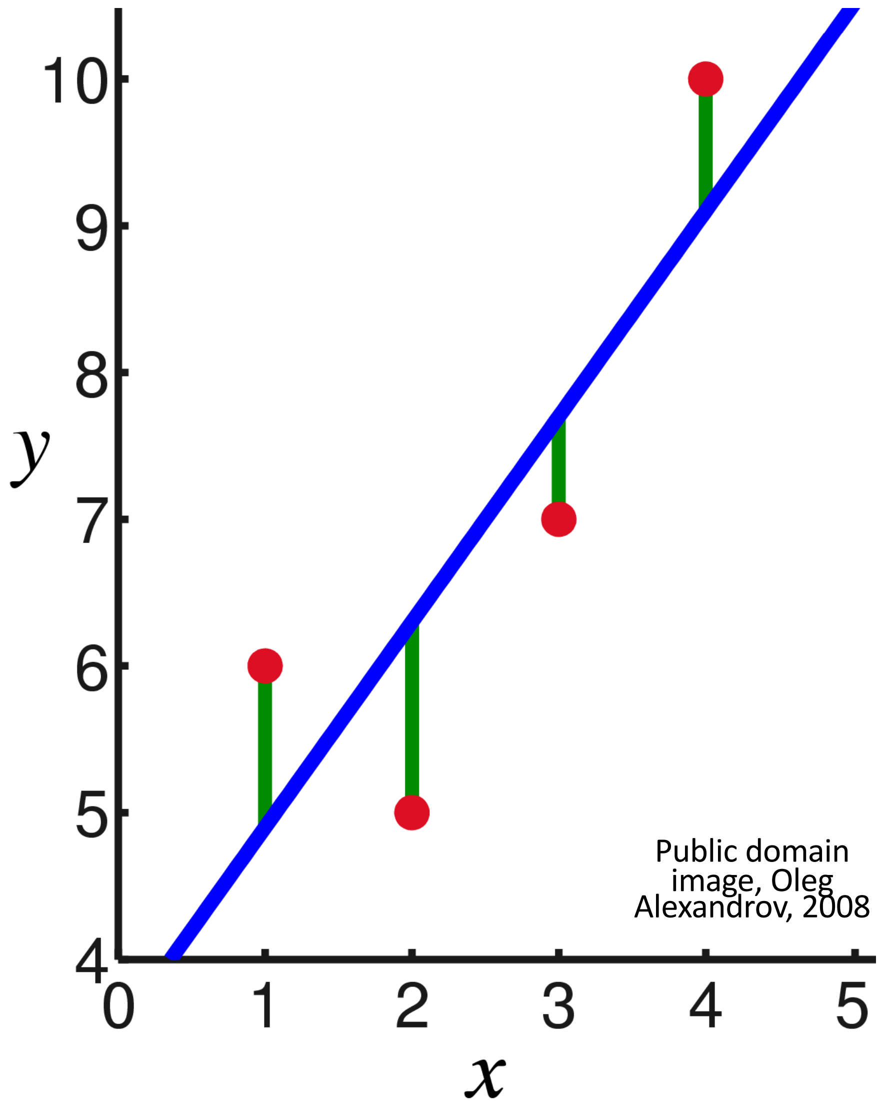
CS440/ECE448 Lecture 7: Linear Regression

