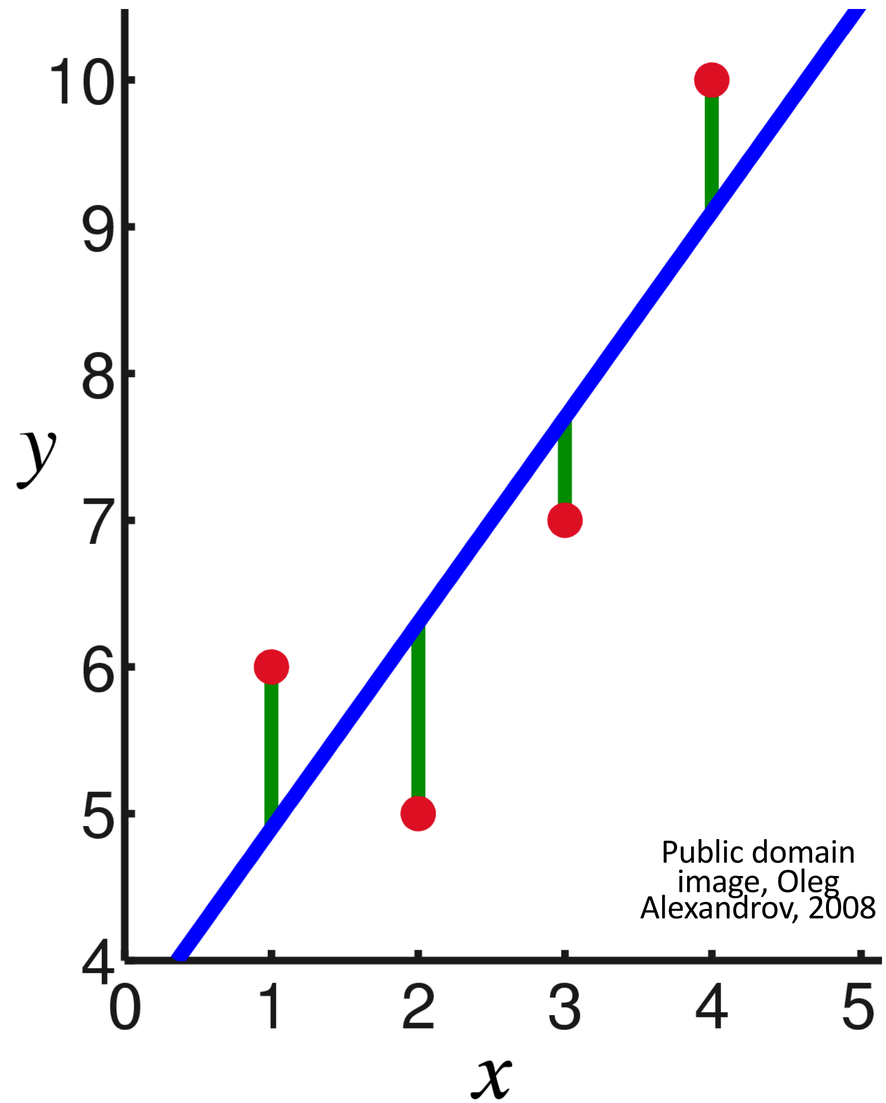
CS440/ECE448 Lecture 7: Linear Regression

Mark Hasegawa-Johnson, 1/2023



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

Python: a variable can be a container

- In python, a variable can hold any value
- The i^{th} element of x is $x[i]$
- The j^{th} element of $x[i]$ is $x[i][j]$
- The k^{th} element of $x[i][j]$ is $x[i][j][k]$

Lecture slides: subscripts = brackets

- On lecture slides (like this one), it's often convenient to use subscripts instead of brackets
- I will use subscripts as a ***synonym*** for brackets
- The i^{th} element of x can be written $x_{:,i}$ or $x[:,i]$
- The j^{th} element of x_i can be written $x_{i,j}$ or $x_i[j]$ or $x[i][j]$
- The k^{th} element of $x_{i,j}$ can be written $x_{i,j,k}$ or $x_{i,j}[k]$ or $x_i[j,k]$

Matrix multiplication

The following equations all mean the same thing:

$$u = v@w$$

$$\begin{bmatrix} u_{0,0} & \cdots & u_{0,N-1} \\ \vdots & \ddots & \vdots \\ u_{L-1,0} & \cdots & u_{L-1,N-1} \end{bmatrix} = \begin{bmatrix} v_{0,0} & \cdots & v_{0,M-1} \\ \vdots & \ddots & \vdots \\ v_{L-1,0} & \cdots & v_{L-1,M-1} \end{bmatrix} @ \begin{bmatrix} w_{0,0} & \cdots & w_{0,N-1} \\ \vdots & \ddots & \vdots \\ w_{M-1,0} & \cdots & w_{M-1,N-1} \end{bmatrix}$$

$$u_{l,n} = \sum_{m=0}^{M-1} v_{l,m} w_{m,n}$$

Vectors

A vector can be either a row vector OR a column vector, on demand, whichever best fits the context, so if $x = [x_0, \ldots, x_{N-1}]$, then

$$x@w = [x_0, \ldots, x_{N-1}]@\begin{bmatrix} w_{0,0} & \cdots & w_{0,N-1} \\ \vdots & \ddots & \vdots \\ w_{N-1,0} & \cdots & w_{N-1,N-1} \end{bmatrix}$$

