CS167: Machine Learning

Optimization for Weight Learning
Gradient Descent
Stochastic Gradient Descent (SGD)

Tuesday, April 2nd, 2024

Drake University

Recap: Learning Weight Parameters with Modified Neuron Model

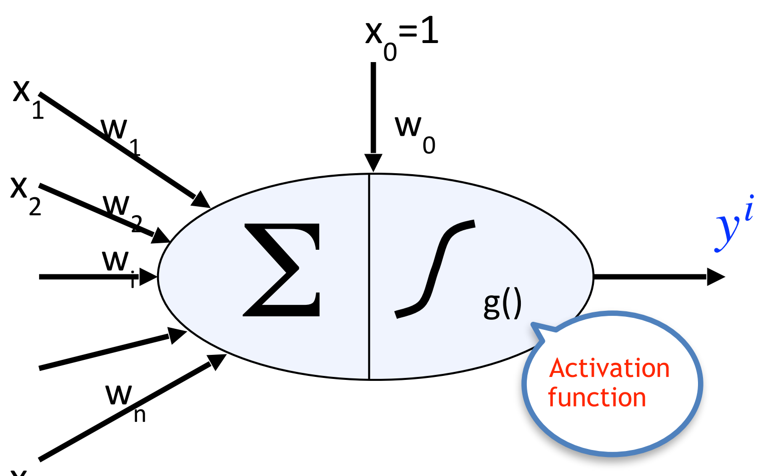
boston-housing dataset training features

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00
1.34284	0.0	19.58	0	0.605	6.066	100.0	1.7573	5	403	14.7	6.43
4.81213	0.0	18.10	0	0.713	6.701	90.0	2.5975	24	666	20.2	16.42

training labels

MEDV
21.7
24.3
16.4

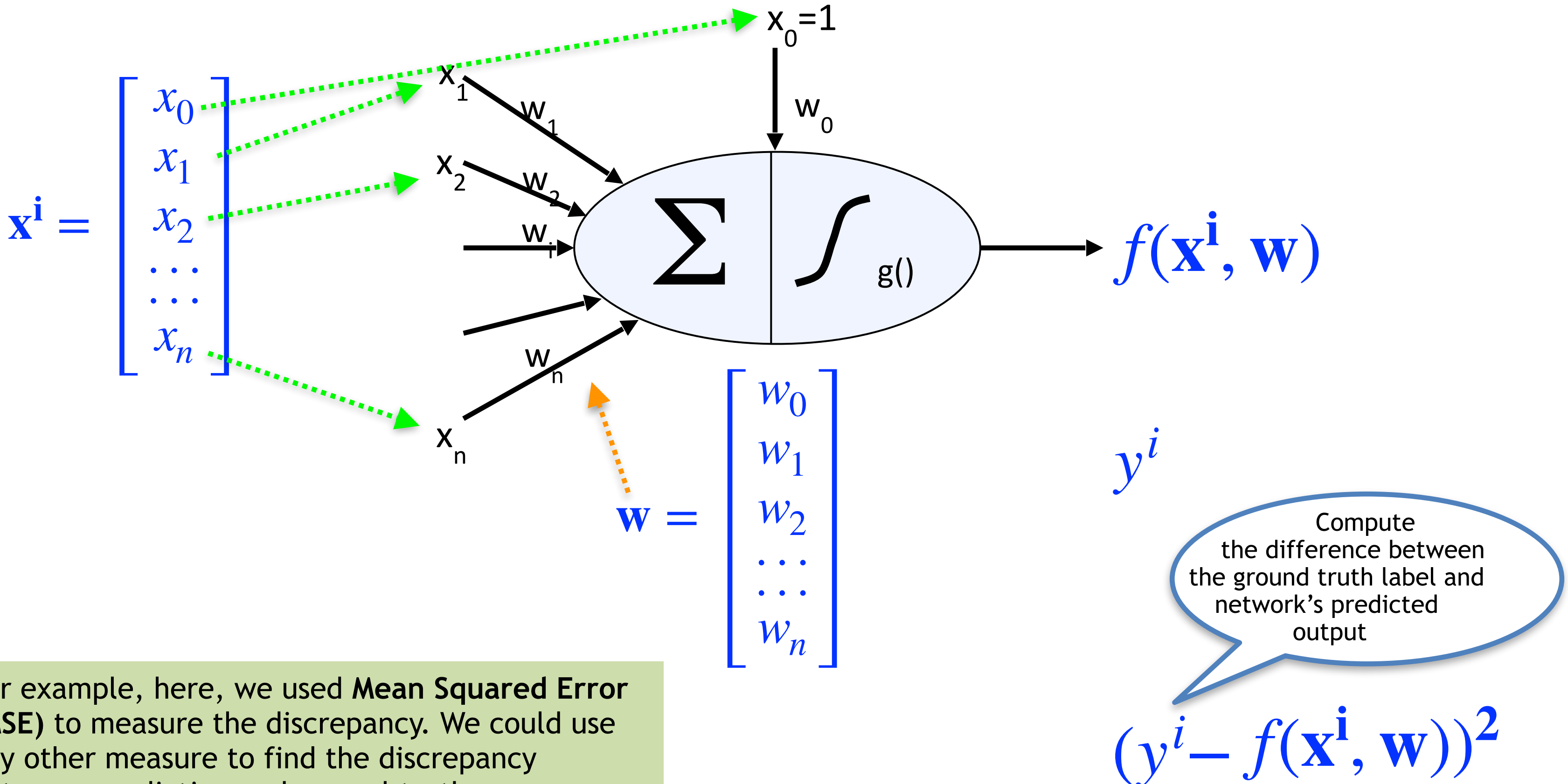
$$\mathbf{x^i} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ \cdots \\ x_n \end{bmatrix}$$



Neuron Model

Recap: Learning Weight Parameters with Modified Neuron Model

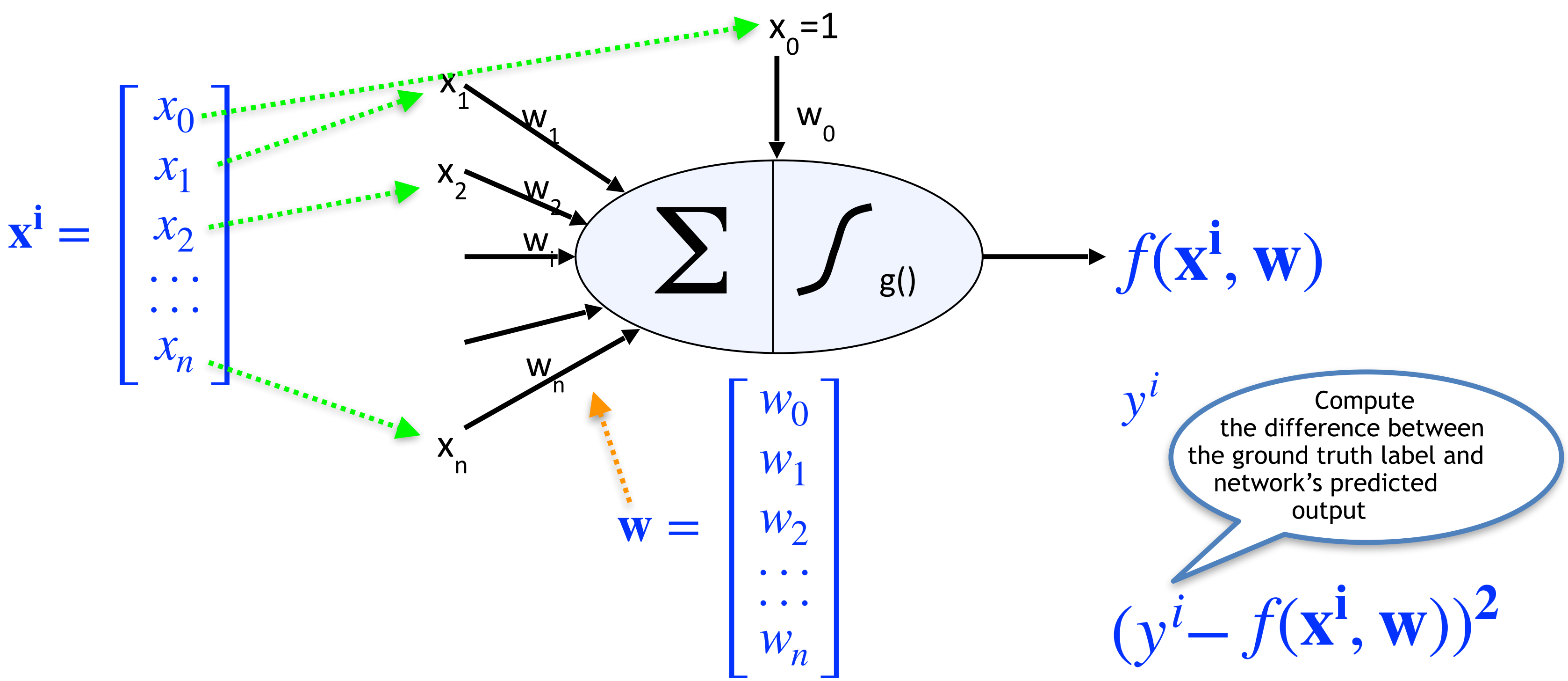
- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