Mark Hasegawa-Johnson, 2/2025



Public domain image, Oleg Alexandrov, 2008

Outline

- Linear regression
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

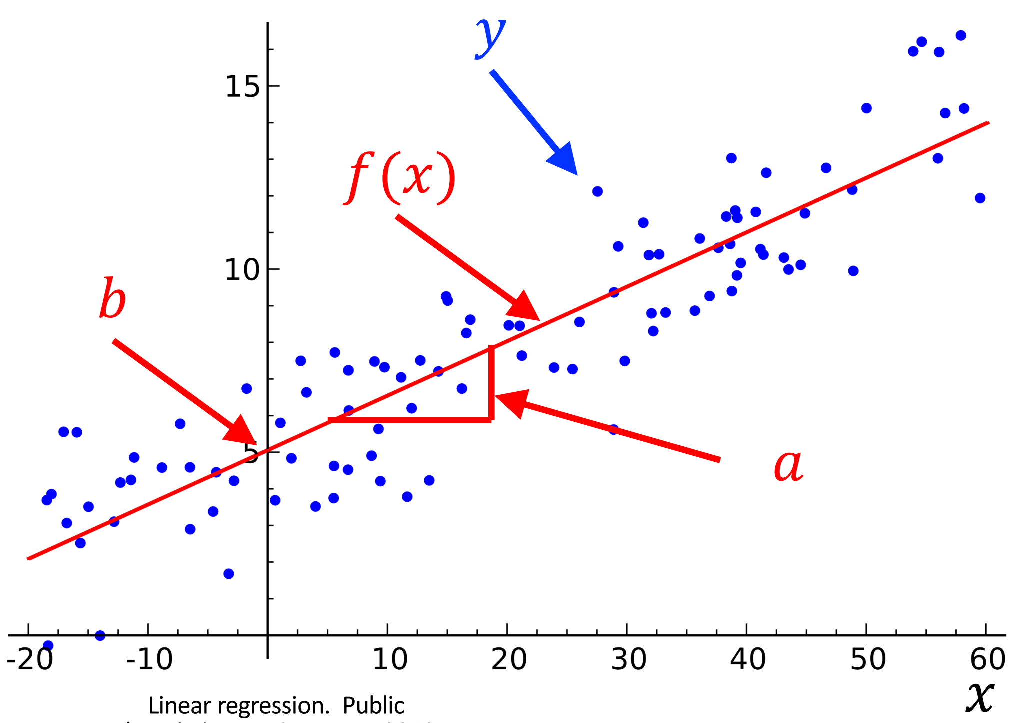
Linear regression

Linear regression is used to estimate a real-valued target variable, y, using a linear function of another variable, x:

