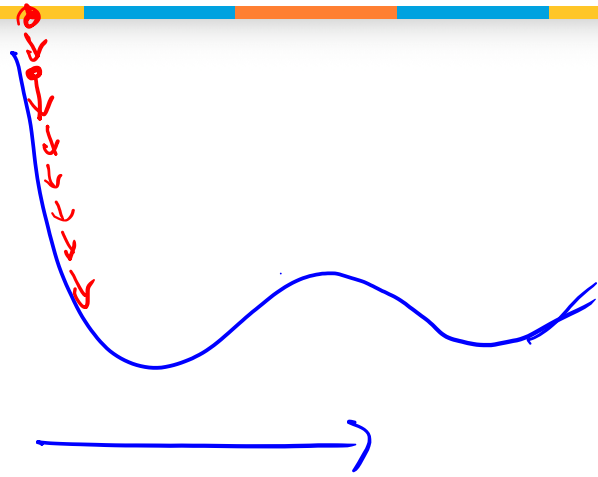
$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla \mathcal{L}(\mathbf{x})$$

- **Gradient:** $\nabla \mathcal{L}(\mathbf{x}) = \left[\frac{\partial}{\partial x_j} \mathcal{L}(\mathbf{x}) \right]^T$

- **Parameter α is called the learning rate**
  - Controls the descent speed

Learning Rate

Update rule, again:



$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla \mathcal{L}(\mathbf{x})$$

Learning rate:

- Too high = overshoot
- Too low = slow convergence

Gradient

Basic Gradient Descent Algorithm

Algorithm

1. Repeat:

$$\mathbf{x} \leftarrow \mathbf{x}_{old} - \alpha \nabla \mathcal{L}(\mathbf{x})$$