For example, here, we used **Mean Squared Error (MSE)** to measure the discrepancy. We could use any other measure to find the discrepancy between prediction and ground-truth

Recap: Learning Weight Parameters with Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input



$$Error(\mathbf{x}^{\mathbf{i}}, y^i, \mathbf{w}) = (y^i - f(\mathbf{x}^{\mathbf{i}}, \mathbf{w}))^2$$

Recap: Learning Weight Parameters with Modified Neuron Model

- Treat the problem as one of *minimization error* between a single training example's label and the network's output, given the example and weights as input

$$Error(\mathbf{x^i}\ , y^i\ , \mathbf{w}) = \ (y^i - f(\mathbf{x^i}, \mathbf{w}))^2$$

- If we consider a collection of training examples and sum the above error over all examples

$$E(\mathbf{w}) = \sum_i \ Error(\mathbf{x^i}\ , y^i\ , \mathbf{w}) = \ \sum_i (y^i - f(\mathbf{x^i}, \mathbf{w}))^2$$

Recap: Learning Weight Parameters with this Modified Neuron Model

- If we consider a collection of training examples and Sum the above error term over all examples

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

- Minimize errors using an optimization algorithm:
  - Gradient Descent (GD)
  - Stochastic Gradient Descent (SGD)

$E(\mathbf{w})$ term is also known as ***loss function***

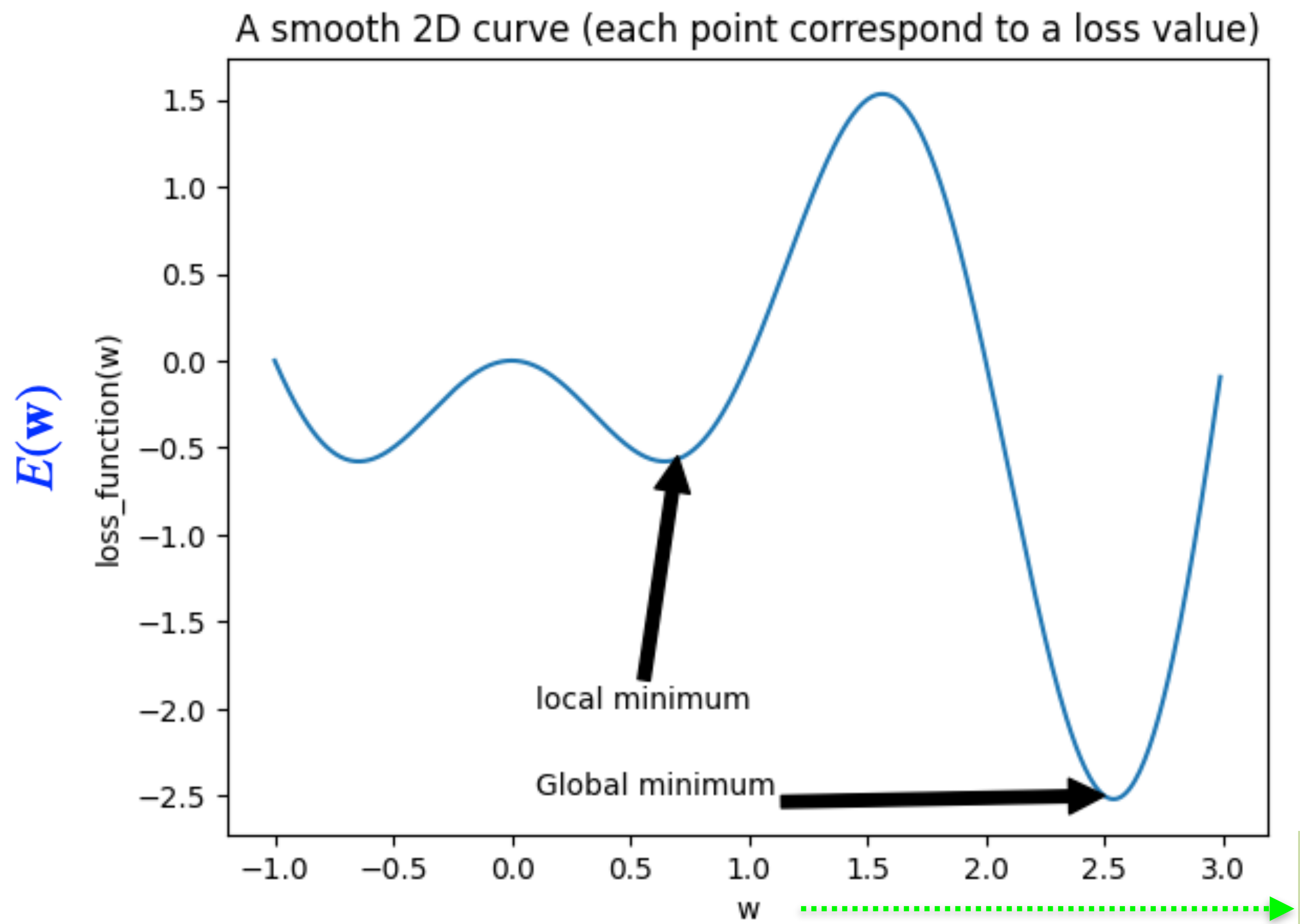
Recap: Optimization

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function

- The term objective function is generalized term which leaves room for the function to be something that we want to either **minimize** or **maximize**. The other terms used for the minimizing setting are as follows:
  - loss function
  - error function
  - cost function

Recap: Optimization Intuition

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function

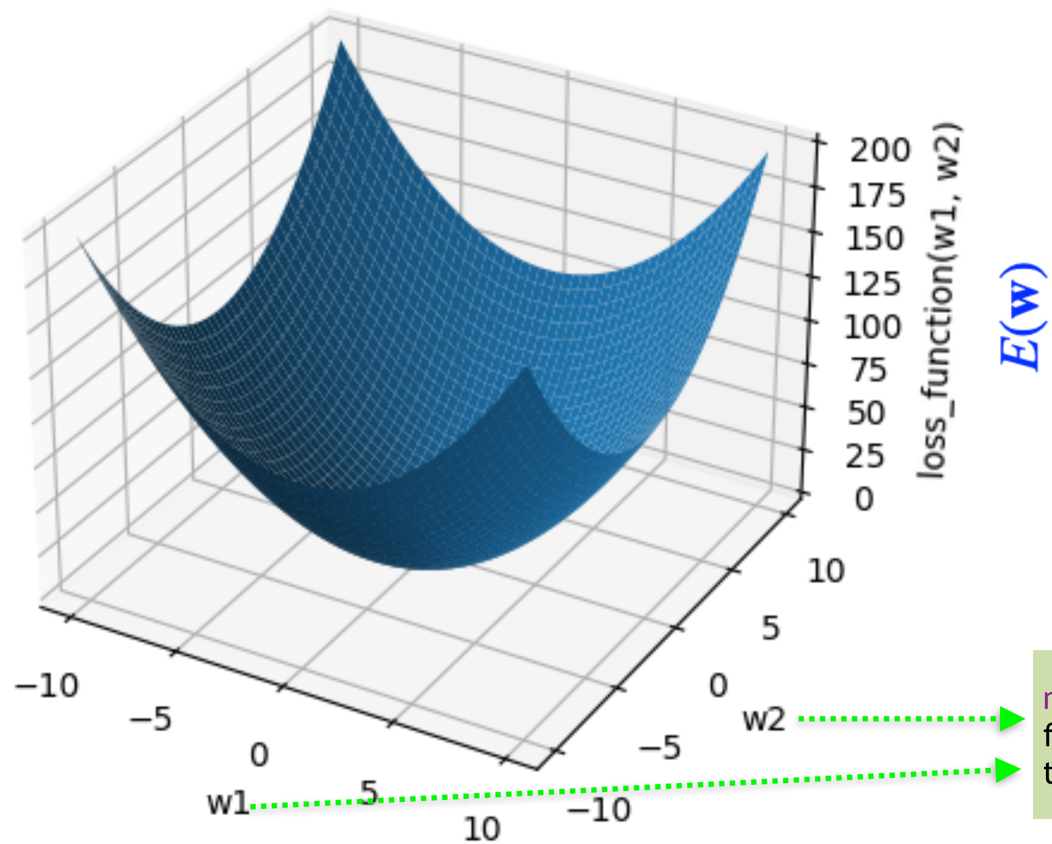


minimizing the loss function with respect to only one value w

Recap: Optimization Intuition

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function

A smooth 3D surface (each point correspond to a loss value)



