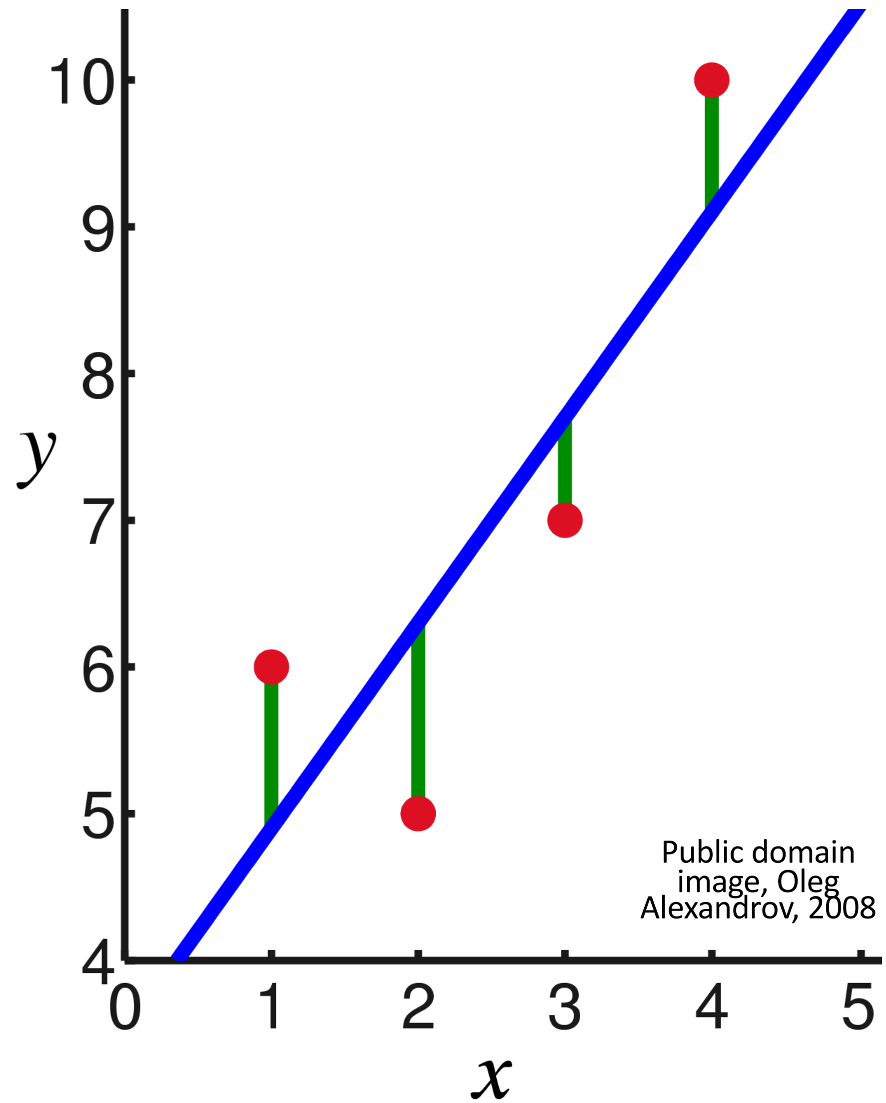
CS440/ECE448 Lecture 9: Linear Regression

Mark Hasegawa-Johnson, 2/2024



Outline

- Notation for vectors and matrices
- Definition of linear regression
- Mean-squared error
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

Vectors are lowercase bold letters

Vectors will always be column vectors. Thus:

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$\boldsymbol{w}^T = [w_1, \cdots, w_n]$$

$$\boldsymbol{w}^T\boldsymbol{x} = [w_1, \cdots, w_n] \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} w_i x_i$$

In numpy, the dot product can be written np.dot(w,x) or w@x.

Matrices are uppercase bold letters

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{W} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{bmatrix}$$

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,n} \\ \vdots & \ddots & \vdots \\ w_{m,1} & \cdots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} w_{1,i} x_i \\ \vdots \\ \sum_{i=1}^{n} w_{m,i} x_i \end{bmatrix}$$

In numpy, the matrix multiplication can be written np.matmul(w,x) or w@x.

Vector and Matrix Gradients

The gradient of a scalar function with respect to a vector or matrix is:

$$\frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}, \qquad \frac{\partial f}{\partial \boldsymbol{W}} = \begin{bmatrix} \frac{\partial f}{\partial w_{1,1}} & \cdots & \frac{\partial f}{\partial w_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial w_{m,1}} & \cdots & \frac{\partial f}{\partial w_{m,n}} \end{bmatrix}$$

The symbol $\frac{\partial f}{\partial x_1}$ means “partial derivative of f with respect to x_1.”

Outline

- Pythonic notation for vectors and matrices
- Definition of linear regression
- Mean-squared error
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