gradient

2. Until:

$$\Delta \mathbf{x} < \epsilon$$

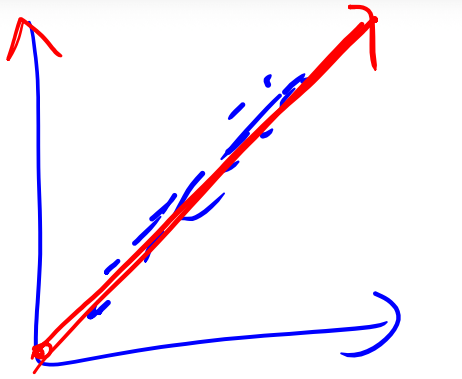
stopping criterion

θ parameters of a ML model

Note: we will replace $\mathbf{x}$ with parameters θ going forward

Linear Regression

- **Fitting line to a set of points**
  - linear regression
- **Line equation:**



$$\hat{y}^{(i)} = \theta_1 x^{(i)} + \theta_2$$

slope

- **Our linear regression defined by parameters:**

$$\theta = [\theta_1, \theta_2]$$

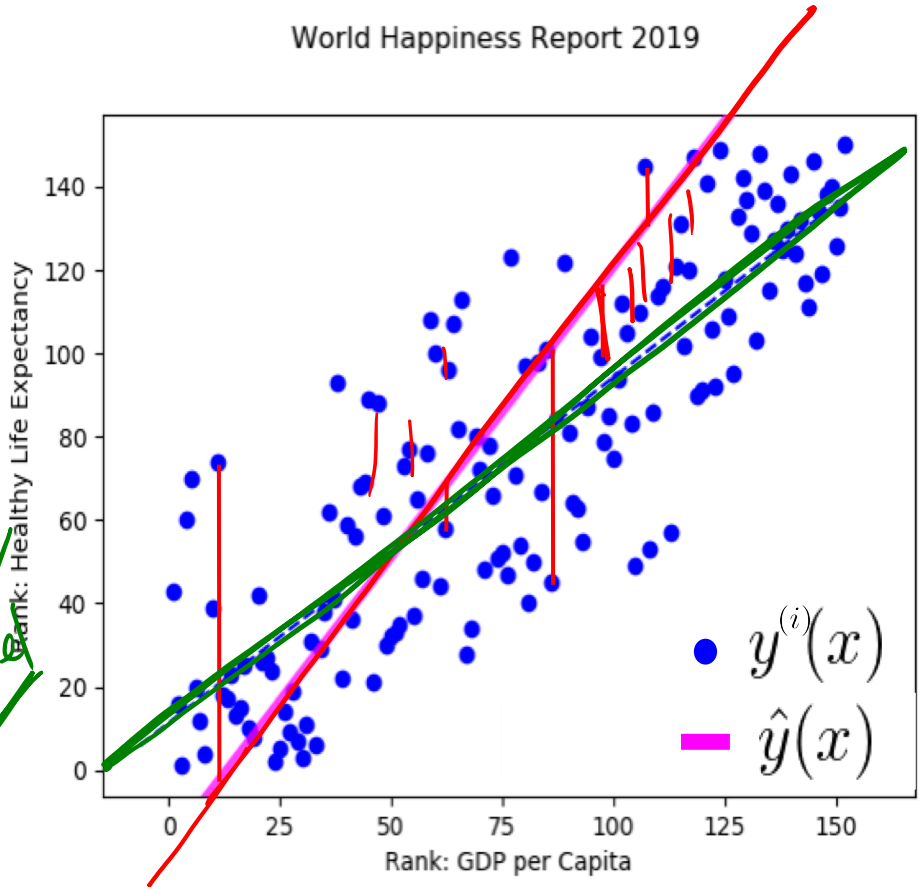
Loss Function

- How do we measure the quality of our linear regression $\hat{y}(x)$?
- A commonly used loss function, **Mean Squared Error (MSE):**

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

true label/output

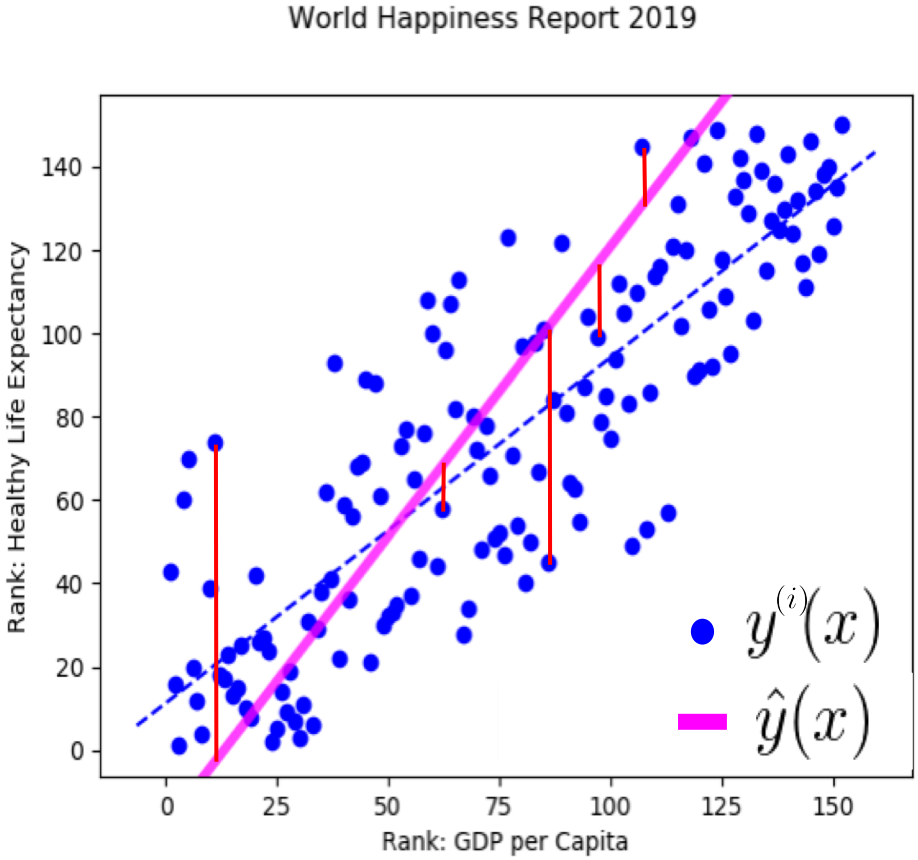
output by ML model



Loss Function

- How to improve our regression?
- Perform gradient descent over the loss function with respect to the regression's parameters

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$



Loss Function of Parameters

- If we define our loss as a function of our parameters, we can optimize our linear regression by moving down the gradient.

loss

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2$$

linear model

$$\hat{y}(x) = \theta_1 x + \theta_2$$

$$\mathcal{L}(\theta_1, \theta_2; X, Y) = \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - (\theta_1 x^{(i)} + \theta_2))^2$$

$x^{(i)}$: inputs; $y^{(i)}$: outputs; label

X, Y: training set

$y - \hat{y}$

Gradient of Loss Function

$$\nabla \mathcal{L} = [\frac{\partial \mathcal{L}}{\partial \theta_1}, \frac{\partial \mathcal{L}}{\partial \theta_2}]$$