$$f(x) = ax + b$$

… so that …

$$f(x) \approx y$$



Linear regression. Public domain image, Seewaqu, 2010

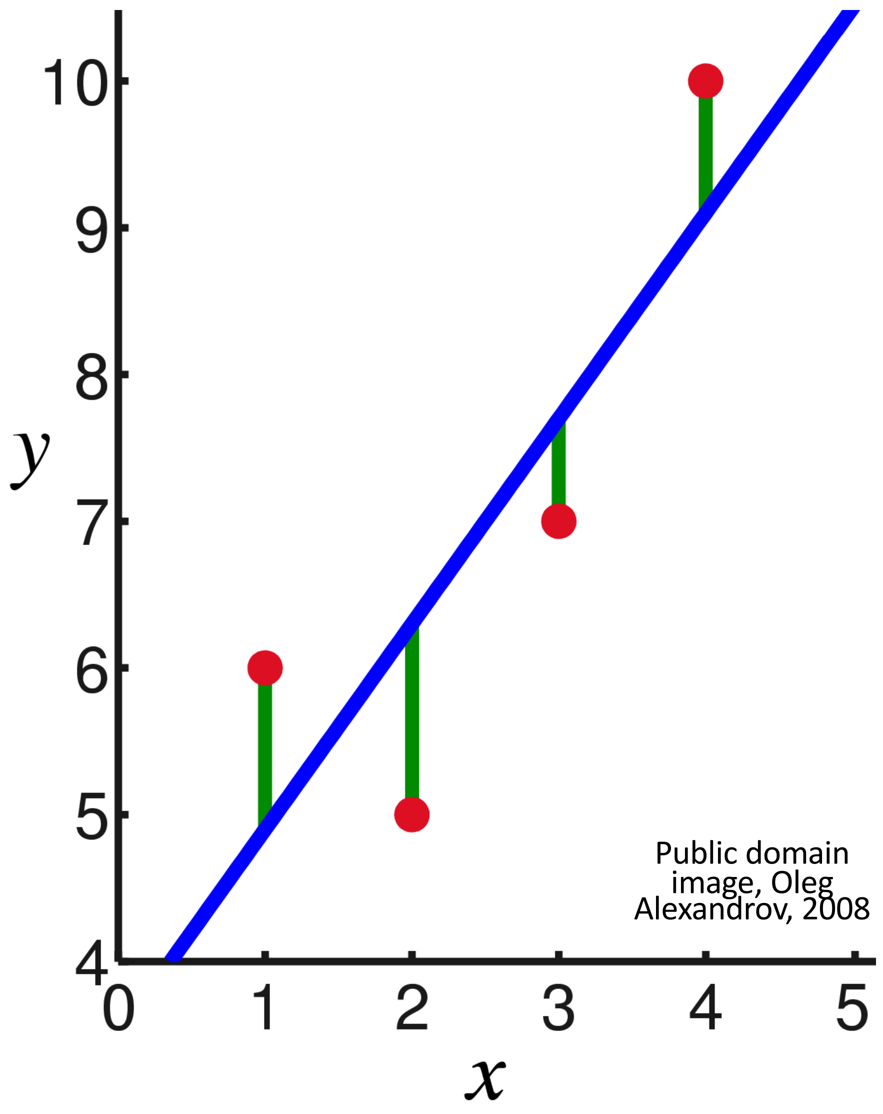
Error

Let's suppose we have n training tokens, x_1 through x_n.

$$f(x_i) = ax_i + b$$

The error is the difference between the actual output, $f(x_i)$, and the desired output, y_i. It's the green vertical bars in the figure at left.

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



Public domain image, Oleg Alexandrov, 2008

Mean squared error

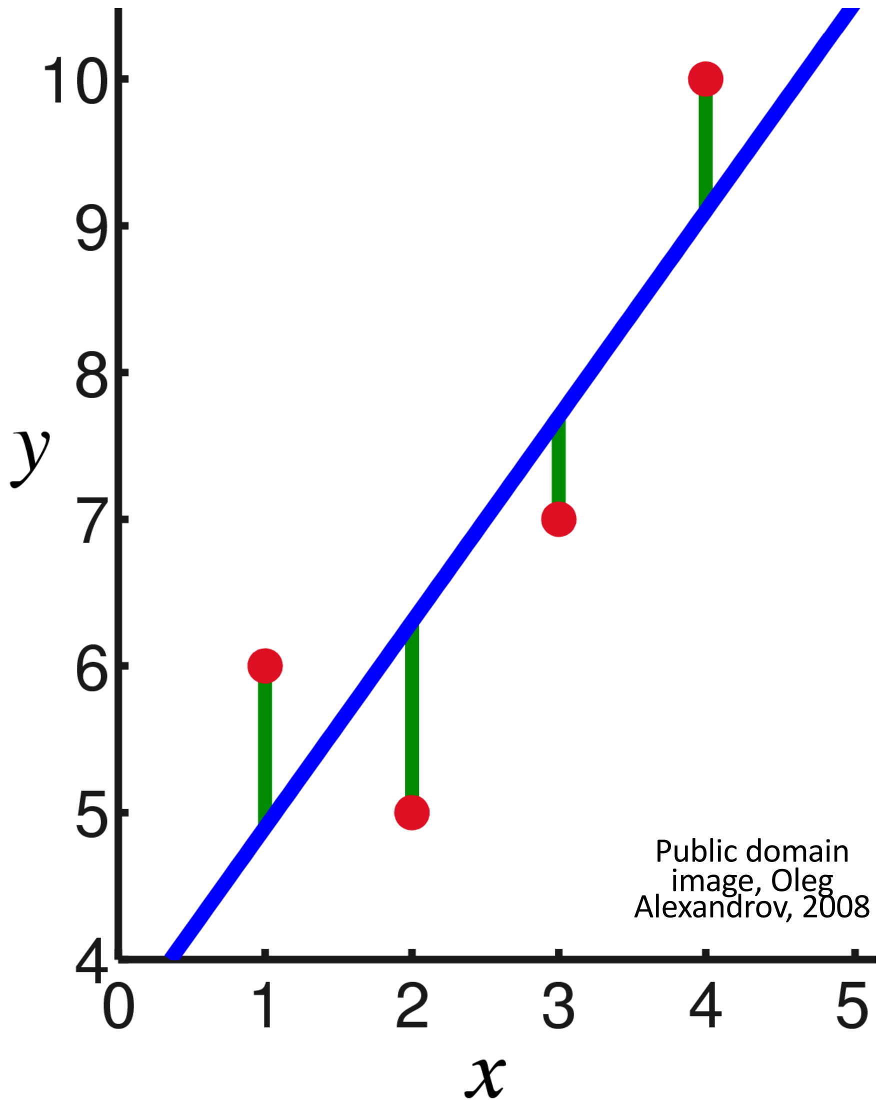
One useful criterion (not the only useful criterion, but perhaps the most common) of "minimizing the error" is to minimize the mean squared error:

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

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.