…but…

$$w@x = \begin{bmatrix} w_{0,0} & \cdots & w_{0,N-1} \\ \vdots & \ddots & \vdots \\ w_{N-1,0} & \cdots & w_{N-1,N-1} \end{bmatrix}@\begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

Dot-product

If $x = [x_0, \dots, x_{N-1}]$ and $y = [y_0, \dots, y_{N-1}]$, then

$$x@y = y@x = [y_0, \dots, y_{N-1}]@\begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \sum_{i=0}^{N-1} x_i y_i$$

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

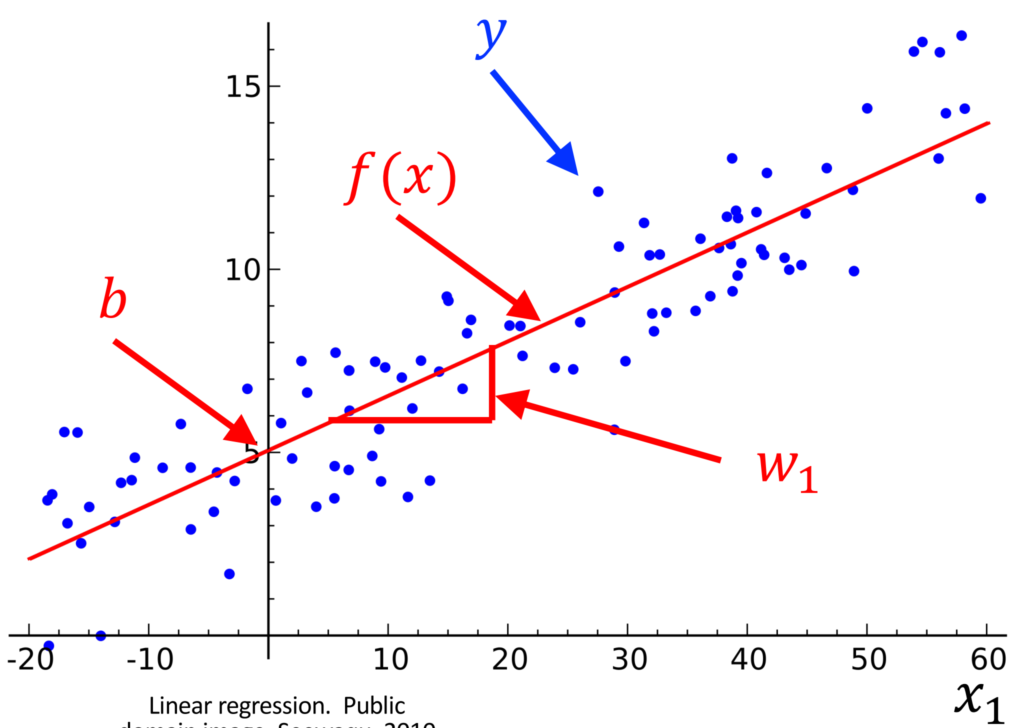
Linear regression

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

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

… so that …

$$f(x) \approx y$$



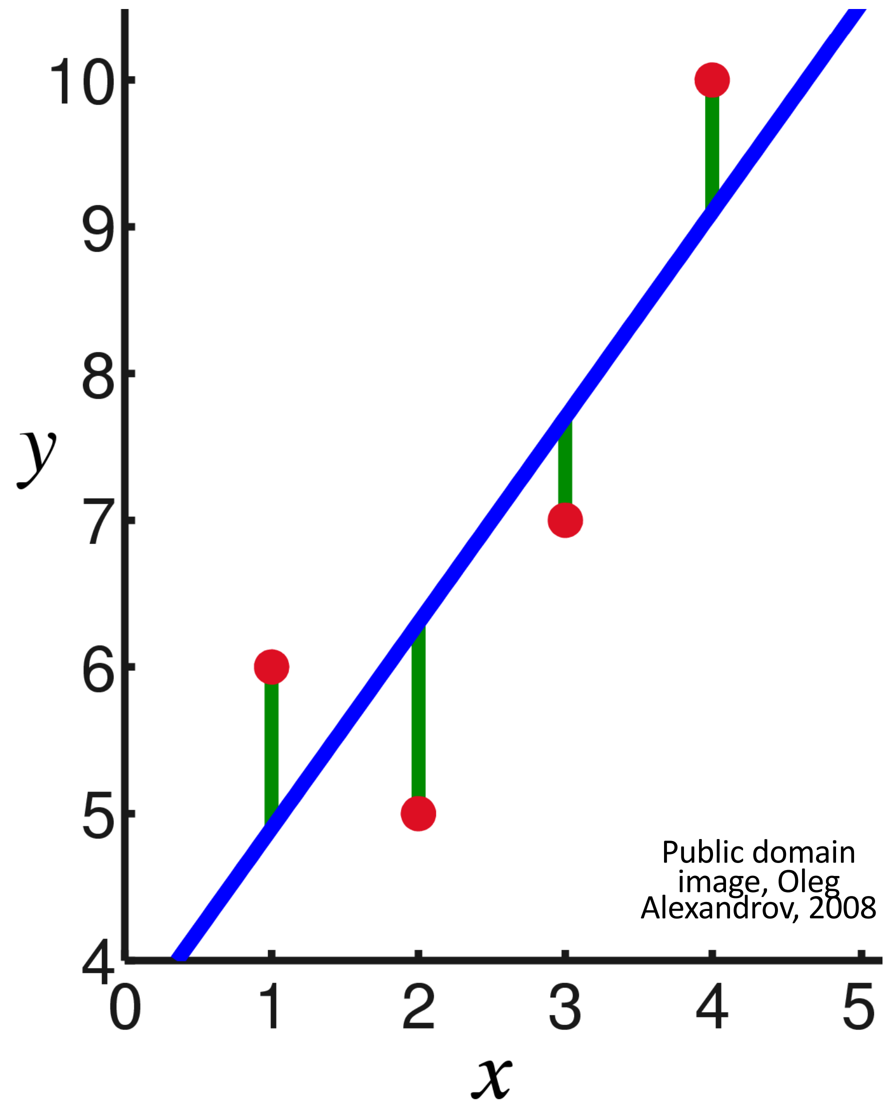
Linear regression. Public domain image, Seewaqu, 2010