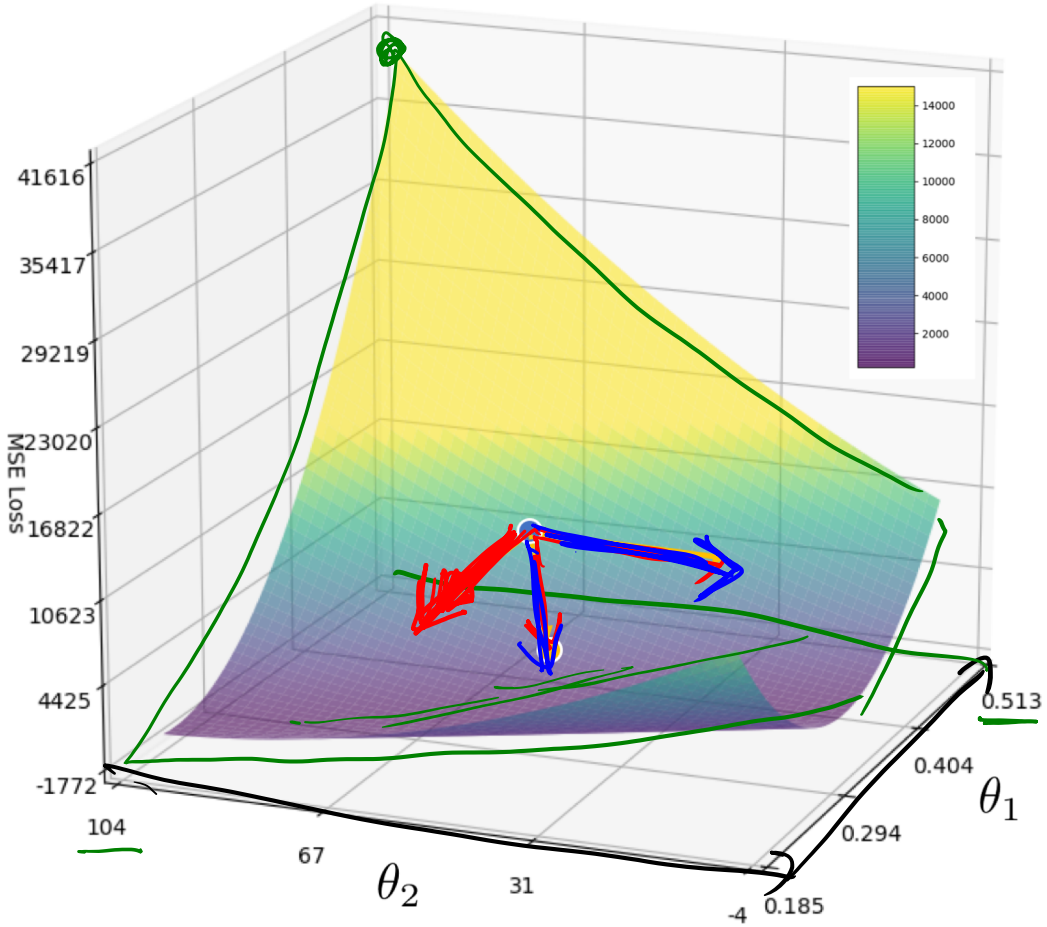
- For our basic linear function:

$$\frac{\partial \mathcal{L}}{\partial \theta_1} = 2x^2\theta_1 + 2x\theta_2 - 2xy$$

$$\frac{\partial \mathcal{L}}{\partial \theta_2} = 2\theta_2 + 2x\theta_1 - 2y$$

Gradient Descent Step

- A step is made for each parameter opposite the gradient of the loss function with respect to the parameter.