Public domain image, Oleg Alexandrov, 2008

Vector notation

Suppose we define a data matrix, a weight vector, and a target vector, like this:

$$\boldsymbol{X} = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix}, \qquad \boldsymbol{w} = \begin{bmatrix} a \\ b \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Then the predictions $f(x_i)$ are given in the vector $\boldsymbol{f}(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{w}$, and the errors are given in the vector $\boldsymbol{\epsilon} = \boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}$, and the mean squared error is

$$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(ax_i + b - y_i)^2 = \frac{1}{2n}(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Linear regression: The closed-form solution

$$\mathcal{L} = \frac{1}{2n}(\boldsymbol{Xw} - \boldsymbol{y})^T(\boldsymbol{Xw} - \boldsymbol{y})$$

We can minimize $\mathcal{L}$ by setting its derivative to zero:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{n}\boldsymbol{X}^T(\boldsymbol{Xw} - \boldsymbol{y}) = 0$$

Solving, we get:

$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & n \end{bmatrix}^{-1} \begin{bmatrix} \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} y_i \end{bmatrix}$$

Outline

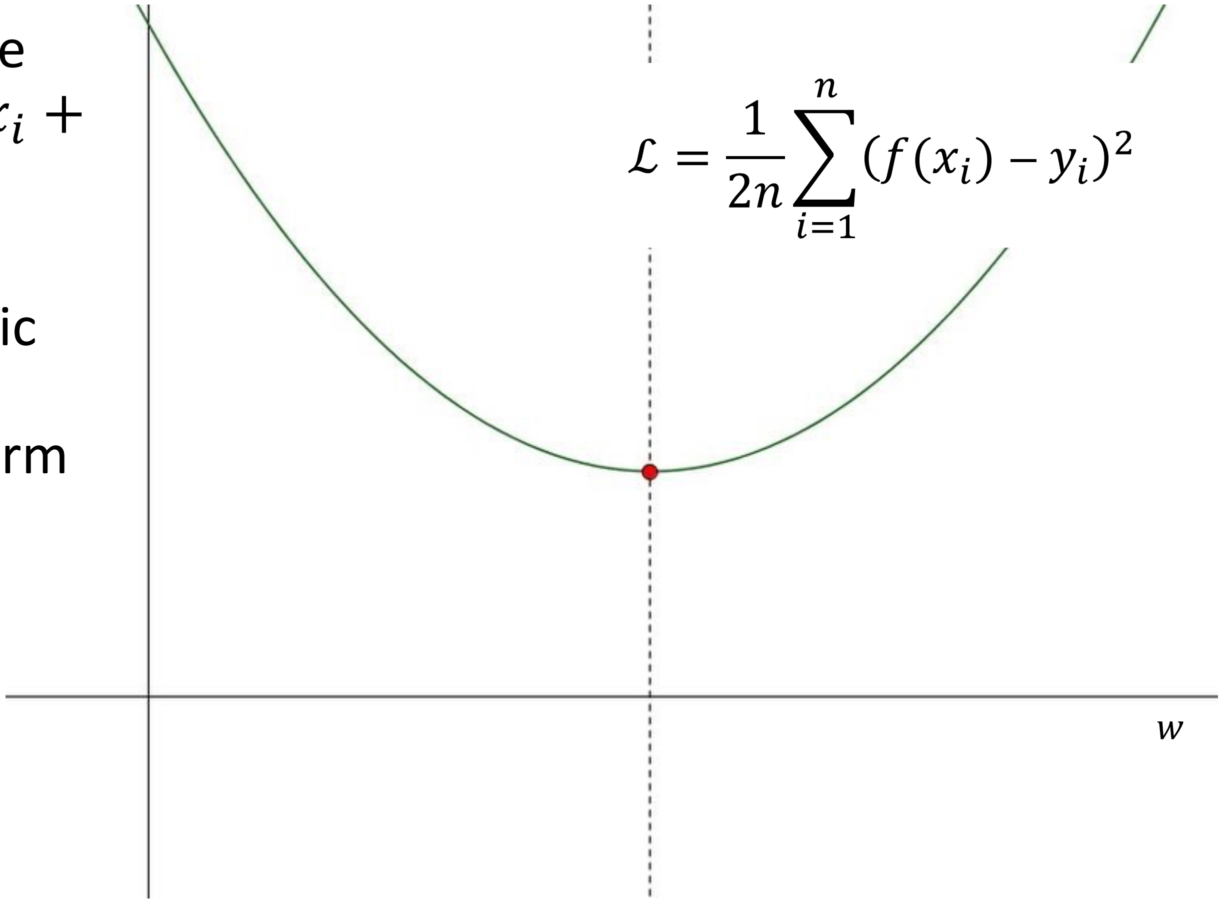
- Linear regression
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