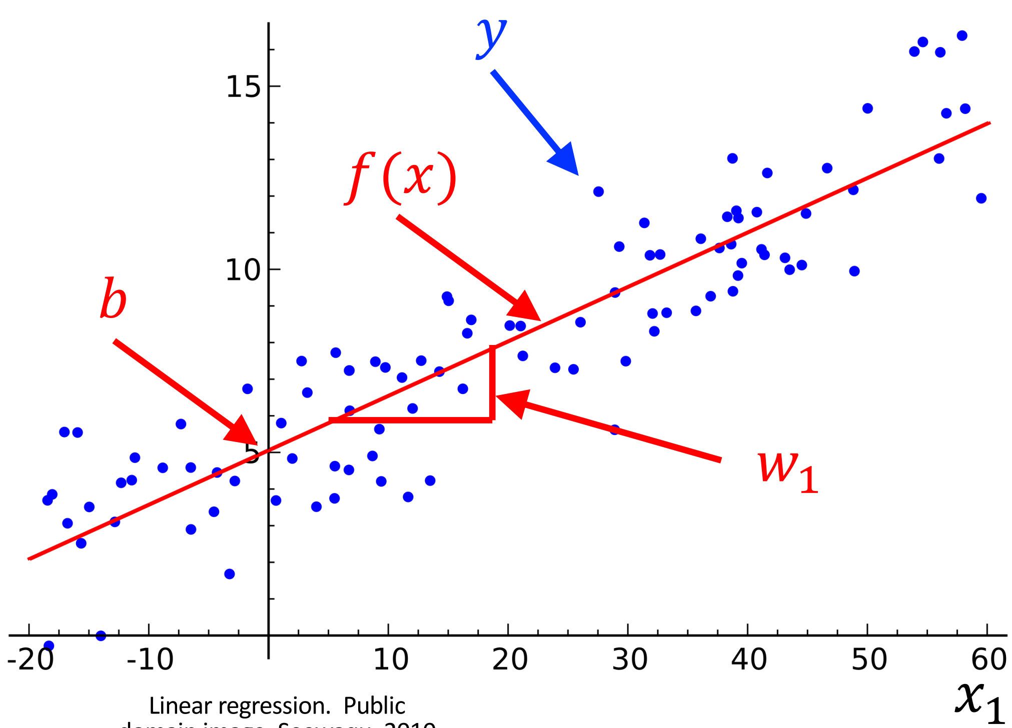
Linear regression

Linear regression is used to estimate a real-valued target variable, y, using a linear combination of real-valued input variables:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_{j=0}^{D-1} w_j x_j + b$$

… so that …

$$f(\boldsymbol{x}) \approx y$$



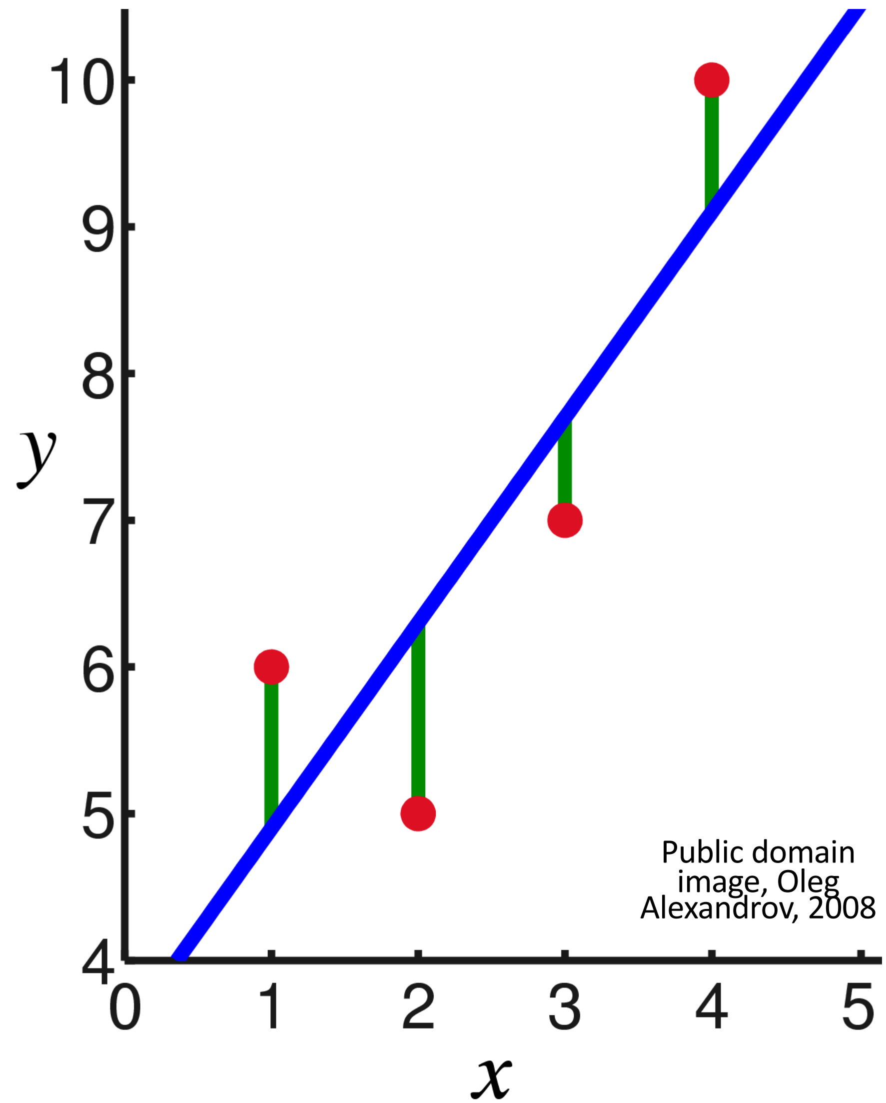
Linear regression. Public domain image, Seewaqu, 2010

Outline

- Pythonic notation for vectors and matrices
- Definition of linear regression
- Mean-squared error
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

What does it mean that $f(\boldsymbol{x}) \approx y$?

- Generally, we want to choose the weights and bias, w and b, in order to minimize the errors.
- The errors are the vertical green bars in the figure at right,
$$\epsilon = f(\boldsymbol{x}) - y$$
- Some of them are positive, some are negative. What does it mean to "minimize" them?