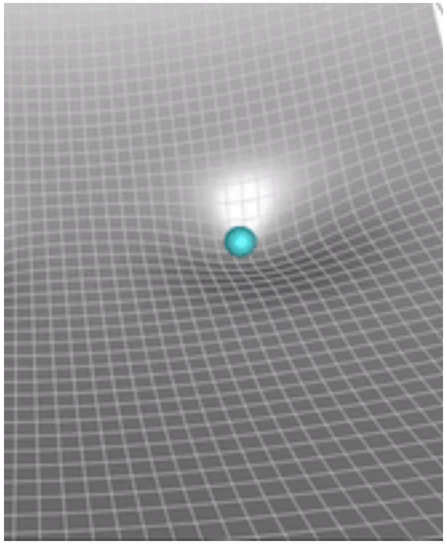
Play with the code I uploaded on Blackboard to generates different surfaces

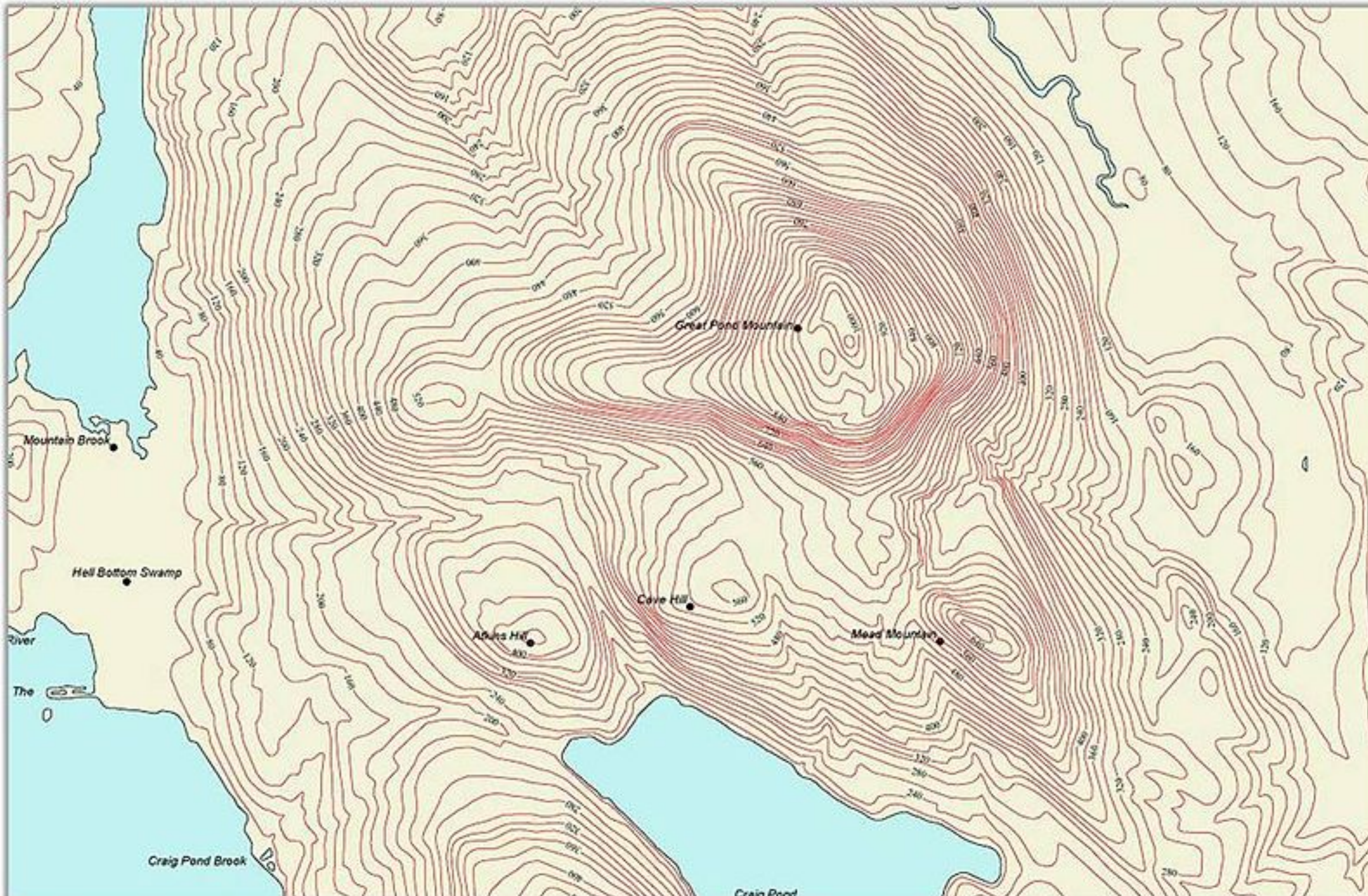
minimizing the loss function with respect to two values w_1 and w_2

Optimization Intuition

- **minimization**: trying to find the subset of values for attributes that gives you the minimum value in the objective function

- **How to reach to the minimum?**
    - we can start at an arbitrary point on the surface and gradually explore the surface until we reach the minimum value