MSE = Parabola

Notice that MSE is a non-negative quadratic function of $f(x_i) = ax_i + b$, therefore it's a non-negative quadratic function of a and b.

Since it's a non-negative quadratic function of $\boldsymbol{w}$, it has a unique minimum, given by the closed-form solution

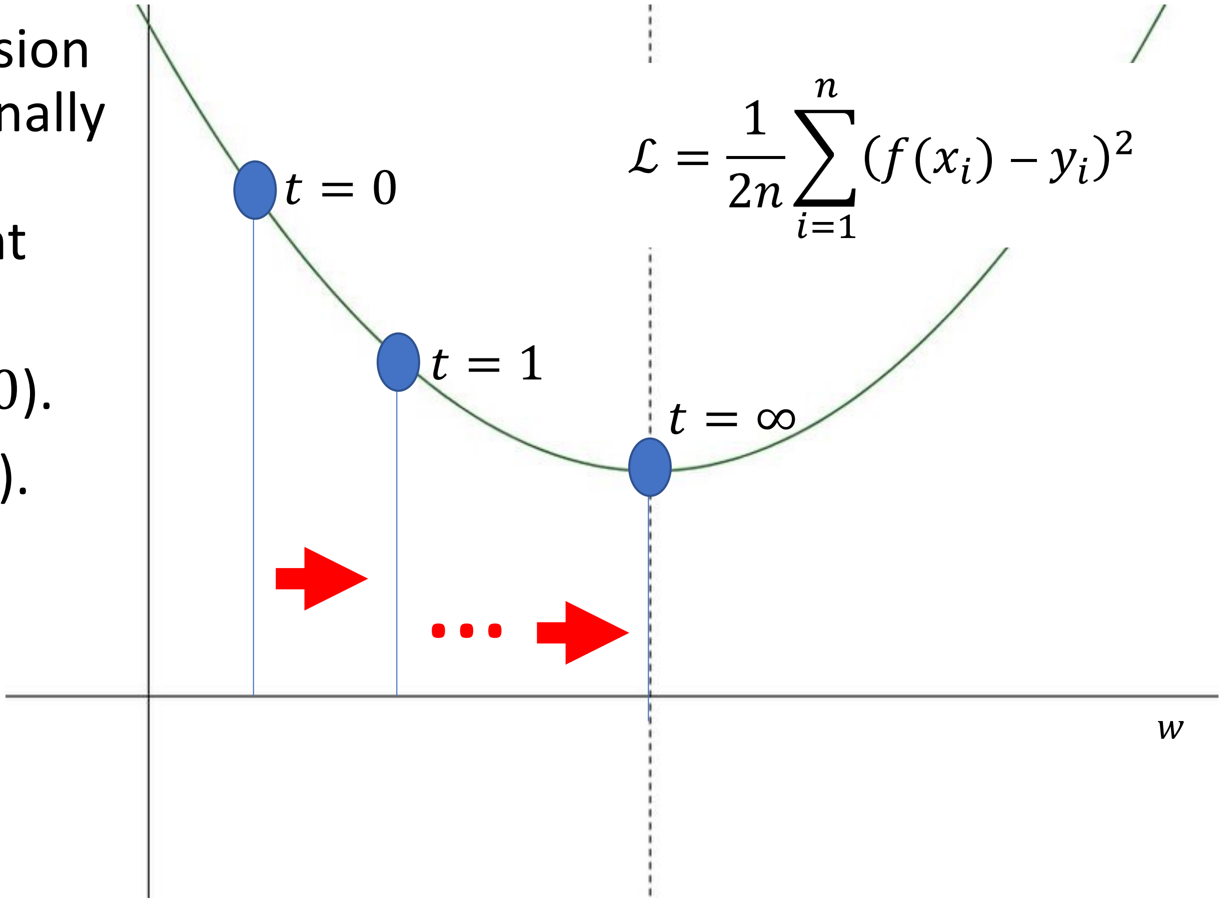
$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$