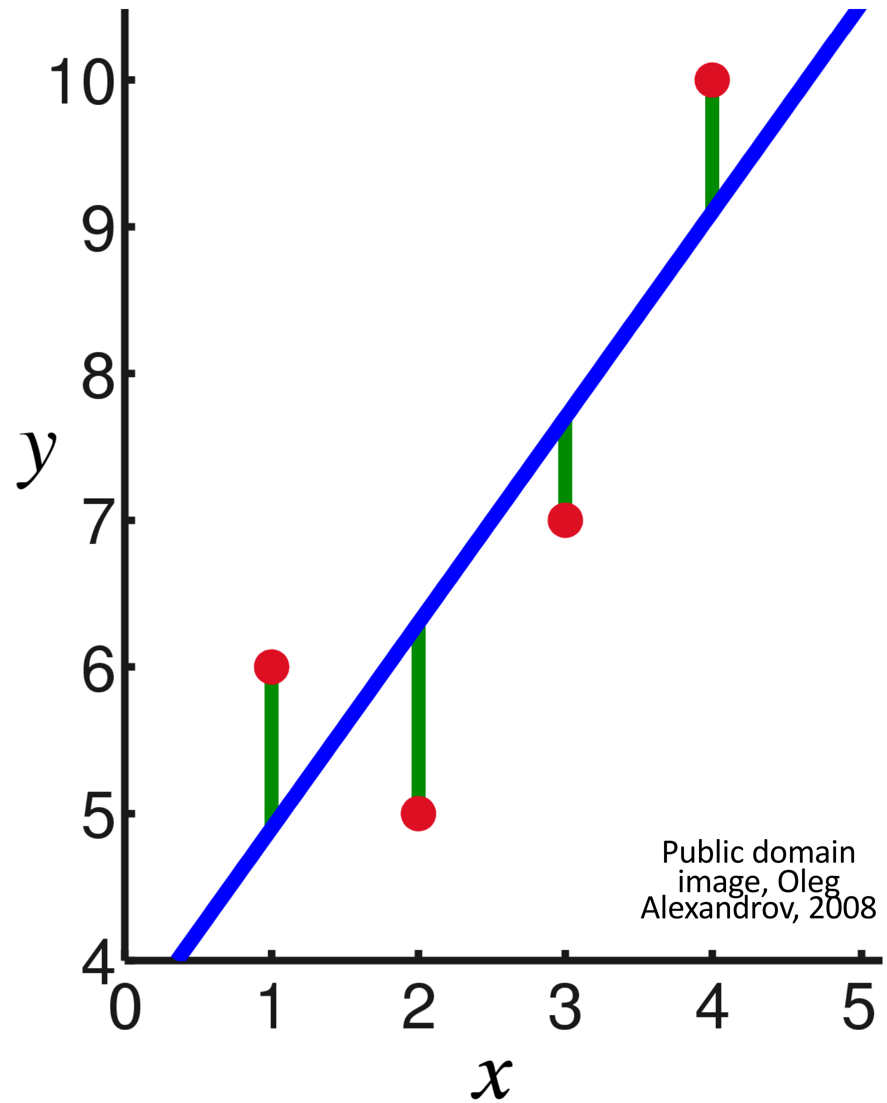
Public domain image, Oleg Alexandrov, 2008

First: count the training tokens

Let's introduce one more index variable. Let i=the index of the training token.

$$\boldsymbol{x}_i = \begin{bmatrix} 1 \\ x_{i,1} \\ \vdots \\ x_{i,n} \end{bmatrix}$$

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i + b = \sum_{j=0}^{D-1} x_{i,j} w_j + b$$

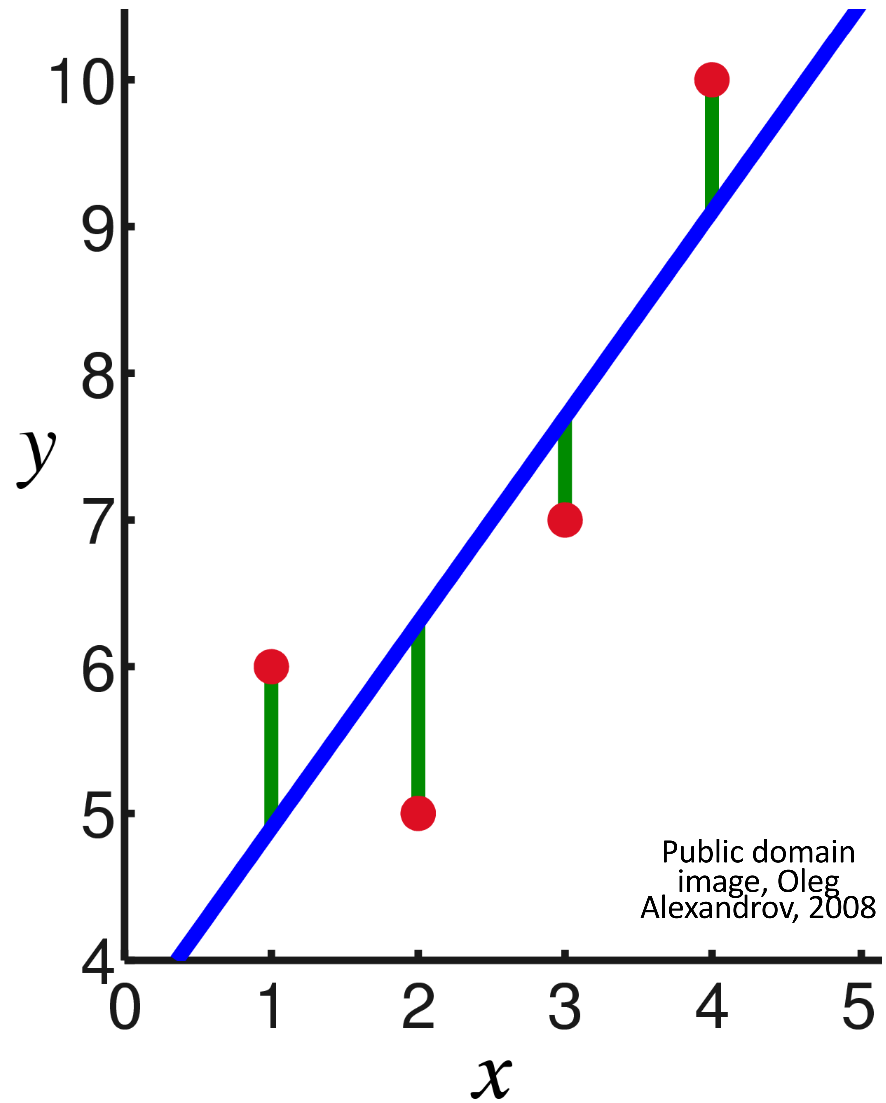


Training token errors

Using that notation, we can define a signed error term for every training token:

$$\epsilon_i = f(\boldsymbol{x}_i) - y_i$$

The error term is positive for some tokens, negative for other tokens. What does it mean to minimize it?