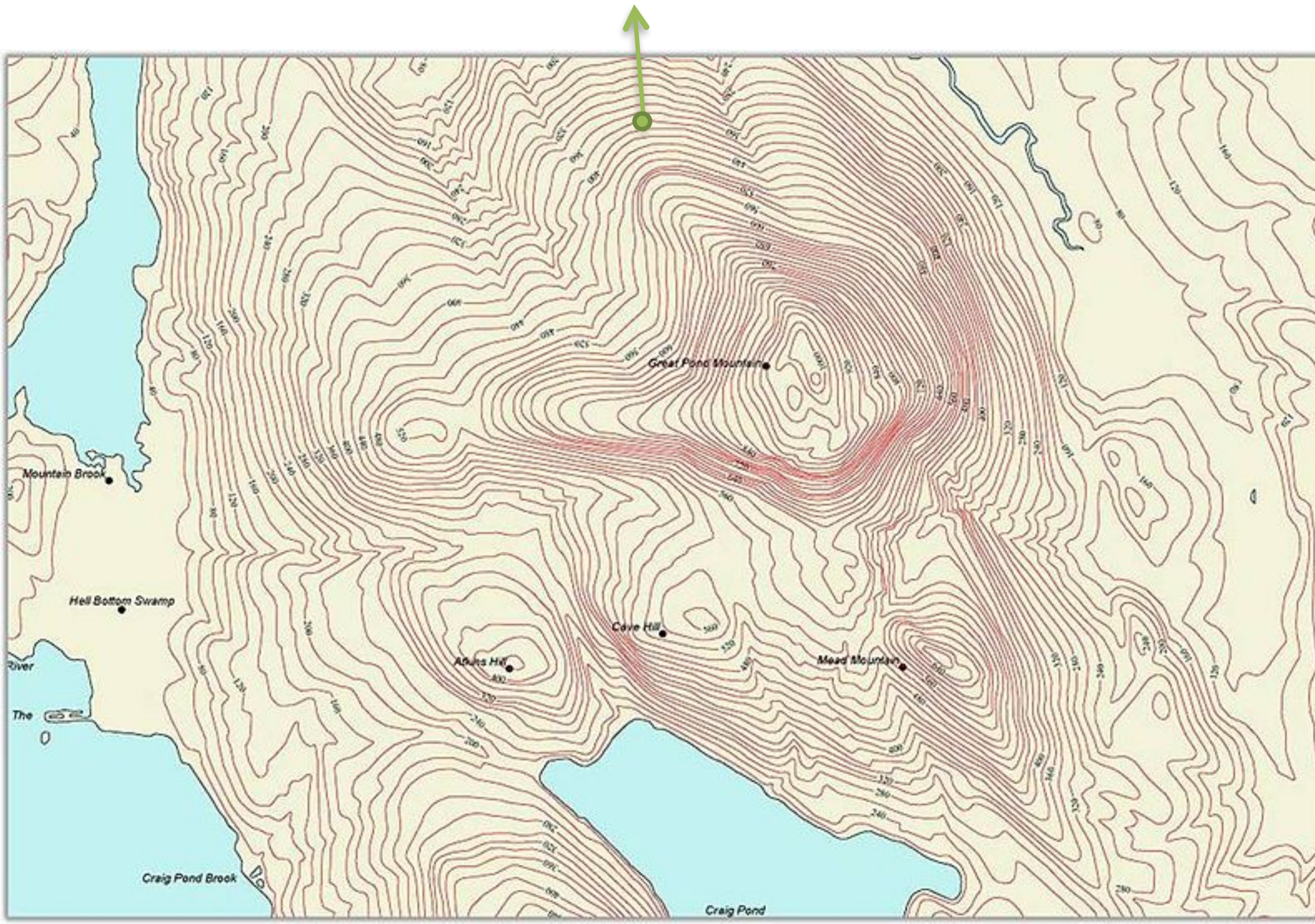




iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

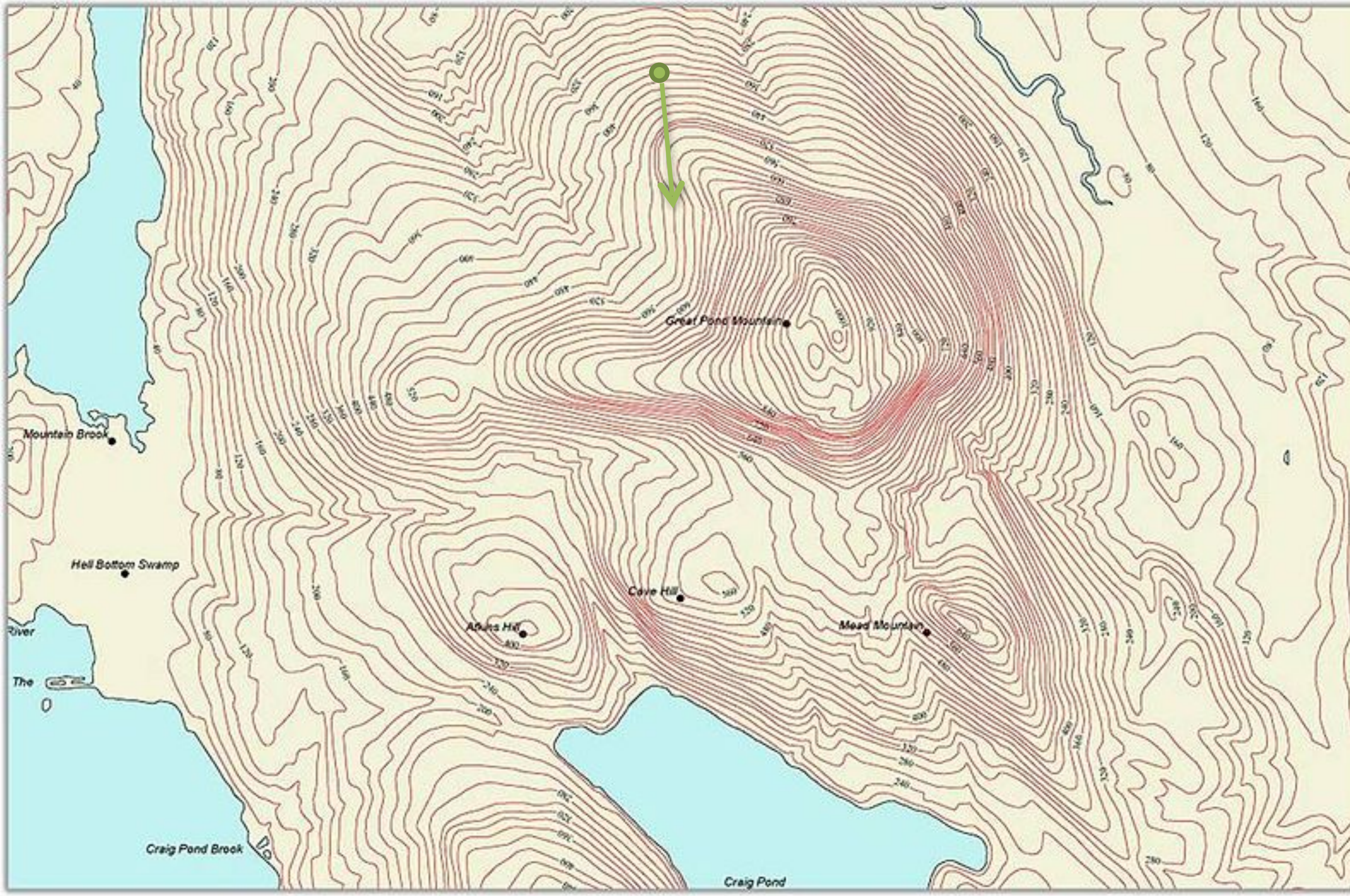
Adapted from K. Hauser's slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}\ , y_i\ , \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

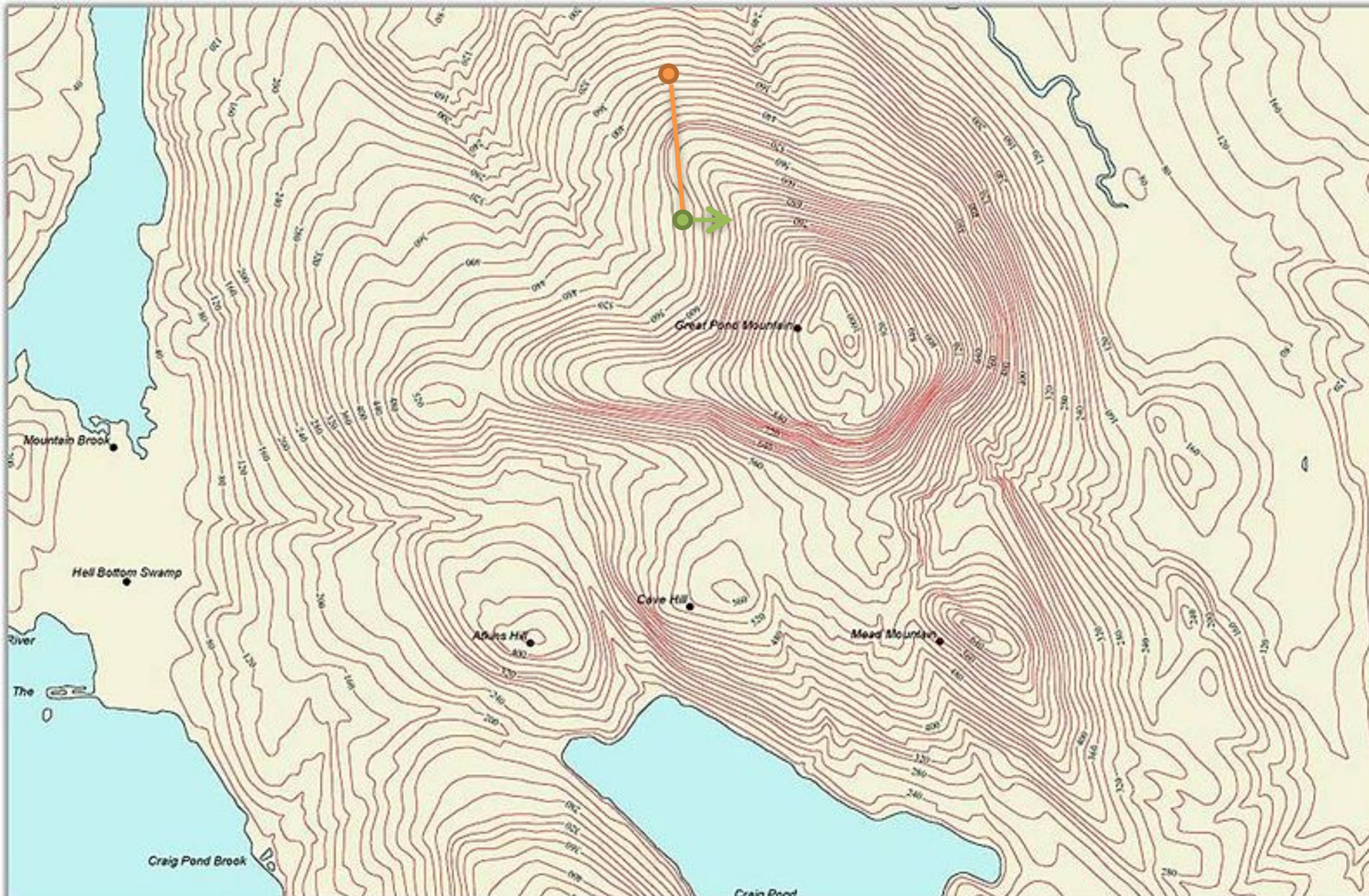
Adapted from K. Hauser’s slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