The iterative solution to linear regression

Sometimes, solving linear regression in closed form is too computationally expensive, so instead we use an iterative algorithm called gradient descent. It works like this:

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

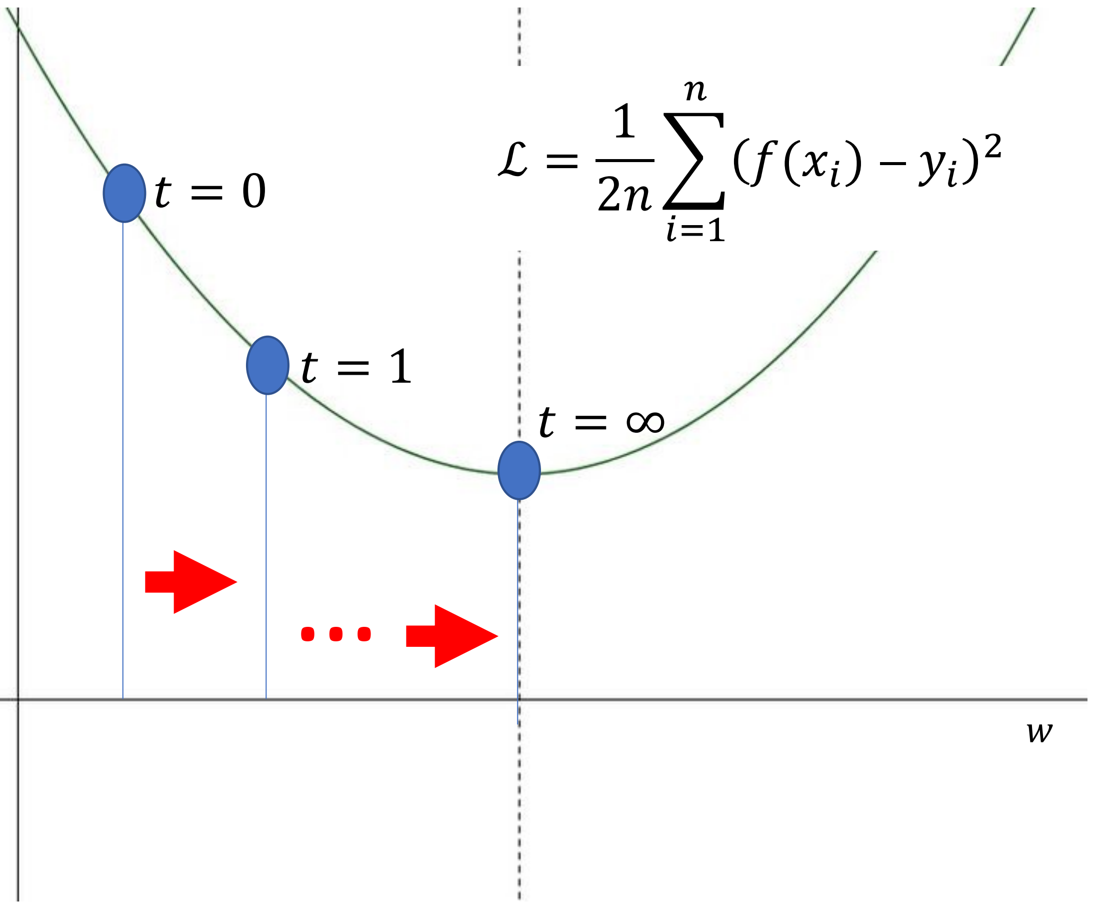


The iterative solution to linear regression

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

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

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



Finding the gradient

We’ve already calculated the gradient! It’s

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{n} \boldsymbol{X}^T (\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$$

Sometimes finding the gradient is still too computationally expensive. In that case we use stochastic gradient descent.