Mean-squared error

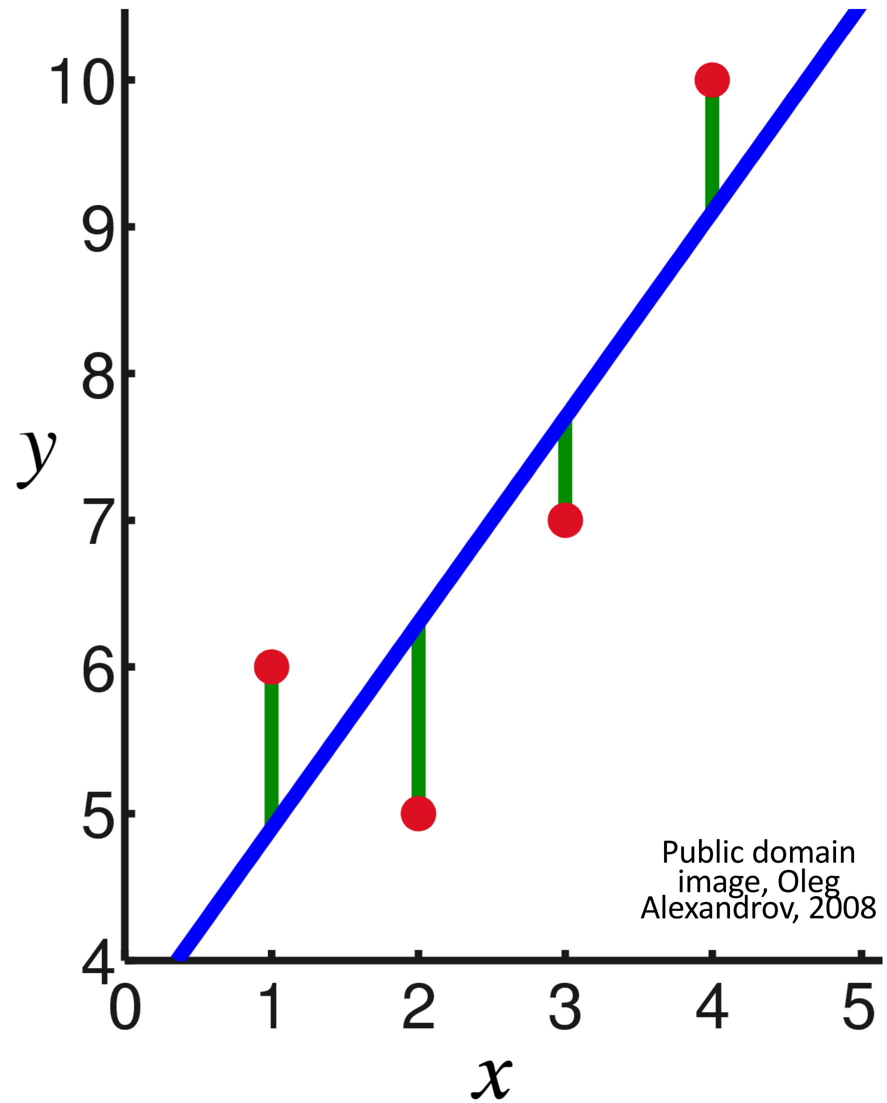
One useful criterion (not the only useful criterion, but perhaps the most common) of "minimizing the error" is to minimize the mean squared error:

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}\epsilon_i^2 = \frac{1}{2n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2$$

Literally,

- … the mean …
- … of the squares …
- … of the error terms.

The factor $\frac{1}{2}$ is included so that, so that when you differentiate $\mathcal{L}$, the 2 and the $\frac{1}{2}$ can cancel each other.