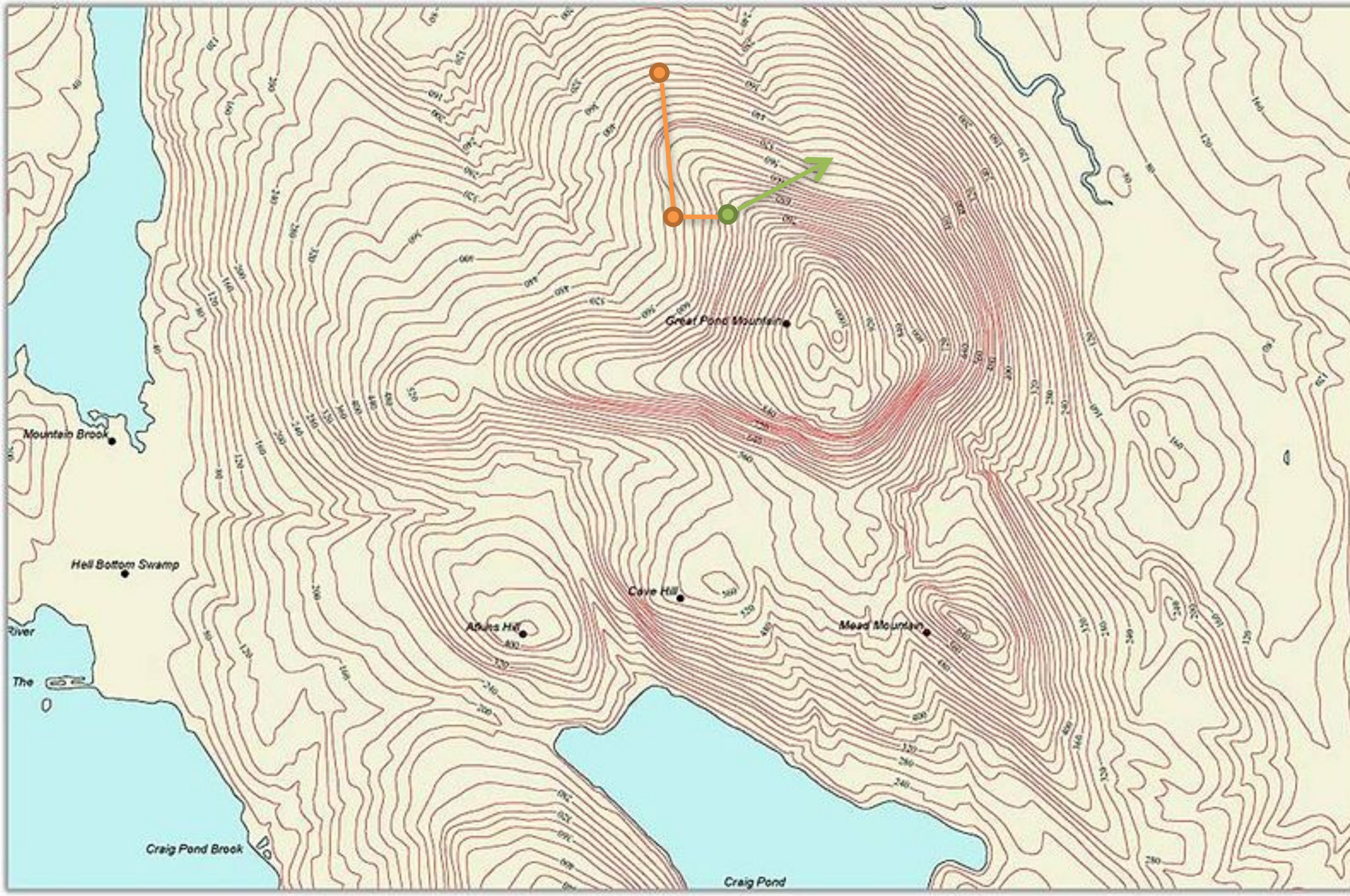
Adapted from K. Hauser's slide



iterately move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

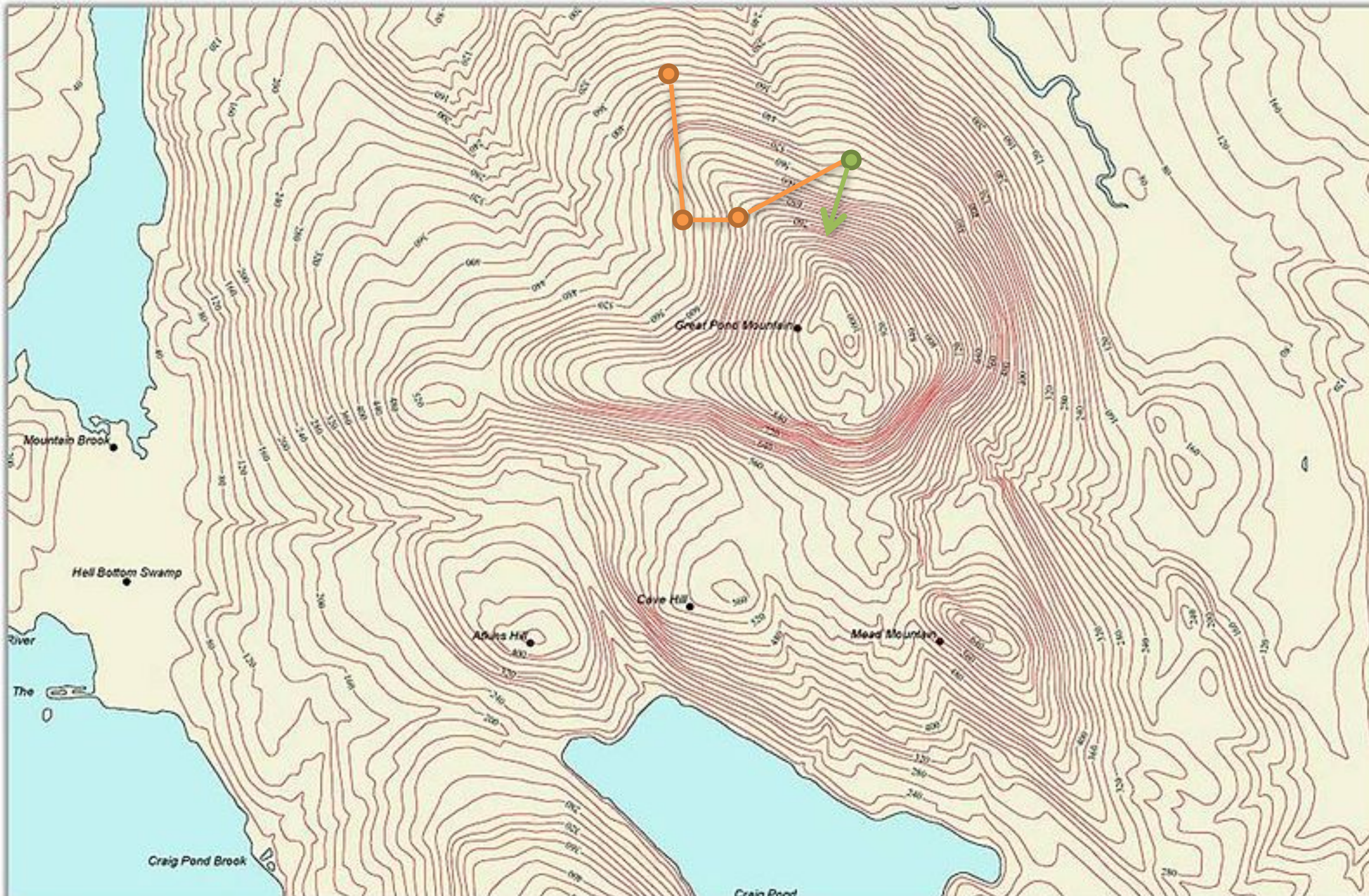
Adapted from K. Hauser's slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