Outline

- Definition of linear regression
- Learning the solution: gradient descent
- Learning the solution: stochastic gradient descent

Stochastic gradient descent

The general idea of stochastic gradient descent is to break up the n-sample problem into a lot of tiny 1-sample problems, like this:

$$\boldsymbol{x}_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}, \qquad \boldsymbol{f}(\boldsymbol{X}) = \begin{bmatrix} f(\boldsymbol{x}_1) \\ \vdots \\ f(\boldsymbol{x}_n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix} \boldsymbol{w} = \begin{bmatrix} \boldsymbol{x}_1^T \boldsymbol{w} \\ \vdots \\ \boldsymbol{x}_n^T \boldsymbol{w} \end{bmatrix}$$

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_i, \qquad \mathcal{L}_i = \frac{1}{2} \left(\boldsymbol{x}_i^T \boldsymbol{w} - y_i \right)^2$$

Stochastic gradient descent

The general idea of stochastic gradient descent is:

1. Choose one training token $(\boldsymbol{x}_i, y_i)$ at random (”stochastically”), and adjusts $\boldsymbol{w}$ to reduce the loss just for that one token:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{w}} = \boldsymbol{w} - \eta \frac{\partial}{\partial \boldsymbol{w}} \left(\frac{1}{2} \left(\boldsymbol{x}_i^T \boldsymbol{w} - y_i \right)^2 \right)$$

2. Repeat.

Stochastic gradient descent

The reason this is useful is because, for each training token, the error is a scalar:

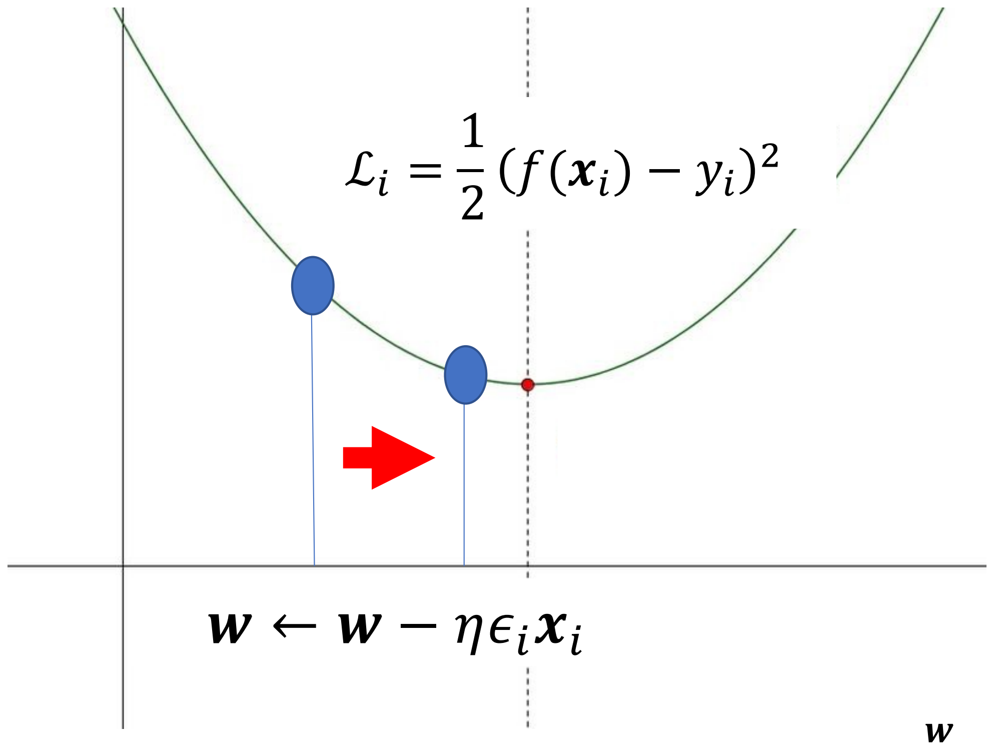
$$\mathcal{L}_i = \frac{1}{2}\epsilon_i^2, \qquad \epsilon_i = \boldsymbol{w}^T\boldsymbol{x}_i - y_i$$