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 $f(x) \approx y$?

- 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,
$$\epsilon = f(x) - y$$
- Some of them are positive, some are negative. What does it mean to "minimize" them?

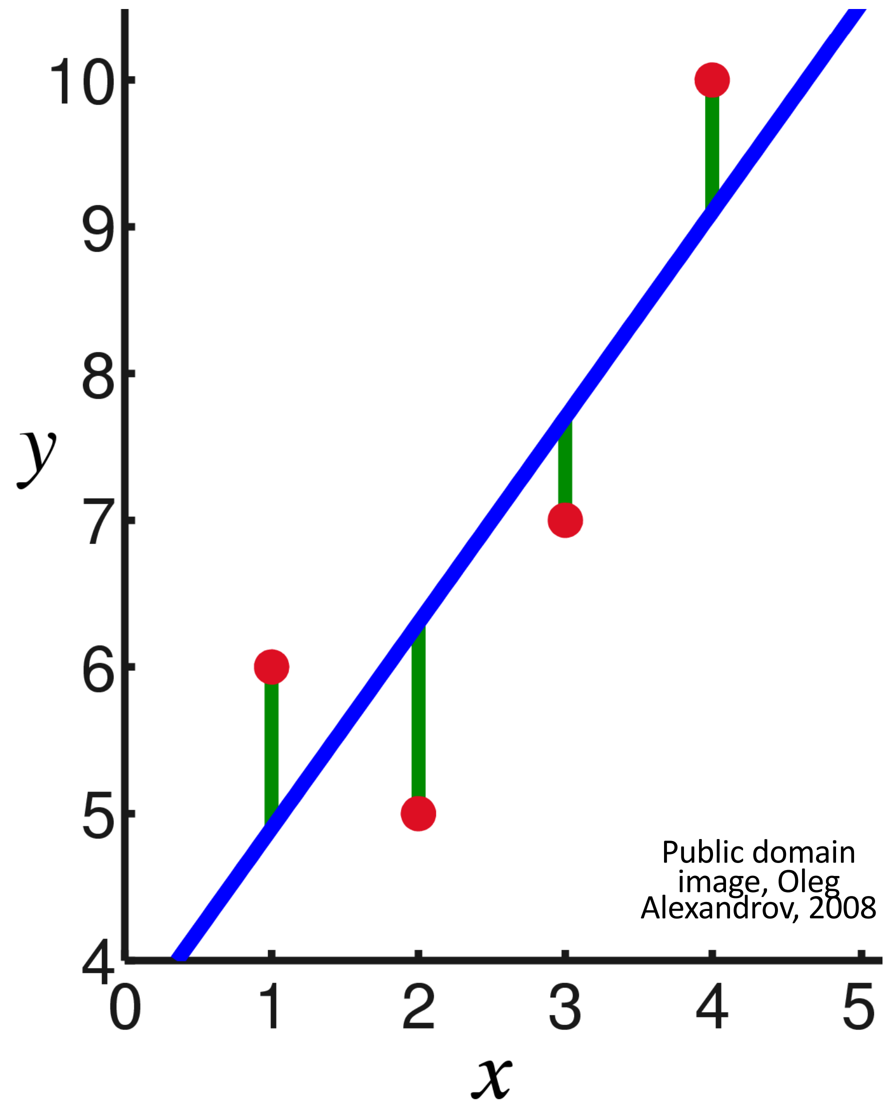


First: count the training tokens

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

$$x_i = \begin{bmatrix} x_{i,0} \\ \vdots \\ x_{i,D-1} \end{bmatrix}$$

$$f(x_i) = w@x_i + b = b + \sum_{j=0}^{D-1} x_{i,j} w_j$$

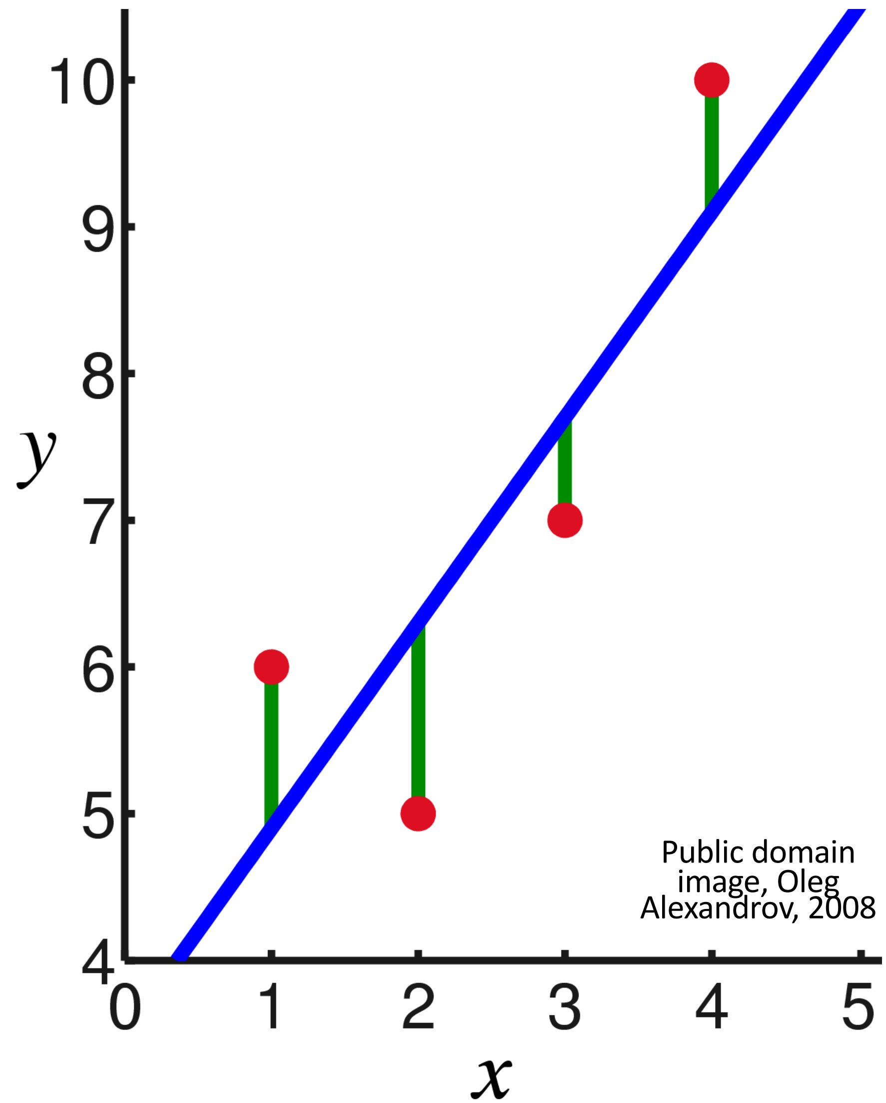