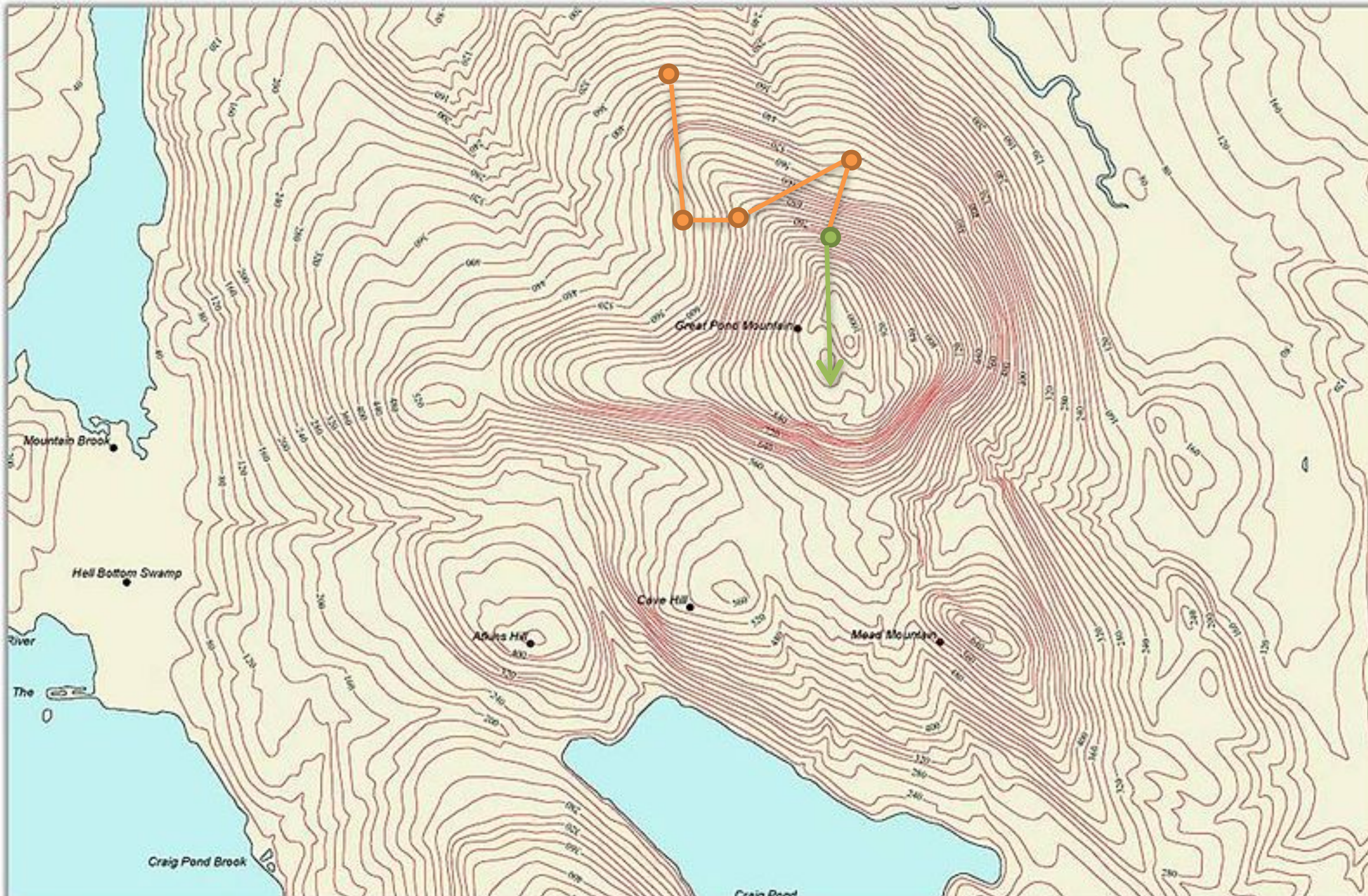
Adapted from K. Hauser's slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

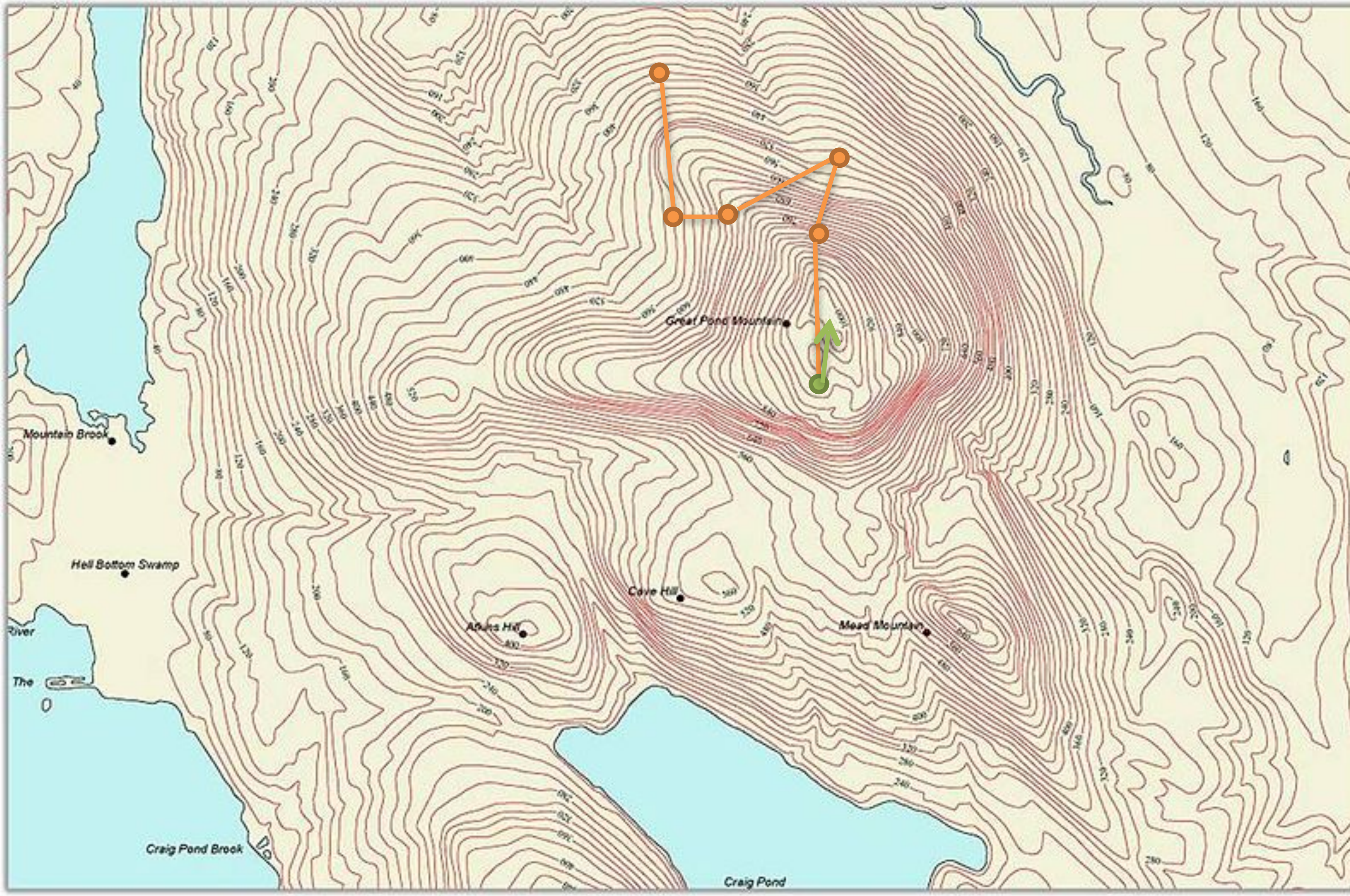
Adapted from K. Hauser's slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

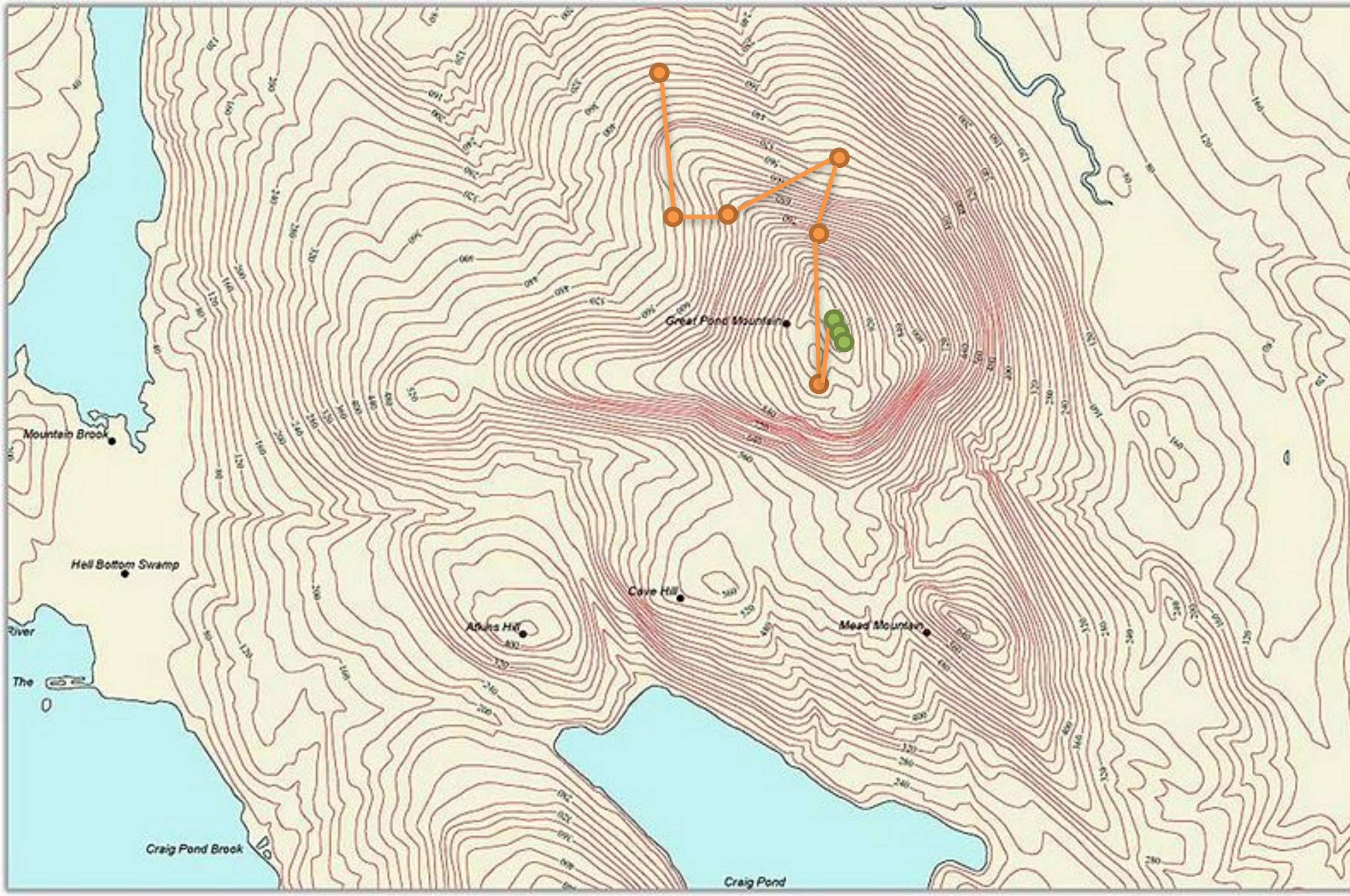
Adapted from K. Hauser's slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