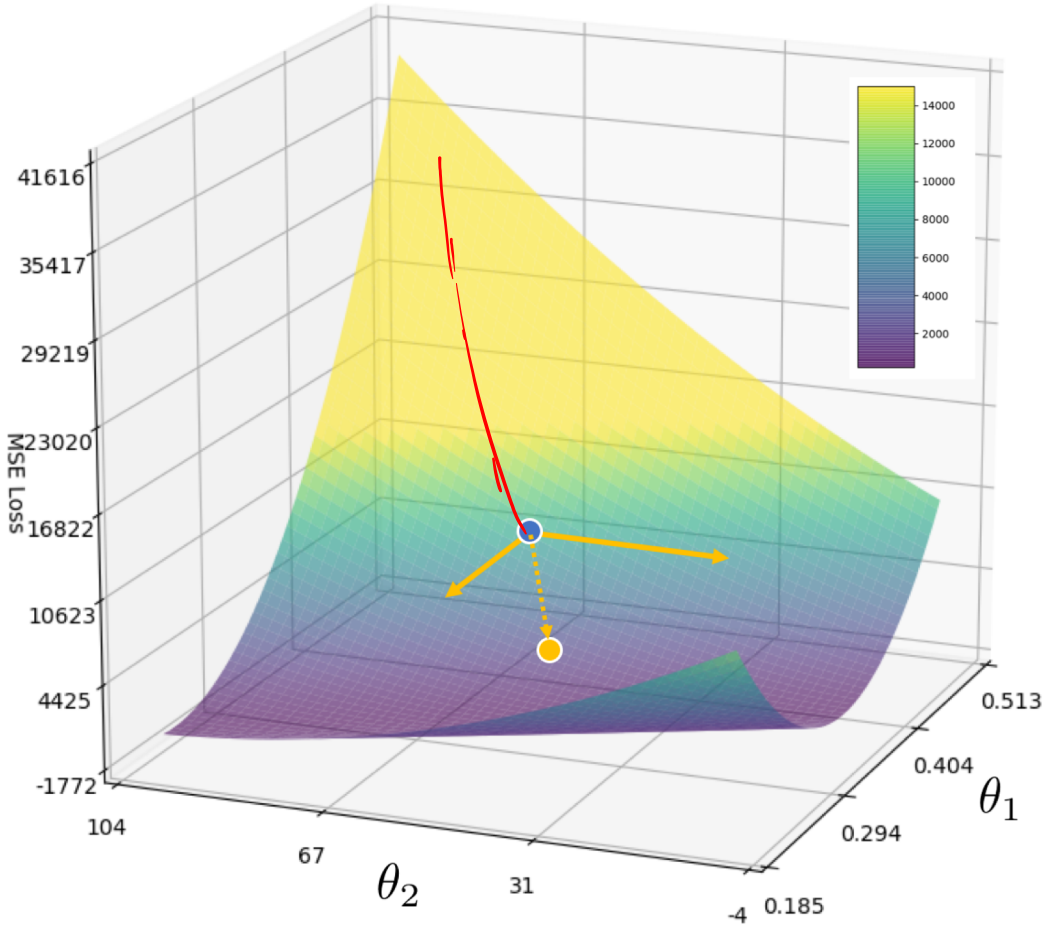
$$-\frac{\partial \mathcal{L}}{\partial \theta_1} \qquad -\frac{\partial \mathcal{L}}{\partial \theta_2}$$

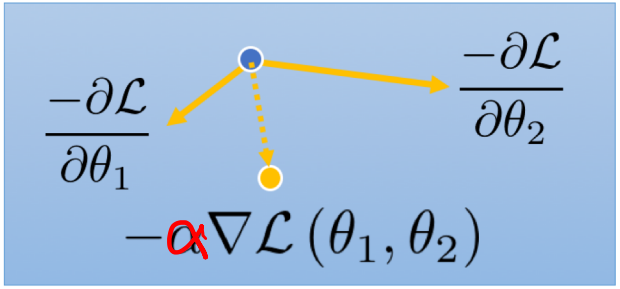
$$-\alpha \nabla \mathcal{L}(\theta_1, \theta_2)$$

direction of steepest desc.

Gradient Descent Step

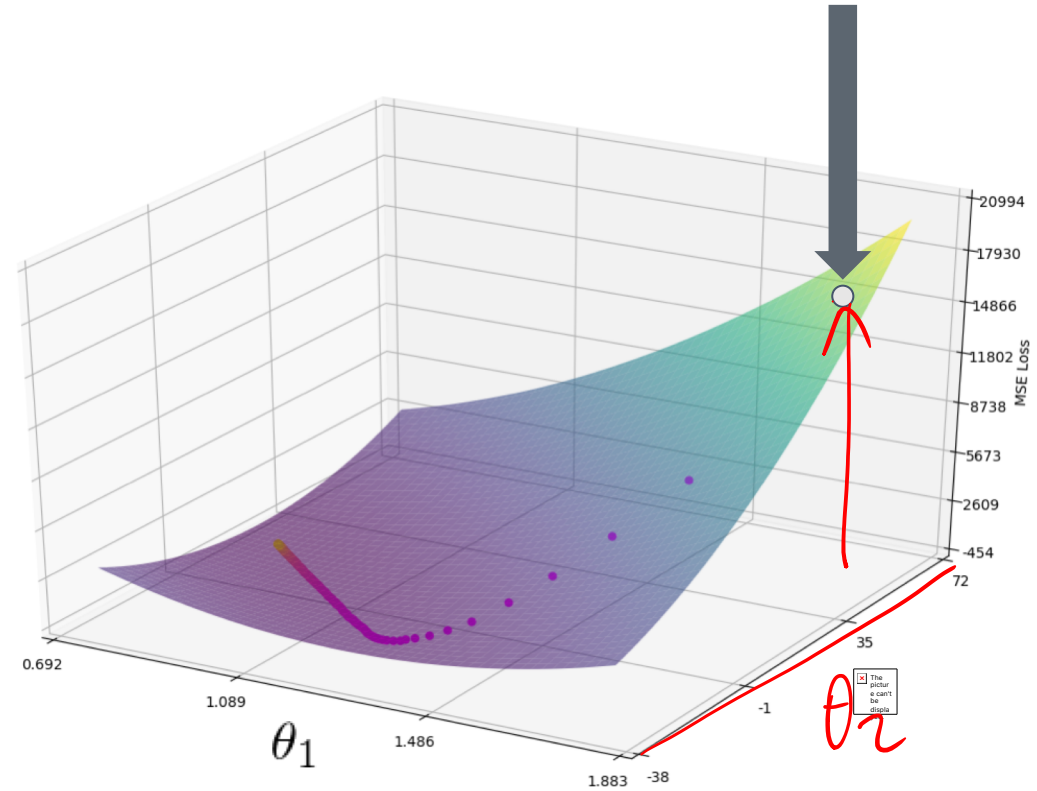
- The size of the step is scaled by the learning rate α .