Training token errors

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

$$\epsilon_i = f(x_i) - y_i$$

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



Mean-squared error

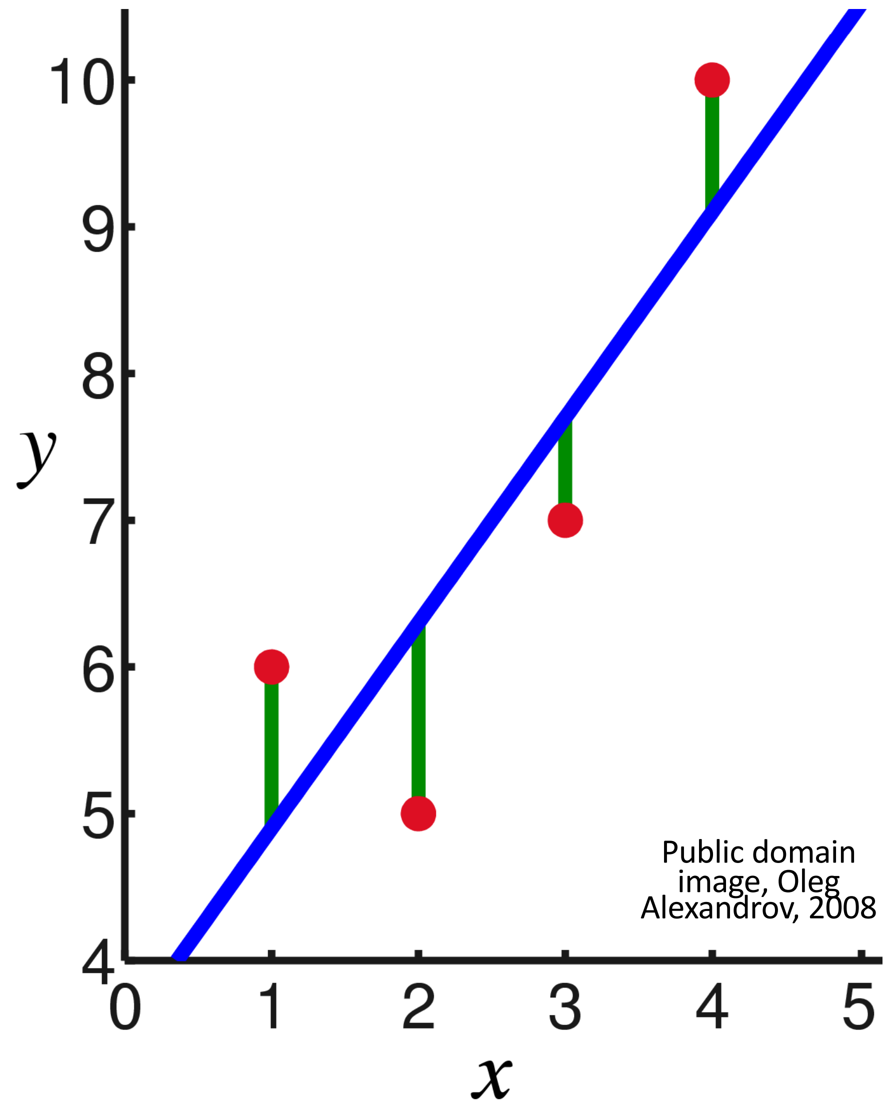
One useful criterion (not the only useful criterion, but perhaps the most common) of "minimizing the error" is to minimize the mean squared error:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\epsilon_i^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}(w@x_i + b - y_i)^2$$