Outline

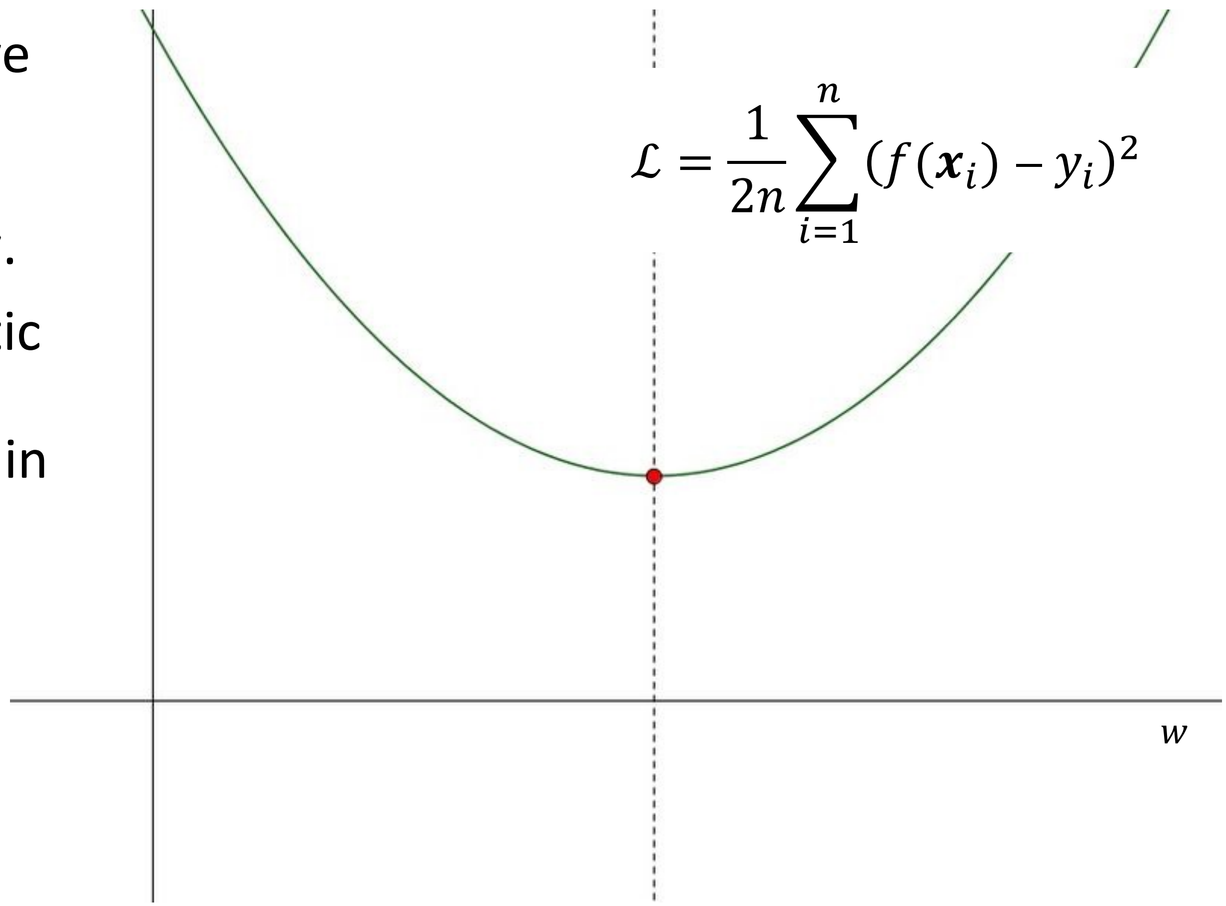
- Pythonic notation for vectors and matrices
- Definition of linear regression
- Mean-squared error
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

MSE = Parabola

Notice that MSE is a non-negative quadratic function of $f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i + b$, therefore it's a non-negative quadratic function of $\boldsymbol{w}$.

Since it's a non-negative quadratic function of $\boldsymbol{w}$, it has a unique minimum that you can compute in closed form!

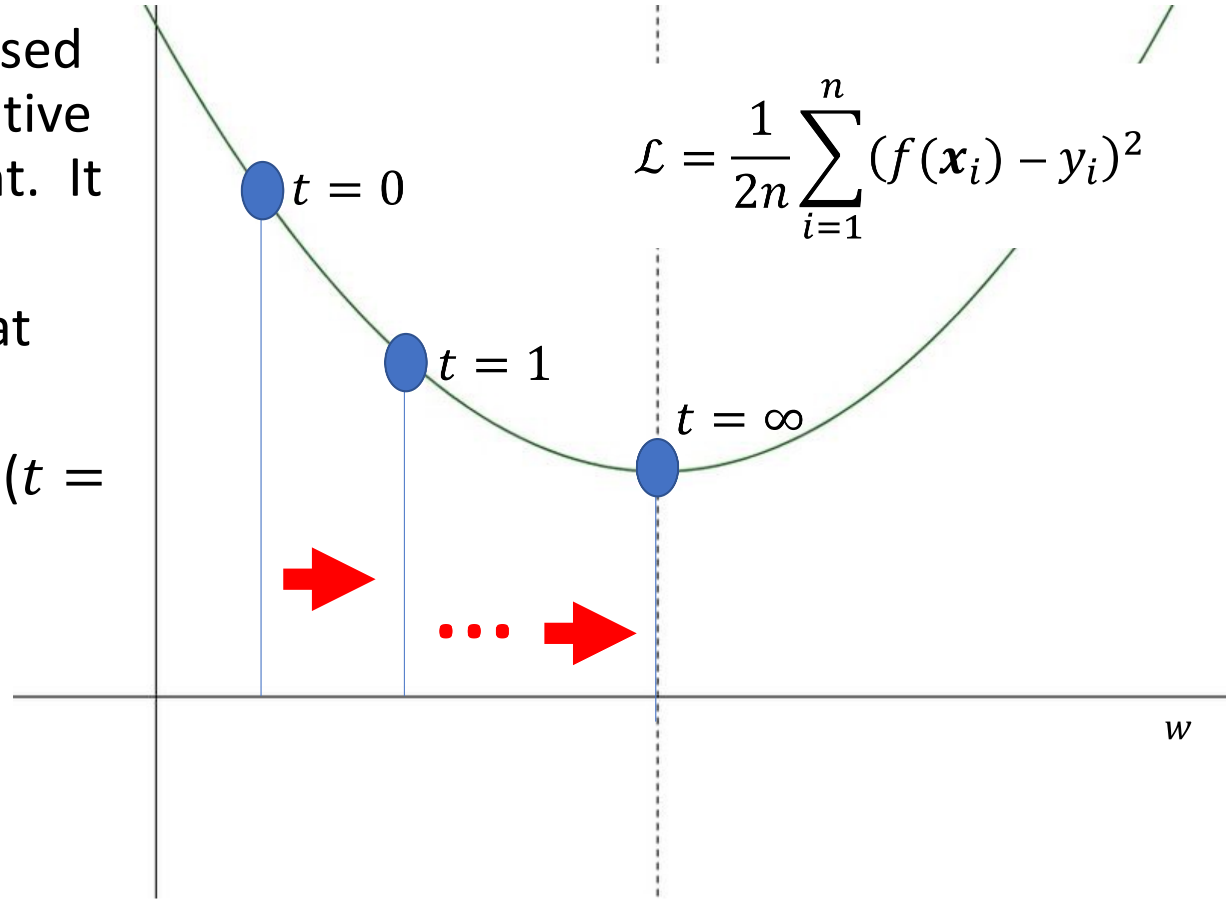
We won't do that today.