… so the derivative can be computed very efficiently:

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

Stochastic gradient descent

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



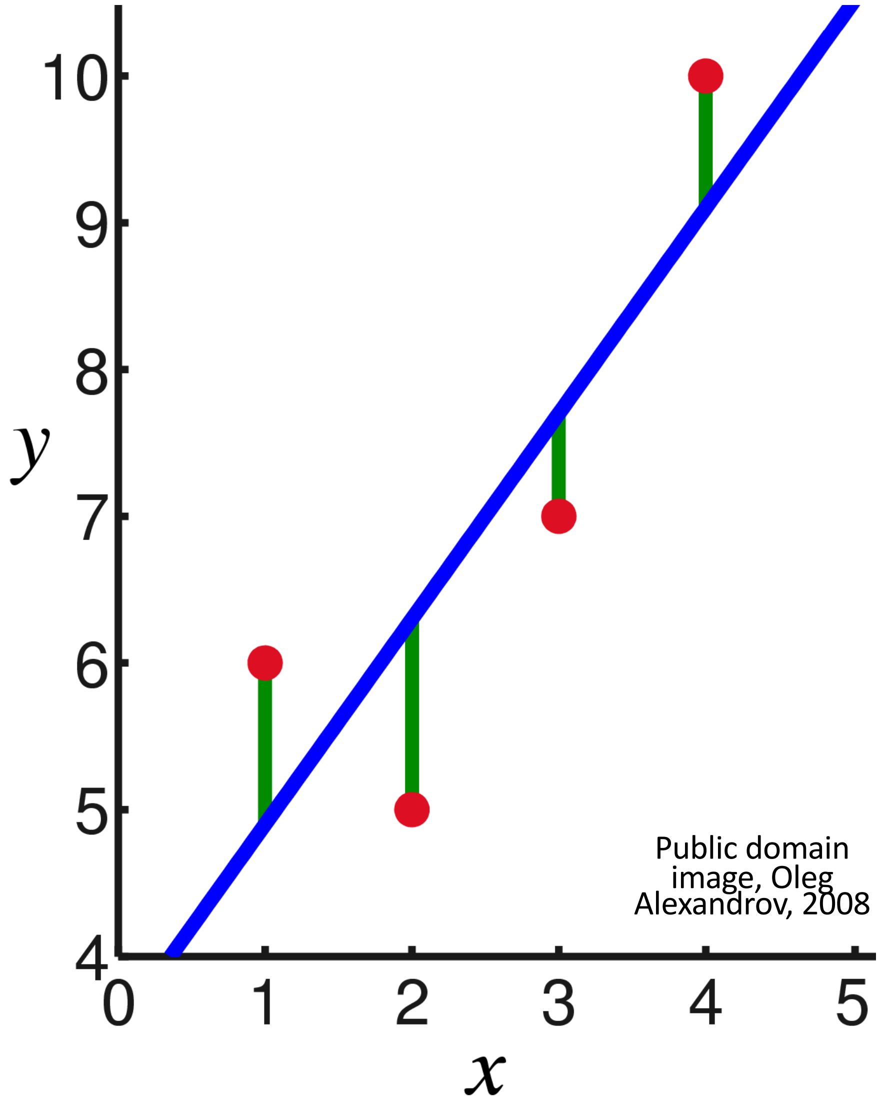
$\mathcal{L}_i = \frac{1}{2}(f(\boldsymbol{x}_i) - y_i)^2$
$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \epsilon_i \boldsymbol{x}_i$
$\boldsymbol{w}$

Intuition:

- Notice the sign:

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

- If ϵ_i is positive ($f(x_i) - y_i > 0$), then we want to **reduce** $f(x_i)$
- If ϵ_i is negative ($f(x_i) - y_i < 0$), then we want to **increase** $f(x_i)$



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, $\epsilon_i = \boldsymbol{w}^T \boldsymbol{x}_i - y_i$
3. Adjust $\boldsymbol{w}$ in the direction opposite the error:
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \epsilon_i \boldsymbol{x}_i$$
4. If the error is still too large, go to step 1. If the error is small enough, stop.

Today’s Quiz

Go to prairielearn, 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

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

- Mean-squared error

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

- 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

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