Literally,

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



Public domain image, Oleg Alexandrov, 2008

Outline

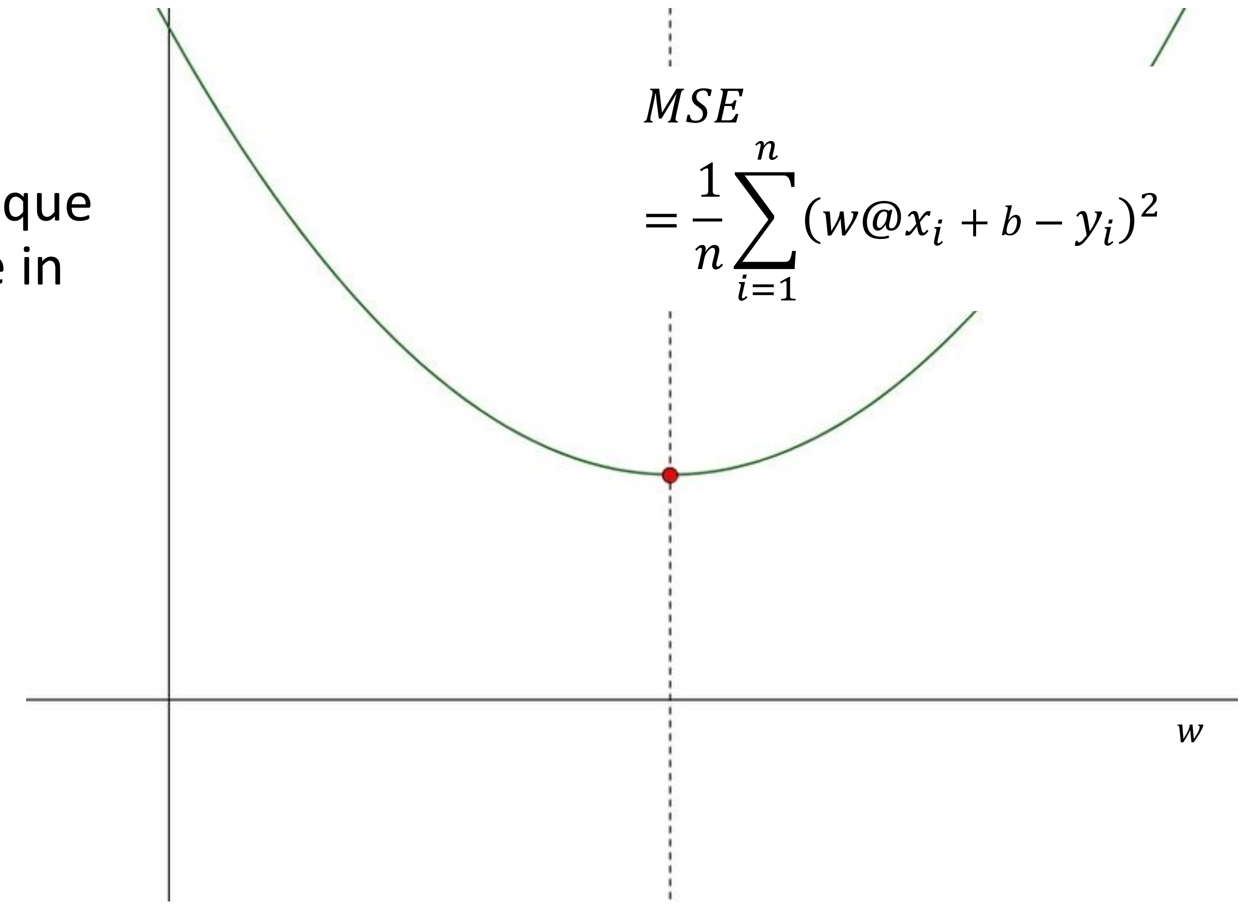
- Pythonic notation for vectors and matrices
- Definition of linear regression
- Mean-squared error
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

MSE = Parabola

Notice that MSE is a parabola in terms of b and w.

Since it's a parabola, it has a unique minimum that you can compute in closed form!