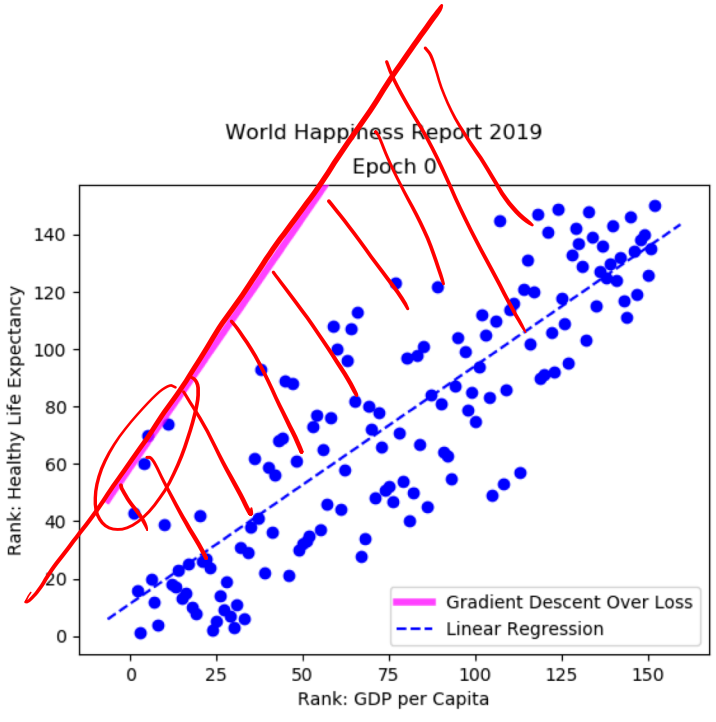




Gradient Descent Process

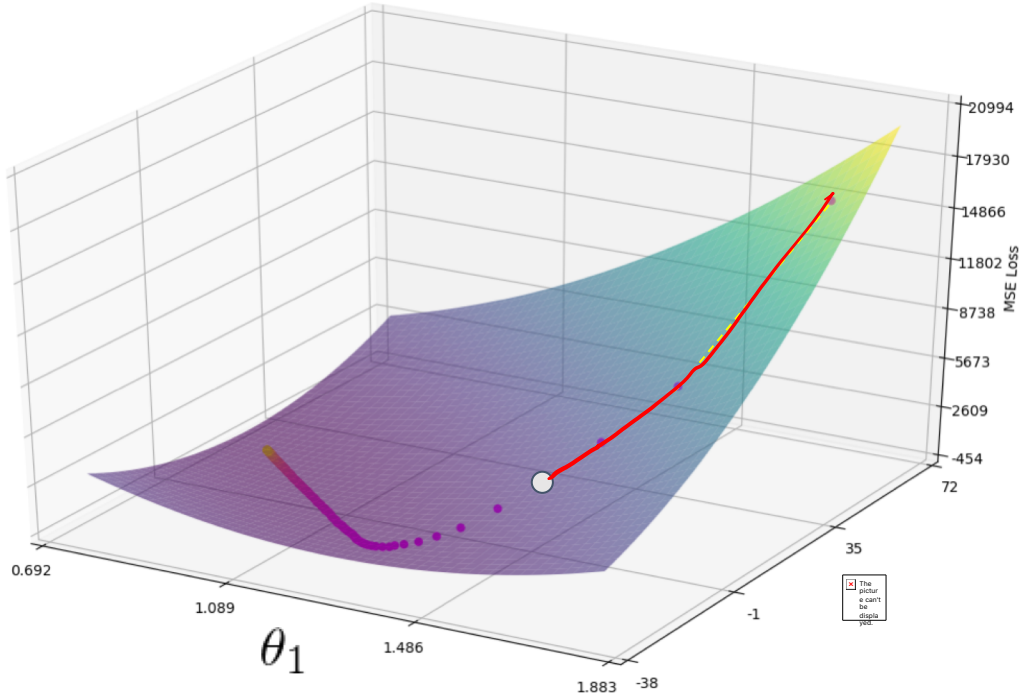
Randomly Initialized Starting Point