Adapted from K. Hauser's slide



iteratively move in direction $-\nabla E$

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

Adapted from K. Hauser's slide

Today's Agenda

- Finding Moving Direction in Loss Surface (Gradient Calculation)
- Gradient Descent
- Stochastic Gradient Descent (SGD)

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- We saw how to calculate the moving direction. We can calculate the error term $E(\mathbf{w})$ over all training examples

$$E(\mathbf{w}) = \sum_i Error(\mathbf{x_i}, y_i, \mathbf{w}) = \sum_i (y_i - f(\mathbf{x_i}, \mathbf{w}))^2$$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \cdots \\ \cdots \\ w_n \end{bmatrix}$$

Sigmoid activation

$$f(\mathbf{x}^{\mathbf{i}}, \mathbf{w}) = \frac{\mathbf{1}}{\mathbf{1} + \mathbf{exp}^{-\mathbf{w}^{\mathbf{T}}\mathbf{x}^{\mathbf{i}}}}$$

$$\mathbf{x}^{\mathbf{i}} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \cdots \\ \cdots \\ x_n \end{bmatrix}$$

y^i

We also know $\mathbf{x}^{\mathbf{i}}$'s ground truth label (integer or floating point number)

training labels

MEDV
21.7
24.3
16.4

boston-housing dataset training split

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00
1.34284	0.0	19.58	0	0.605	6.066	100.0	1.7573	5	403	14.7	6.43
4.81213	0.0	18.10	0	0.713	6.701	90.0	2.5975	24	666	20.2	16.42

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_n \end{bmatrix}$$

$$\nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\delta E(\mathbf{w})}{\delta w_0} \\ \frac{\delta E(\mathbf{w})}{\delta w_1} \\ \cdots \\ \frac{\delta E(\mathbf{w})}{\delta w_n} \end{bmatrix}$$

partial derivative w.r.t scalar term w_0

partial derivative w.r.t scalar term w_1

partial derivative w.r.t scalar term w_n

Gradient of $E(\mathbf{w})$ with respect to $\mathbf{w}$ is defined by the vector of partial derivatives

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

Gradient of $E(\mathbf{w})$ with respect to $\mathbf{w}$ is defined by the vector of partial derivatives

$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_n \end{bmatrix} \qquad \nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\delta E(\mathbf{w})}{\delta w_0} \\ \frac{\delta E(\mathbf{w})}{\delta w_1} \\ \cdots \\ \frac{\delta E(\mathbf{w})}{\delta w_n} \end{bmatrix}$$

each gradient term, eg, partial derivative

$$\frac{\delta E(\mathbf{w})}{\delta w_1}$$

is a measure of change. It indicates the rate of change of E(w) when we change the value of only w_1 and don't change any other values (retain the old values for remaining: w_0, w_2, …, w_n)

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

Gradient of $E(\mathbf{w})$ with respect to $\mathbf{w}$ is defined by the vector of partial derivatives

- Let's simplify the model **f(x, w)** and take only 1 training example to see an example of Gradient calculation

boston-housing dataset training split

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00

training labels

MEDV
21.7

$$E(\mathbf{w}) = (y_1 - f(\mathbf{x_1}, \mathbf{w}))^2$$

$$f(\mathbf{x_1}, \mathbf{w}) = \mathbf{w}^{\mathbf{T}}\mathbf{x_1}$$

vector-vector dot product

$$= w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_{12}x_{12}$$

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

Gradient of $E(\mathbf{w})$ with respect to $\mathbf{w}$ is defined by the vector of partial derivatives

boston-housing dataset training split

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
0.10793	0.0	8.56	0	0.520	6.195	54.4	2.7778	5	384	20.9	13.00

training labels

MEDV
21.7

$$E(\mathbf{w}) = (y_1 - f(\mathbf{x_1}, \mathbf{w}))^2$$

$$f(\mathbf{x_1}, \mathbf{w}) = w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_{12}x_{12}$$

Let's plug-in the values from the training example and calculate E(w)

$$\begin{aligned} E(\mathbf{w}) &= (21.7 - w_0x_0 - w_1x_1 - w_2x_2 - \ldots - w_{12}x_{12})^2 \\ &= (21.7 - w_0 - 0.11w_1 - 0w_2 - \ldots - 13.0w_{12})^2 \end{aligned}$$

Iteratively Moving in Direction of $-\nabla E(\mathbf{w})$

- Need to calculate the Gradient of a scalar-valued $E(\mathbf{w})$ with respect to the weight vector $\mathbf{w}$

partial differentiation of $E(\mathbf{w})$ with respect to $\mathbf{w}$

Differential calculus

Gradient of $E(\mathbf{w})$ with respect to $\mathbf{w}$ is defined by the vector of partial derivatives

$$E(\mathbf{w}) = (21.7 - w_0 - 0.11w_1 - 0w_2 - \ldots - 13.0w_{12})^2$$

$$\nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\delta E(\mathbf{w})}{\delta w_0} \\ \frac{\delta E(\mathbf{w})}{\delta w_1} \\ \ldots \\ \frac{\delta E(\mathbf{w})}{\delta w_n} \end{bmatrix}$$

gradient term, eg, partial derivative

$$\frac{\delta E(\mathbf{w})}{\delta w_1}$$

$$\begin{aligned} \frac{\delta E(\mathbf{w})}{\delta w_1} &= \frac{\delta(21.7 - w_0 - 0.11w_1 - 0w_2 + \ldots - 13.0w_{12})^2}{\delta w_1} \\ &= \frac{\delta Z^2}{\delta Z}\frac{\delta Z}{\delta w_1} \\ &= 2Z\frac{\delta Z}{\delta w_1} \\ &= 2Z\frac{\delta(21.7 - w_0 - 0.11w_1 - 0w_2 - \ldots - 13.0w_{12})}{\delta w_1} \\ &= 2Z(-0.11) \end{aligned}$$

chain-rule in derivative

Today's Agenda

- Finding Moving Direction in Loss Surface (Gradient Calculation)
- Gradient Descent
- Stochastic Gradient Descent (SGD)

Gradient Descent

- Initialize the weight vector at a random position $\mathbf{w}^{\mathbf{old}}$ (random set of values)
- Keep doing the following two steps sequentially until the loss function gets to a low value (eg, below a threshold)

  - Step 1: calculate how much the loss function would change if we make a small change with respect to one weight component without perturbing any other weight terms. This is the gradient term corresponding to that particular weight. When you put them all together, they become the gradient vector:

$$\nabla E(\mathbf{w})$$

  - Step 2: adjust (or update) the values of the weights based on the gradient vector computed in the previous step:

$$\mathbf{w}^{\mathbf{new}} = \mathbf{w}^{\mathbf{old}} - \eta \nabla \mathbf{E}(\mathbf{w})$$

a fixed learning rate

Gradient Descent

- Initialize the weight vector at a random position $\mathbf{w}^{\mathbf{old}}$ (random set of values)
- Keep doing the following two steps sequentially until the loss function gets to a low value (eg, below a threshold)