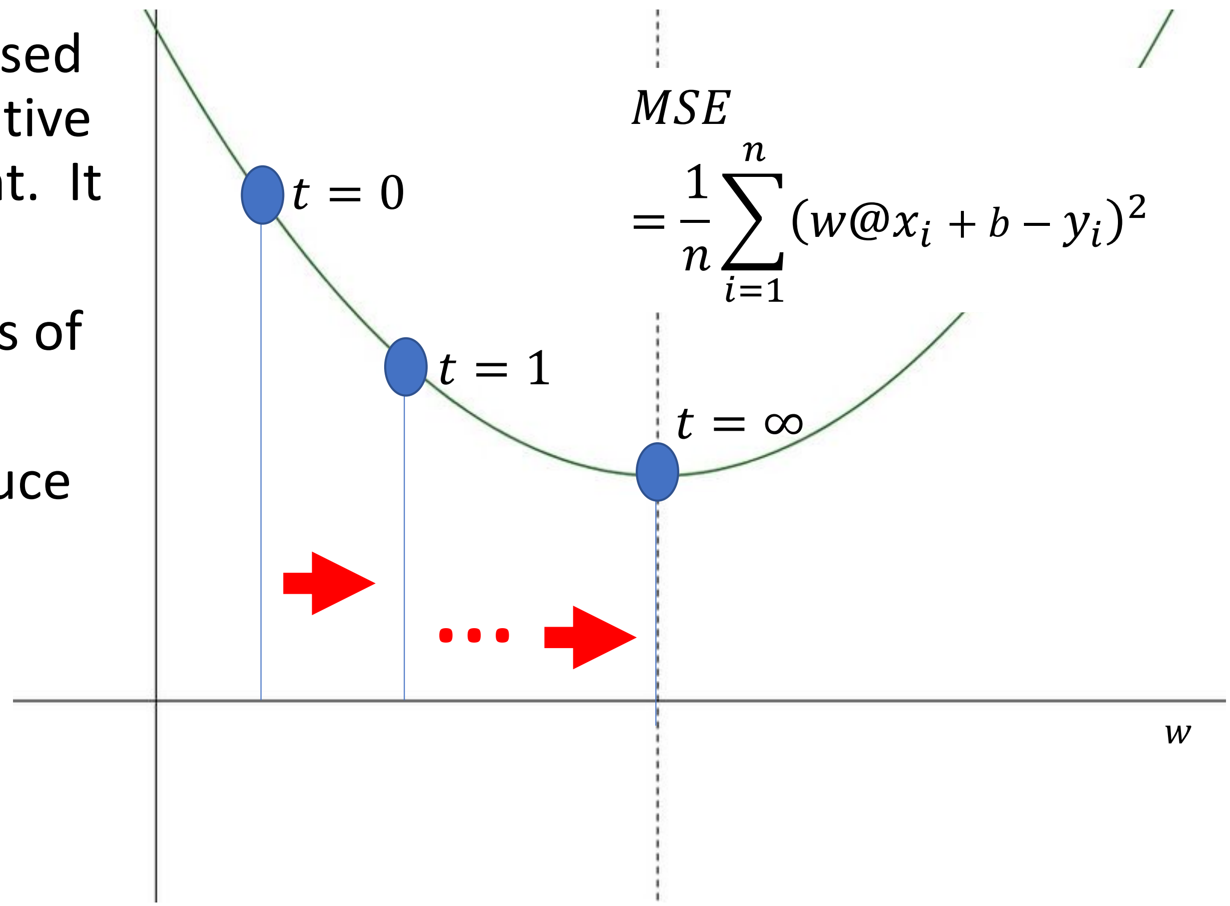
We won't do that today.



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 from random initial values of w and b (at $t = 0$).
- Adjust w and b in order to reduce MSE ($t = 1$).
- Repeat until you reach the optimum ($t = \infty$).