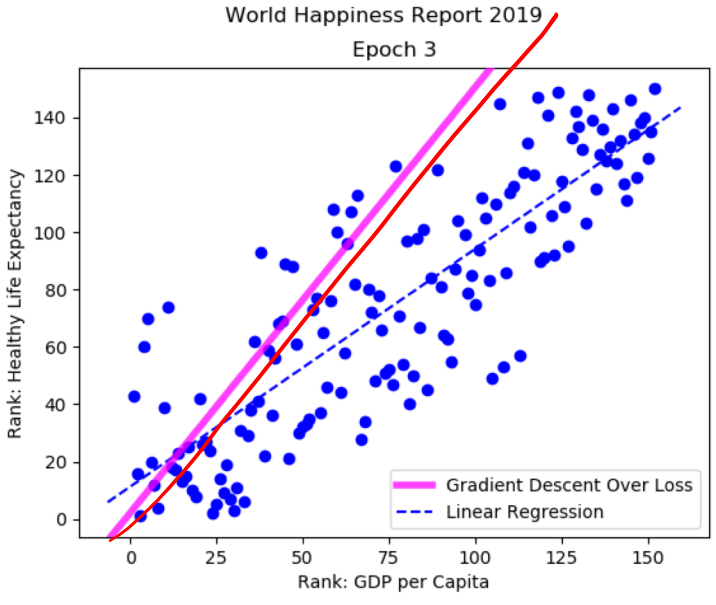




Gradient Descent Process

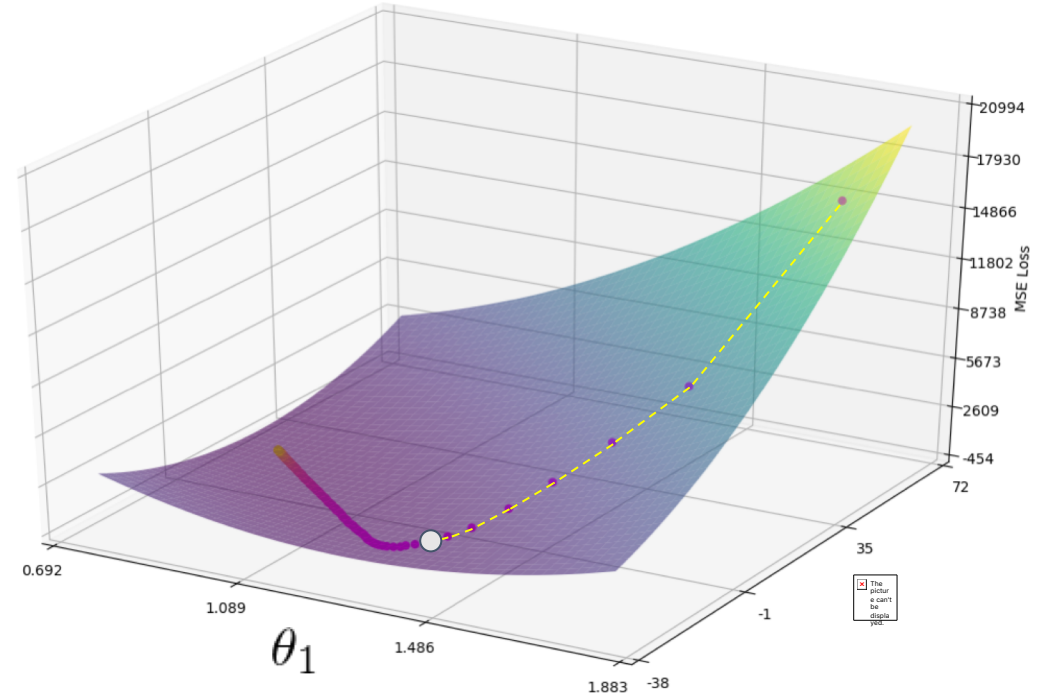
- Typically steeper gradients to start result in larger step sizes