  - Step 1: calculate the gradient vector $\nabla E(\mathbf{w})$

  - Step 2: adjust (or update) the values of the weights based on the gradient vector computed in the previous step:

$$\mathbf{w}^{\mathbf{new}} = \mathbf{w}^{\mathbf{old}} - \eta \nabla \mathbf{E}(\mathbf{w})$$

$$\begin{bmatrix} w_0^{new} \\ w_1^{new} \\ \ldots \\ w_n^{new} \end{bmatrix} = \begin{bmatrix} w_0^{old} \\ w_1^{old} \\ \ldots \\ w_n^{old} \end{bmatrix} - \eta \begin{bmatrix} \frac{\delta E(\mathbf{w})}{\delta w_0} \\ \frac{\delta E(\mathbf{w})}{\delta w_1} \\ \ldots \\ \frac{\delta E(\mathbf{w})}{\delta w_n} \end{bmatrix}$$

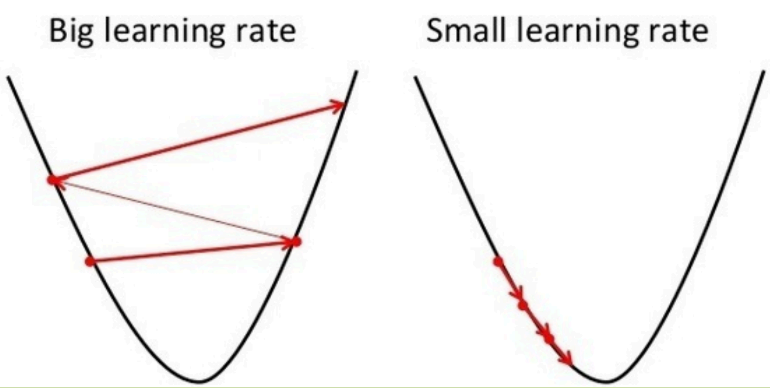
Gradient Descent and Learning Rate

- The learning rate is a fixed parameter that controls how much jump we make in each step of gradient descent

$$\mathbf{w}^{\mathbf{new}} = \mathbf{w}^{\mathbf{old}} - \eta \nabla \mathbf{E}(\mathbf{w})$$

a fixed learning rate

- we should not use too large η otherwise it will cause drastic update might diverge from the minimum point
- should not use too small η otherwise it will converge very slowly



Pick a learning rate that ensures that we change our weights at the right pace, not making any changes that are too big or too small

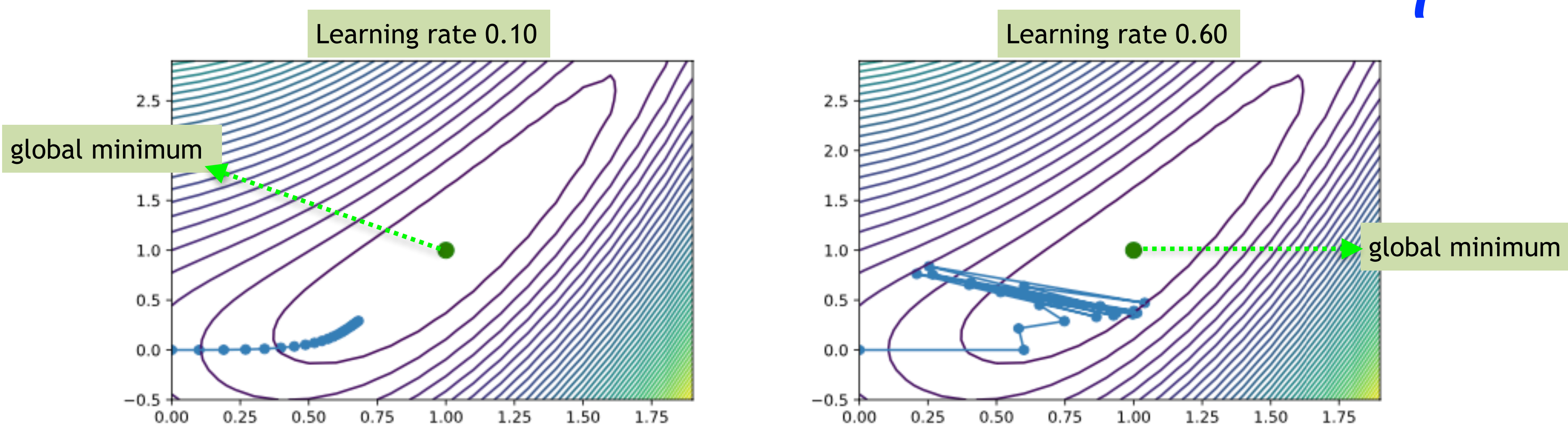
Gradient Descent and Learning Rate

- The learning rate is a fixed parameter that controls how much jump we make in each step of gradient descent

$$\mathbf{w}^{\mathbf{new}} = \mathbf{w}^{\mathbf{old}} - \eta \nabla \mathbf{E}(\mathbf{w})$$

a fixed learning rate

- We don't want to jump too fast; we should not use too large η
- Neither we want to move very slowly; should not use too small η



Another visualization of effect of different learning rates (from Probabilistic Machine Learning by Kevin Murphy textbook)

Today's Agenda

- Finding Moving Direction in Loss Surface (Gradient Calculation)
- Gradient Descent
- Stochastic Gradient Descent (SGD)

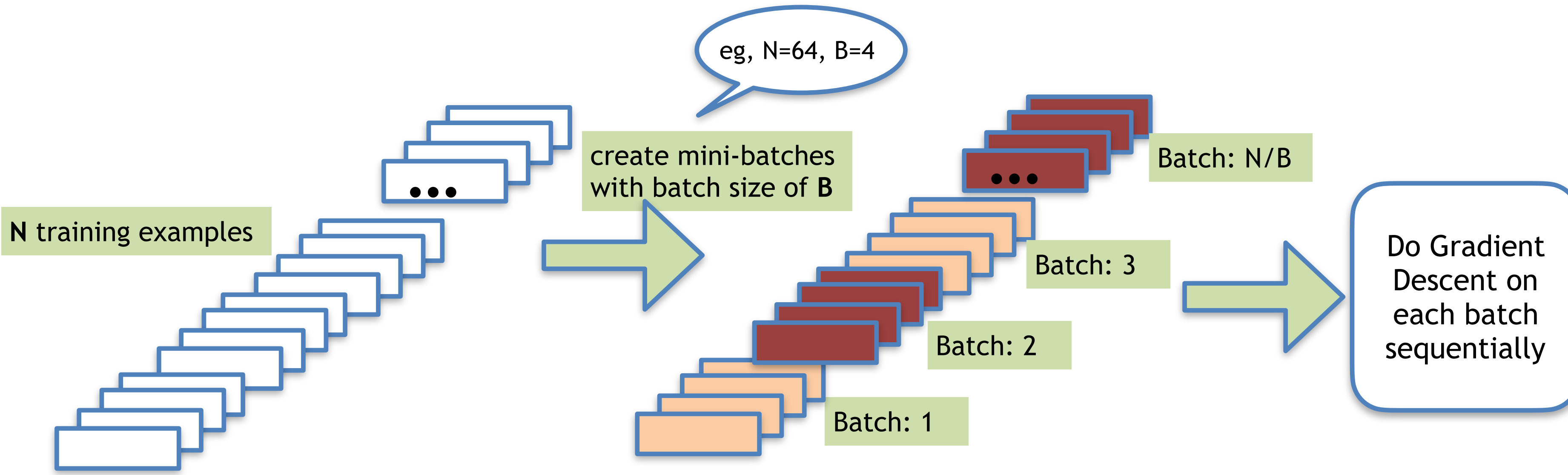
Stochastic Gradient Descent (SGD)

- Keep doing the Gradient Descent, but instead of using all the training samples, *use small subset of training samples* picked randomly when computing the gradient vector

  - divide the entire training data into mini batches
  - calculate the gradient vector based on that batch $\nabla E(\mathbf{w})$
  - adjust (or update) the values of the weights based on the gradient vector to that batch

$$\mathbf{w}^{\mathbf{new}} = \mathbf{w}^{\mathbf{old}} - \eta \nabla \mathbf{E}(\mathbf{w})$$

Stochastic Gradient Descent (SGD)



Useful Online Resources for Gradient Descent

- Another [mathematical derivation for Gradient Descent]
- One more [mathematical derivation for Gradient Descent]
- [Google course's Gradient Descent]
- [Visualization of Gradient Descent]
- [Visual explanation of Gradient Descent and other optimizers]

Group Activity: Stochastic Gradient Descent (SGD)

- SGD is doing a Gradient Descent, but instead of using all the training samples, it uses a *small subset of training samples* picked randomly when computing the gradient vector

In-class activity#7 (Stochastic Gradient Descent - SGD)
Due date: 4/2/24, 11:59 PM
Visible to students

Complete the group activity from class today and upload your notebook. Here is the reference notebook: https://github.com/alimoorreza/CS167-sp24-notes/blob/main/Day16_Stochastic_Gradient_Descent_SGD.ipynb