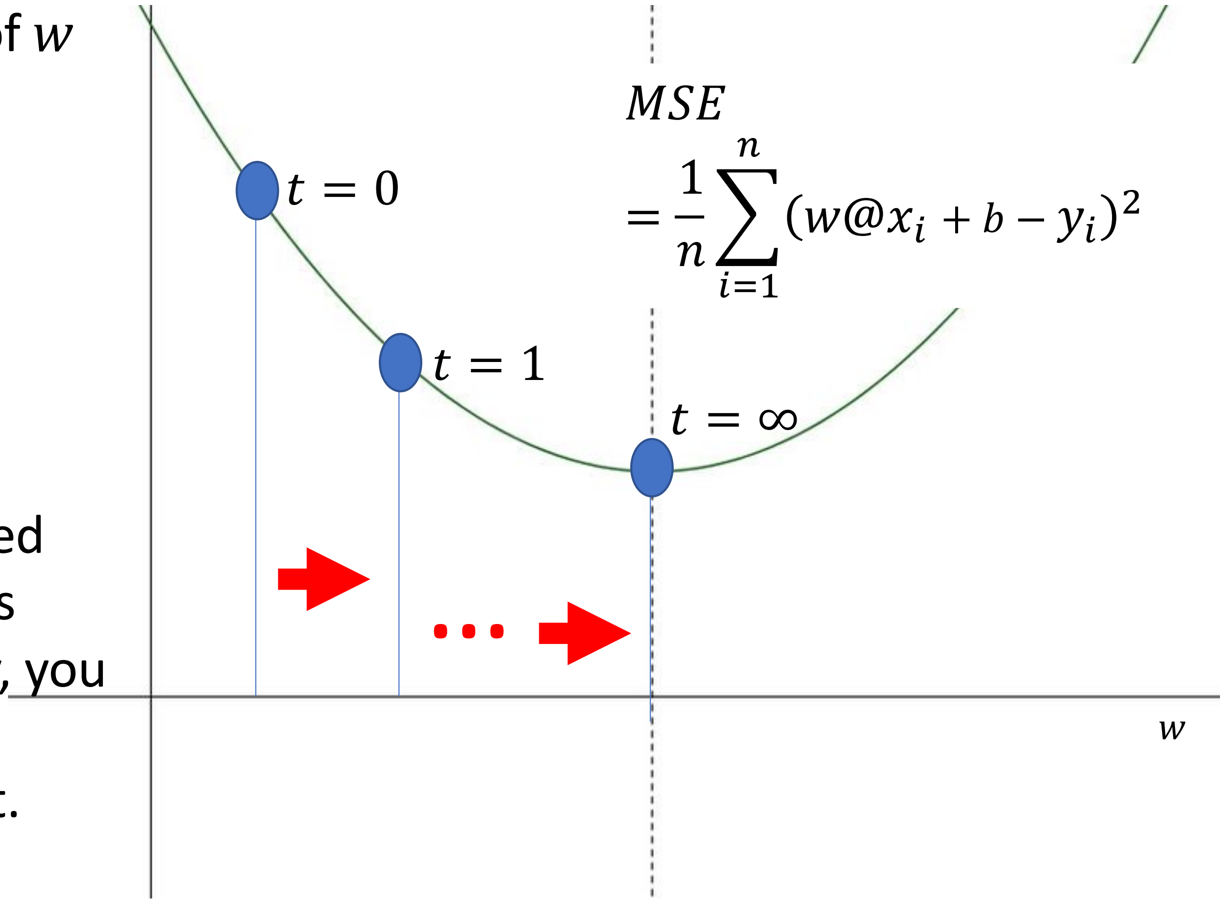
The iterative solution to linear regression

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

$$w \leftarrow w - \frac{\eta}{2} \nabla_w MSE$$

$$b \leftarrow b - \frac{\eta}{2} \frac{\partial MSE}{\partial b}$$

…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

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^2$$

To find the gradient, we use the chain rule of calculus. Remember that $\epsilon_i = wx_i + b - y_i$, and therefore

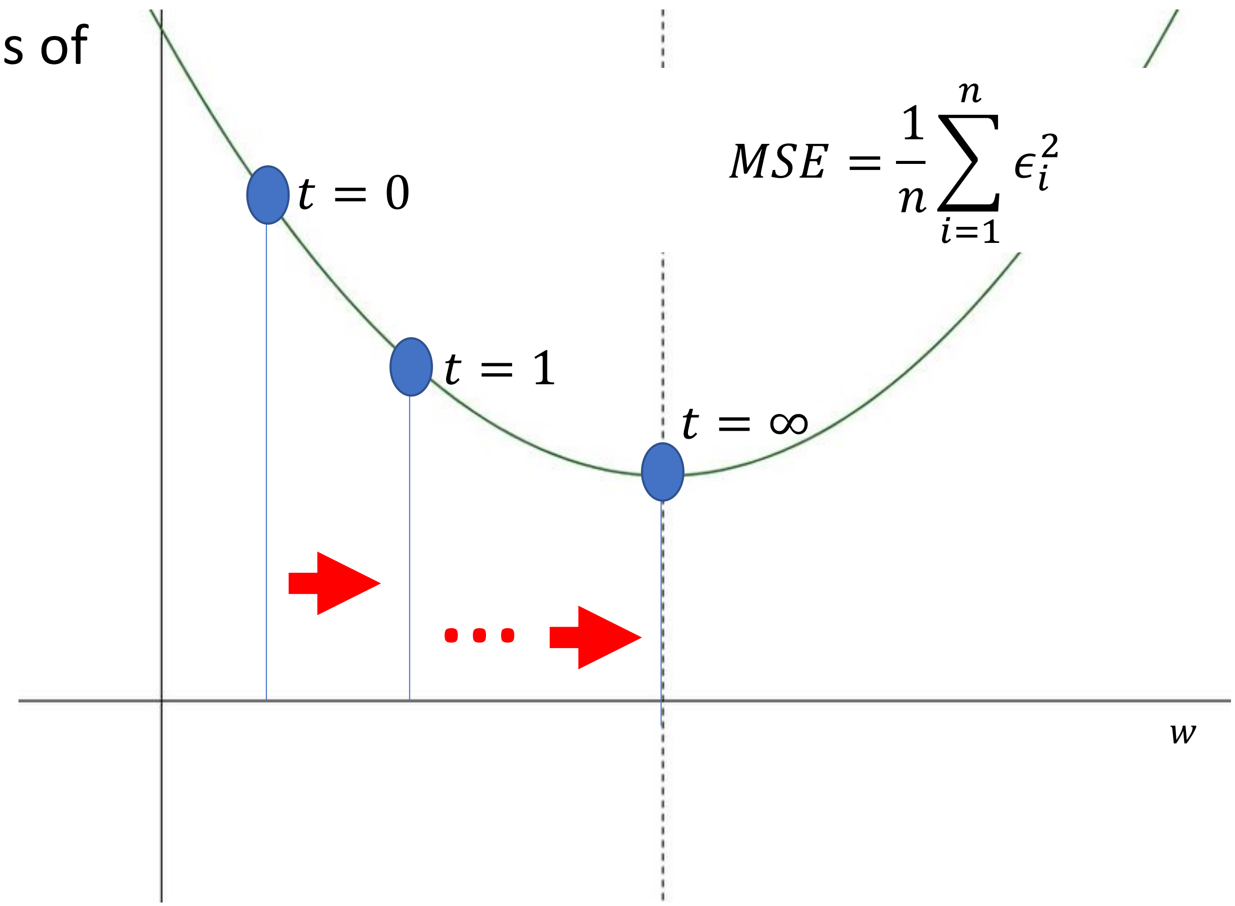
$$\nabla_w MSE = \frac{1}{n}\sum_{i=1}^{n} 2\epsilon_i \nabla_w \epsilon_i = \frac{2}{n}\sum_{i=1}^{n} \epsilon_i x_i$$