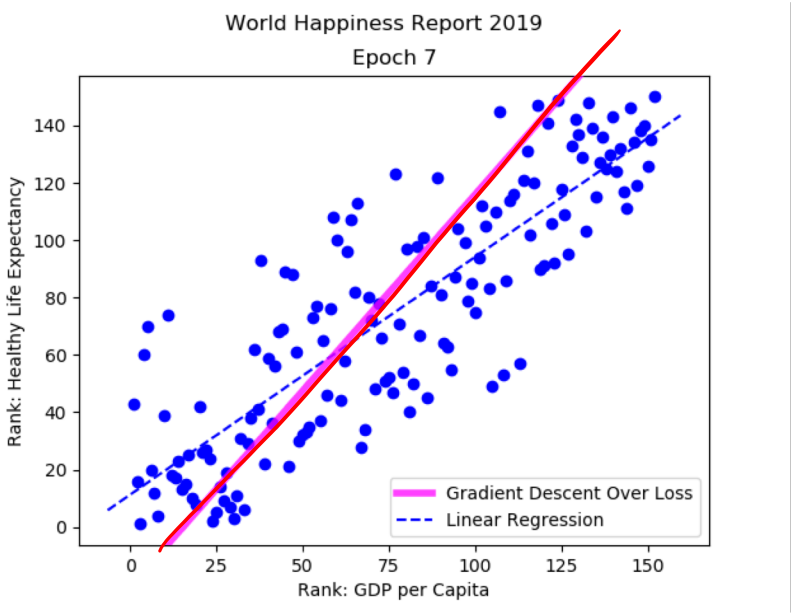




Gradient Descent Process

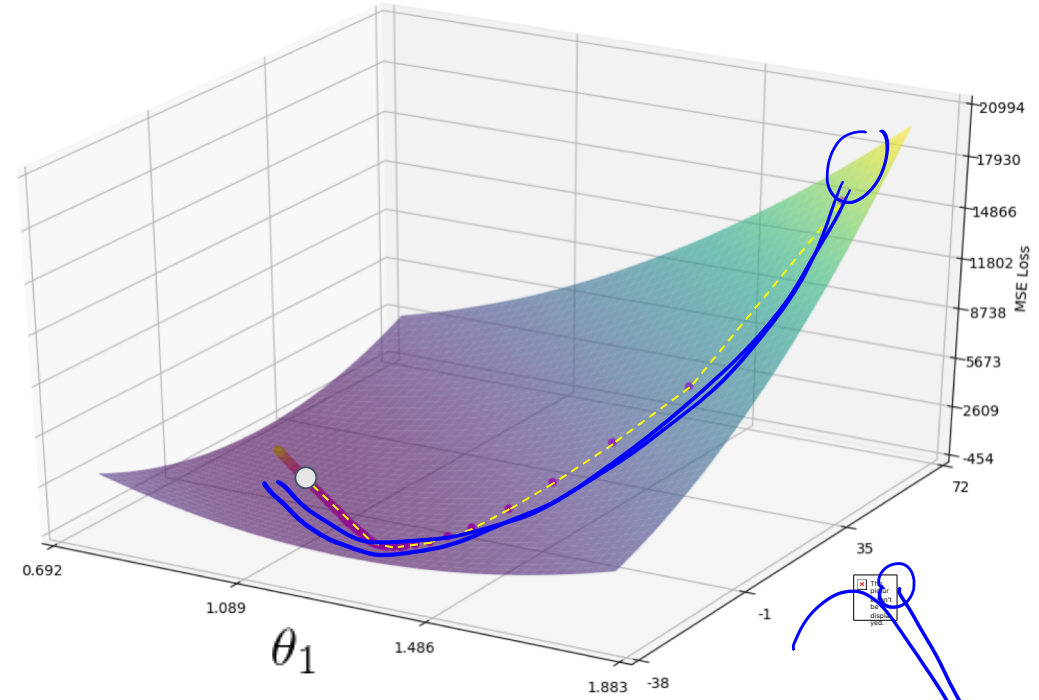
- As the gradient becomes less steep, the step size naturally gets smaller.

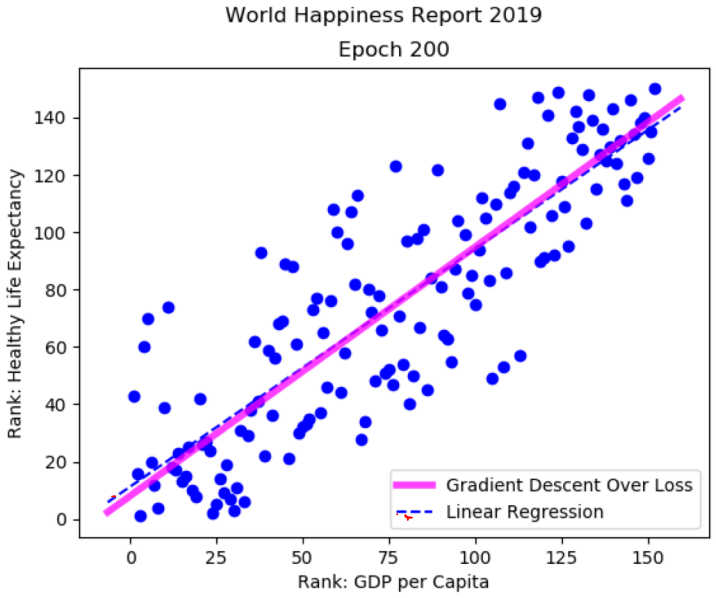




Gradient Descent Process

- As the gradient becomes less steep, the step size naturally gets smaller.