The iterative solution to linear regression

Instead of minimizing MSE in closed form, we're going to use an iterative algorithm called gradient descent. It works like this:

- Start: random initial $\boldsymbol{w}$ and b (at $t = 0$).
- Adjust $\boldsymbol{w}$ and b to reduce MSE ($t = 1$).
- Repeat until you reach the optimum ($t = \infty$).

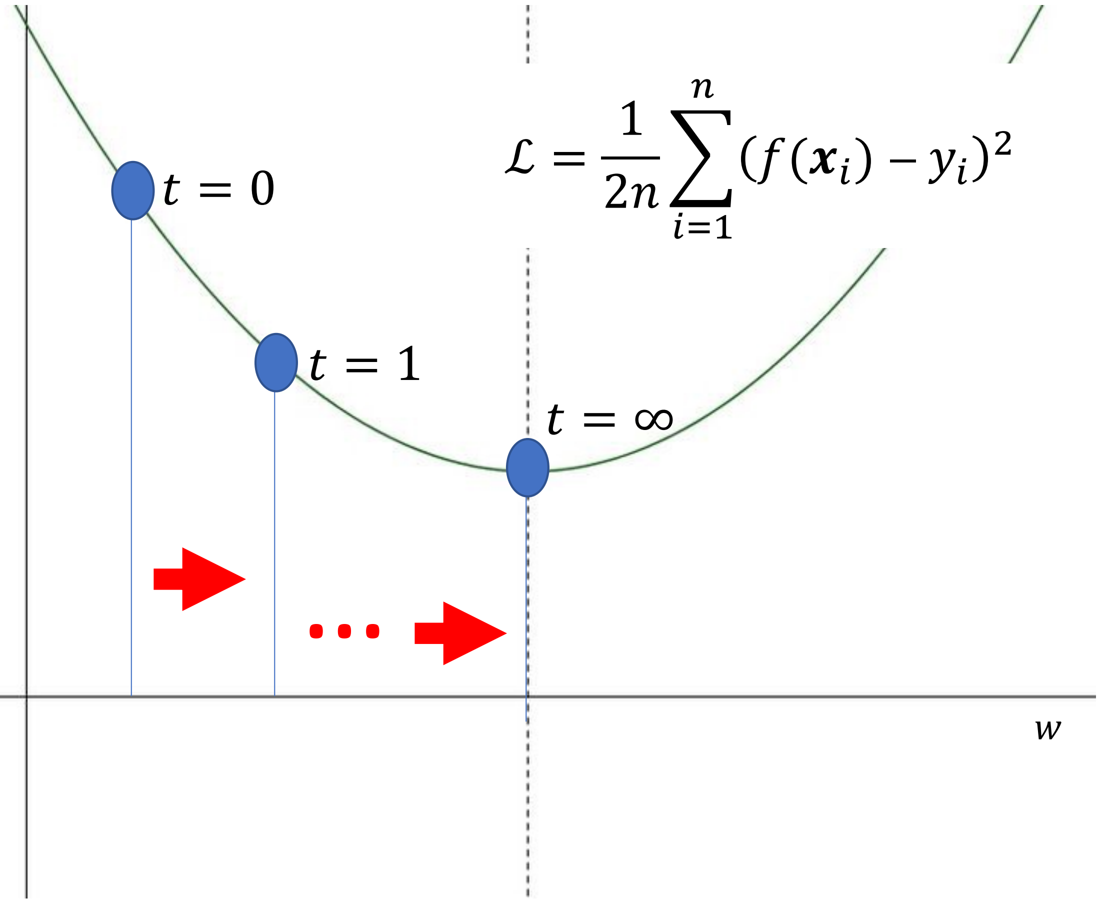


The iterative solution to linear regression

- Start from random initial values of $\boldsymbol{w}$ and b (at $t = 0$).
- Adjust w and b according to:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}$$
$$b \leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b}$$

…where η is a hyperparameter called the "learning rate," that determines how big of a step you take. Usually, you need to adjust η in order to get optimum performance on a dev set.



Finding the gradient

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n} \mathcal{L}_i\,, \qquad \mathcal{L}_i = \epsilon_i^2, \qquad \epsilon_i = \boldsymbol{w}^T\boldsymbol{x}_i + b - y_i$$

To find the gradient, we use the chain rule of calculus:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{2n}\sum_{i=1}^{n} \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}} = 2\epsilon_i \frac{\partial \epsilon_i}{\partial \boldsymbol{w}}, \qquad \frac{\partial \epsilon_i}{\partial \boldsymbol{w}} = \boldsymbol{x}_i$$