The iterative solution to linear regression

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

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

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



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 (x_i, y_i) at random (”stochastically”), and adjusts w in order to reduce the error a little bit for that one token:

$$w \leftarrow w - \frac{\eta}{2} \nabla_w \epsilon_i^2$$

…where

$$\epsilon_i^2 = (w@x_i + b - y_i)^2$$

Stochastic gradient descent

$$\epsilon_i^2 = (w@x_i + b - y_i)^2$$

If we differentiate that, we discover tha

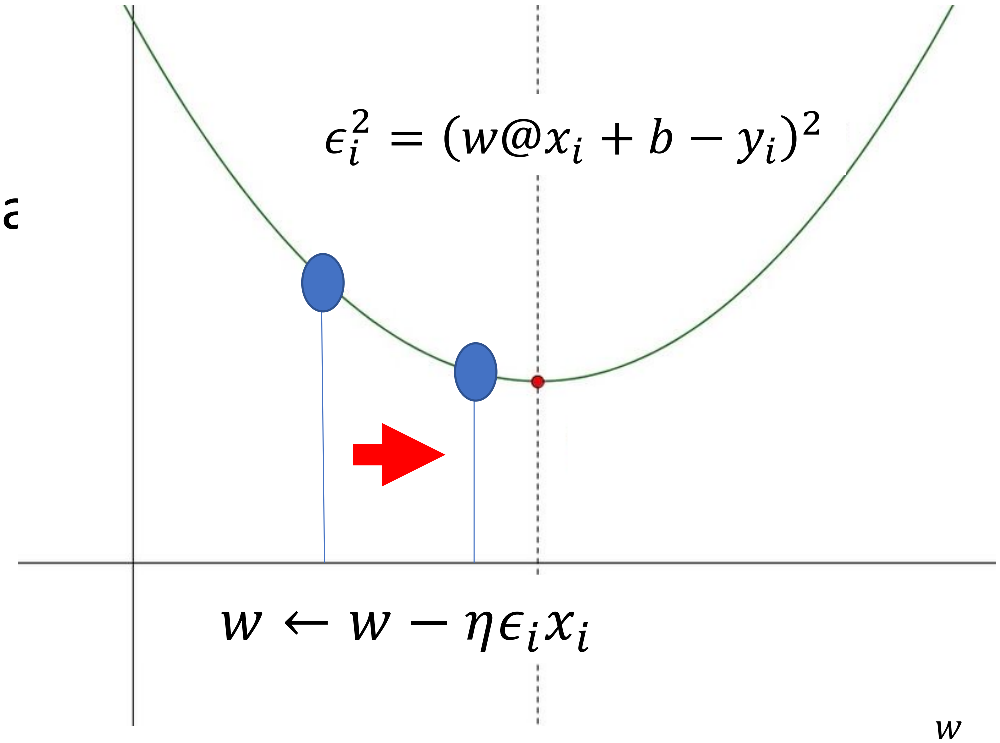
$$\nabla_w \epsilon_i^2 = 2\epsilon_i x_i$$

$$\frac{\partial \epsilon_i^2}{\partial b} = 2\epsilon_i$$

So the stochastic gradient descent algorithm is:

$$w \leftarrow w - \eta\epsilon_i x_i$$

$$b \leftarrow b - \eta\epsilon_i$$



The Stochastic Gradient Descent Algorithm

1. Choose a sample (x_i, y_i) at random from the training data
2. Compute the error of this sample, $\epsilon_i = w@x_i + b - y_i$
3. Adjust w in the direction opposite the error:

$$w \leftarrow w - \eta\epsilon_i x_i$$
$$b \leftarrow b - \eta\epsilon_i$$

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/129874/assessment/2329685, and 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

$$f(x) = b + w@x$$

- Mean-squared error

$$MSE = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^2$$

- Learning the solution: gradient descent

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

- Learning the solution: stochastic gradient descent

$$w \leftarrow w - \eta \epsilon_i x_i$$