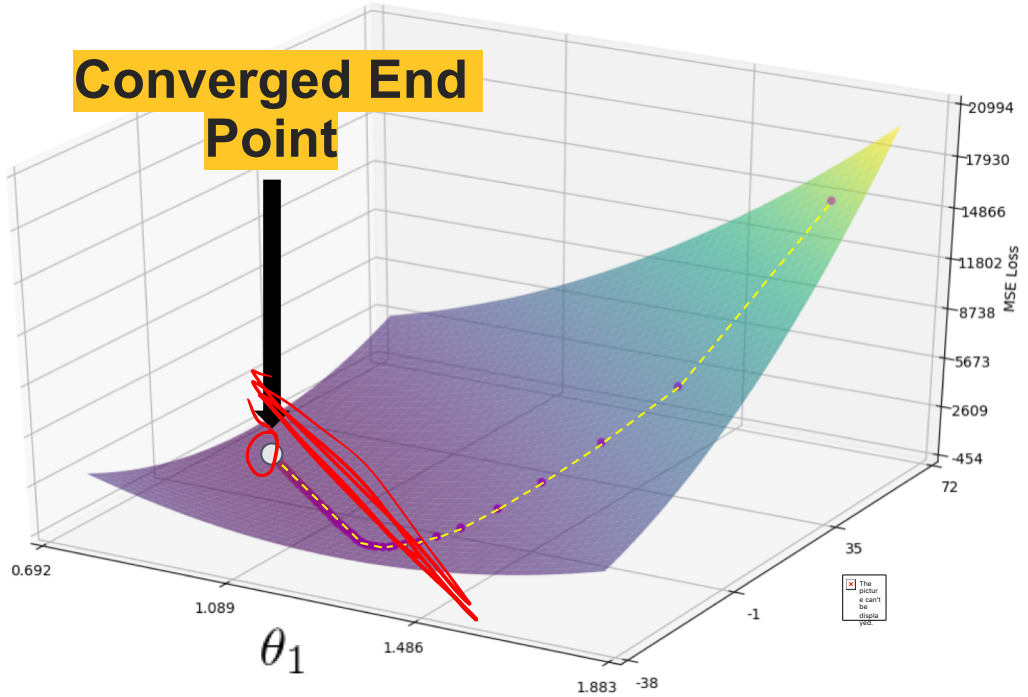


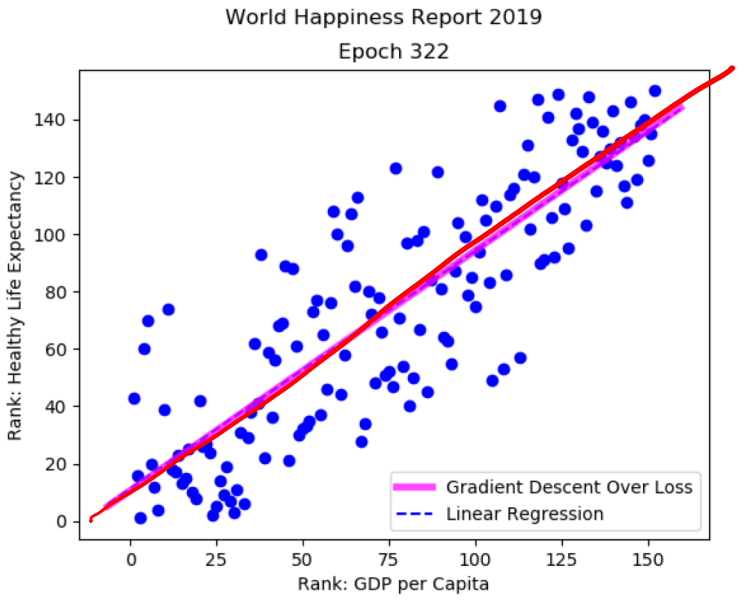


Gradient Descent Process

- Gradient descent will continue to make steps until it has settled around a local minima.

minimum





- Depending on many factors, *such as initialization*, this **may** or **may not** be a good solution.

Summary

- Optimization for machine learning
- Random search
- Hill climbing
- Gradient descent
- Linear regression
- Mean square error loss function