Putting it all together,

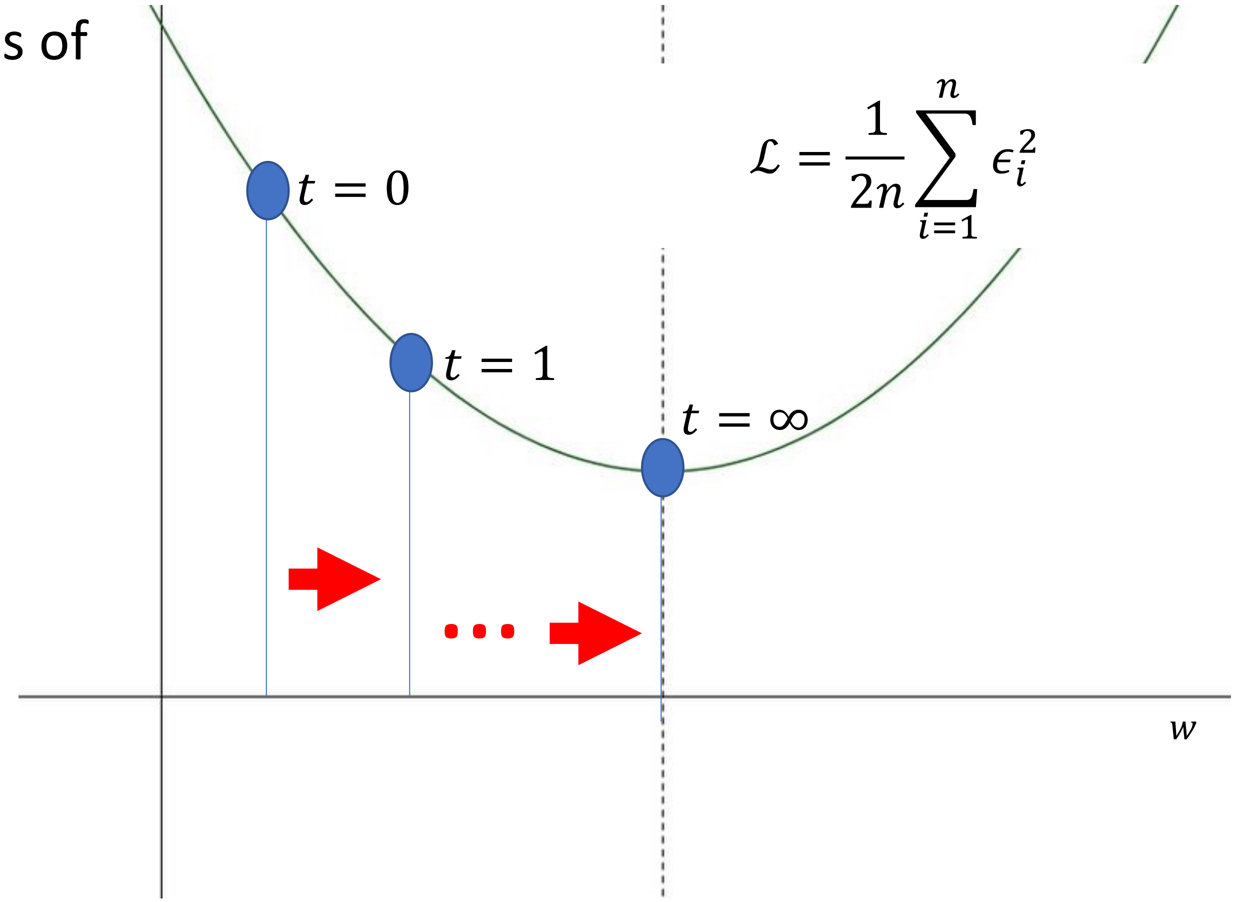
$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i$$

The iterative solution to linear regression

- Start from random initial values of $\boldsymbol{w}$ and b (at $t = 0$).
- Adjust $\boldsymbol{w}$ and b according to:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \frac{\eta}{n}\sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i$$

$$b \leftarrow b - \frac{\eta}{n}\sum_{i=1}^{n} \epsilon_i$$

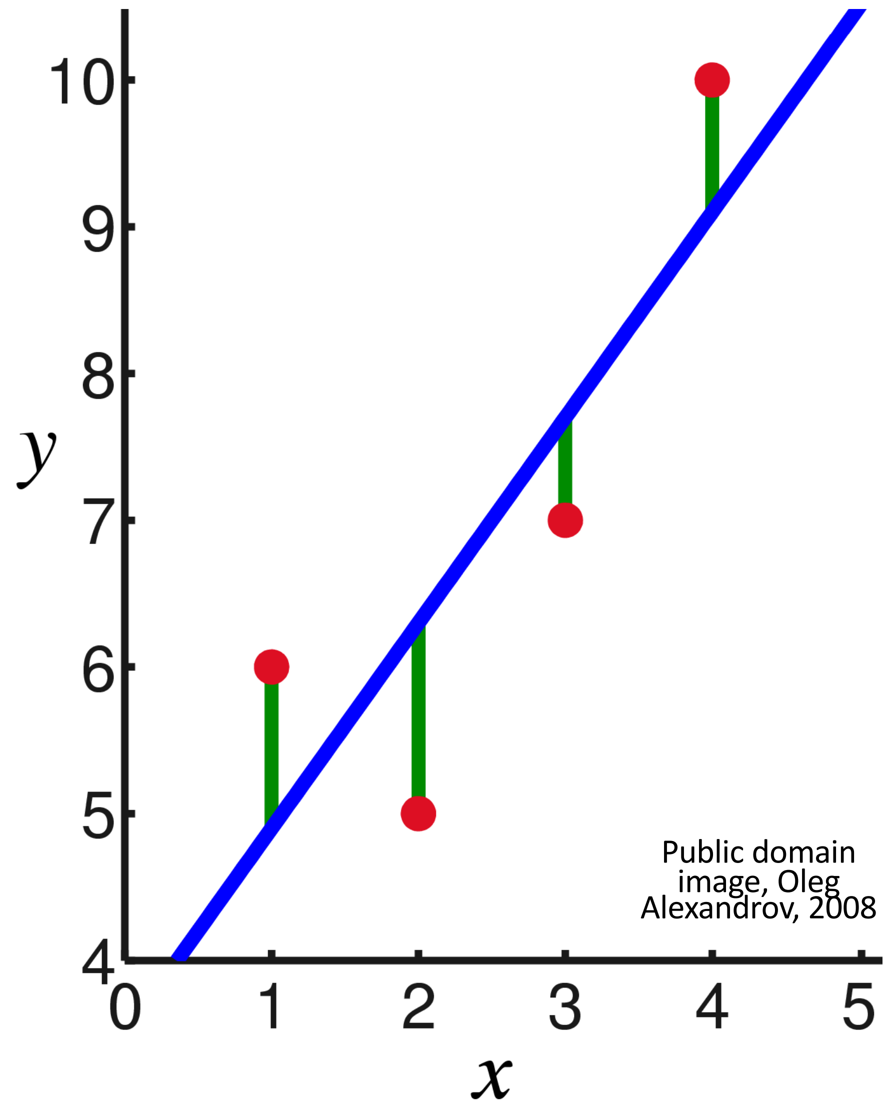


Intuition:

- Notice the sign:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \frac{\eta}{n}\sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i$$

- If ϵ_i is positive ($f(\boldsymbol{x}_i) > y_i$), then we want to **<u>reduce</u>** $f(\boldsymbol{x}_i)$, so we make $\boldsymbol{w}$ less like $\boldsymbol{x}_i$
- If ϵ_i is negative ($f(\boldsymbol{x}_i) < y_i$), then we want to **<u>increase</u>** $f(\boldsymbol{x}_i)$, so we make $\boldsymbol{w}$ more like $\boldsymbol{x}_i$



Public domain image, Oleg Alexandrov, 2008

Outline

- Pythonic notation for vectors and matrices
- Definition of linear regression
- Mean-squared error
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

Stochastic gradient descent

- If n is large, computing or differentiating MSE can be expensive.
- The stochastic gradient descent algorithm picks one training token $(\boldsymbol{x}_i, y_i)$ at random (”stochastically”), and adjusts w in order to reduce the error a little bit for that one token:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}}$$

…where

$$\mathcal{L}_i = \epsilon_i^2 = \frac{1}{2}\left(f(\boldsymbol{x}_i) - y_i\right)^2$$

Stochastic gradient descent

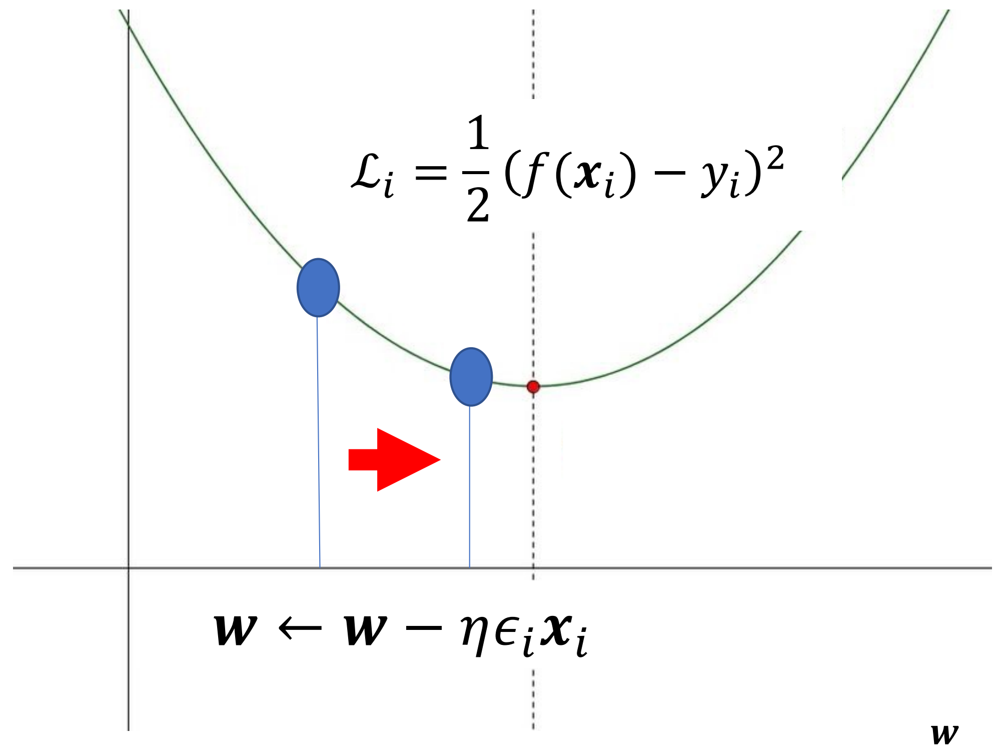
$$\mathcal{L}_i = \epsilon_i^2 = \frac{1}{2}(\boldsymbol{w}^T\boldsymbol{x}_i + b - y_i)^2$$

If we differentiate that, we discover that:

$$\frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}} = \epsilon_i \boldsymbol{x}_i, \qquad \frac{\partial \mathcal{L}_i}{\partial b} = \epsilon_i$$

So the stochastic gradient descent algorithm is:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\epsilon_i\boldsymbol{x}_i, \qquad b \leftarrow b - \eta\epsilon_i$$



The Stochastic Gradient Descent Algorithm

1. Choose a sample $(\boldsymbol{x}_i, y_i)$ at random from the training data
2. Compute the error of this sample, $\epsilon_i = \boldsymbol{w}^T \boldsymbol{x}_i + b - y_i$
3. Adjust $\boldsymbol{w}$ and b in the direction opposite the error:
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \epsilon_i \boldsymbol{x}_i$$
$$b \leftarrow b - \eta \epsilon_i$$
4. If the error is still too large, go to step 1. If the error is small enough, stop.

Today’s Quiz

Go to

https://us.prairielearn.com/pl/course_instance/147925/assessment/2395554, try the quiz!

Video of SGD

https://upload.wikimedia.org/wikipedia/commons/5/57/Stochastric_Gradient_Descent.webm

In this video, the different colored dots are different, randomly chosen starting points.

Each step of SGD uses a randomly chosen training token, so the direction is a little random.

But after a while, it reaches the bottom of the parabola!

Summary

- Definition of linear regression

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

- Mean-squared error

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} \mathcal{L}_i, \qquad \mathcal{L}_i = \epsilon_i^2, \qquad \epsilon_i = f(\boldsymbol{x}_i) - y_i$$

- Gradient descent

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{n} \sum_{i=1}^{n} \epsilon_i \boldsymbol{x}_i$$

- Stochastic gradient descent

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}}, \qquad \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}} = \epsilon_i \boldsymbol{x}_i$$