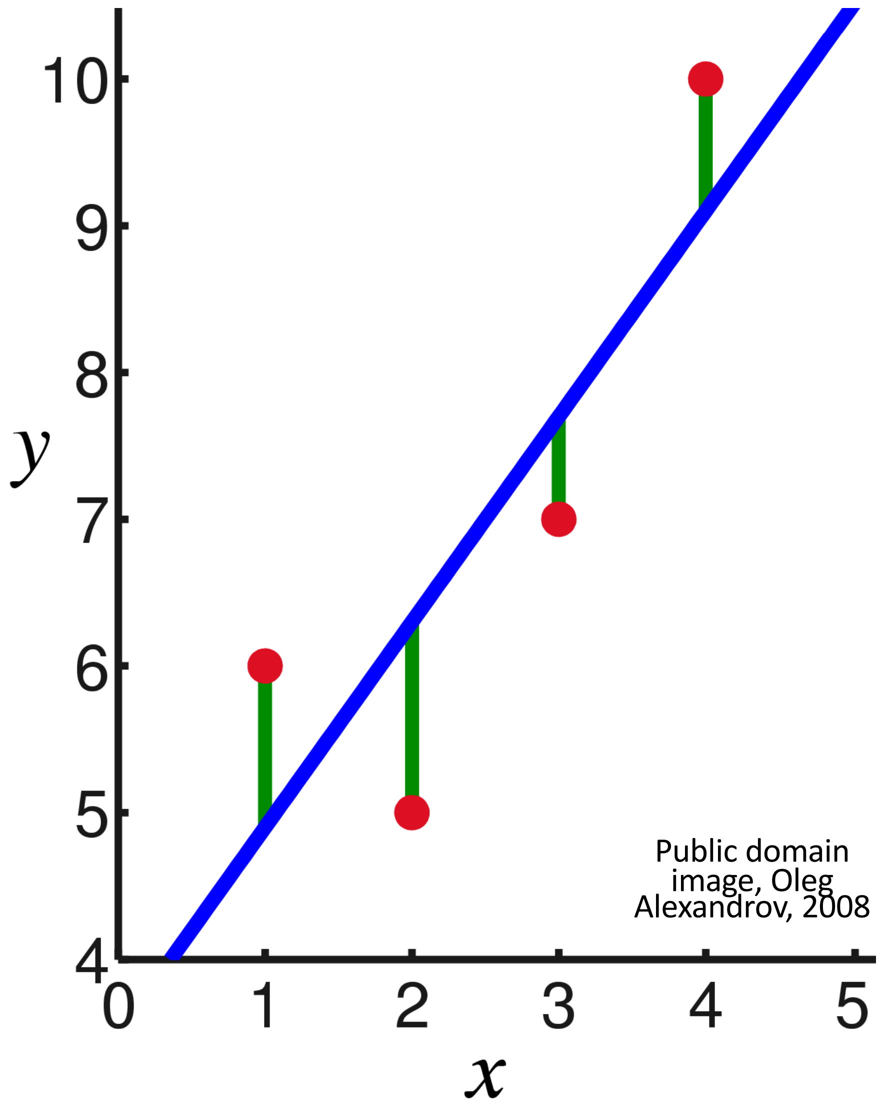
CS440/ECE448 Lecture 9: Linear Regression

Mark Hasegawa-Johnson

These lecture slides are in the public domain. Re-use, remix, or redistribute at will.



Public domain image, Oleg Alexandrov, 2008

Outline

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

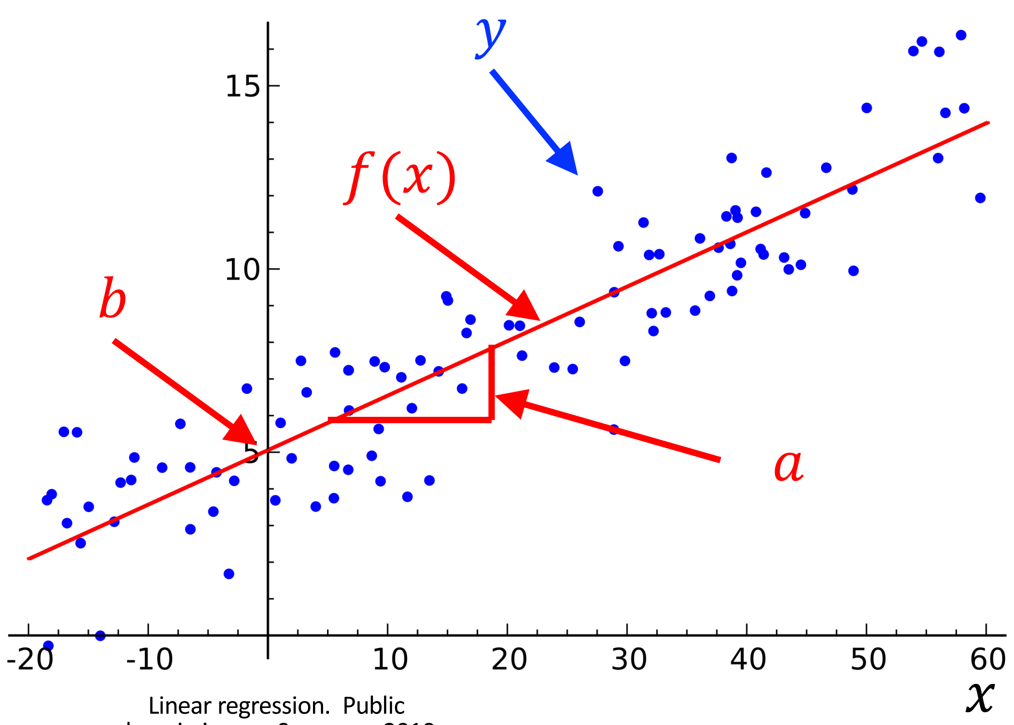
Linear regression

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

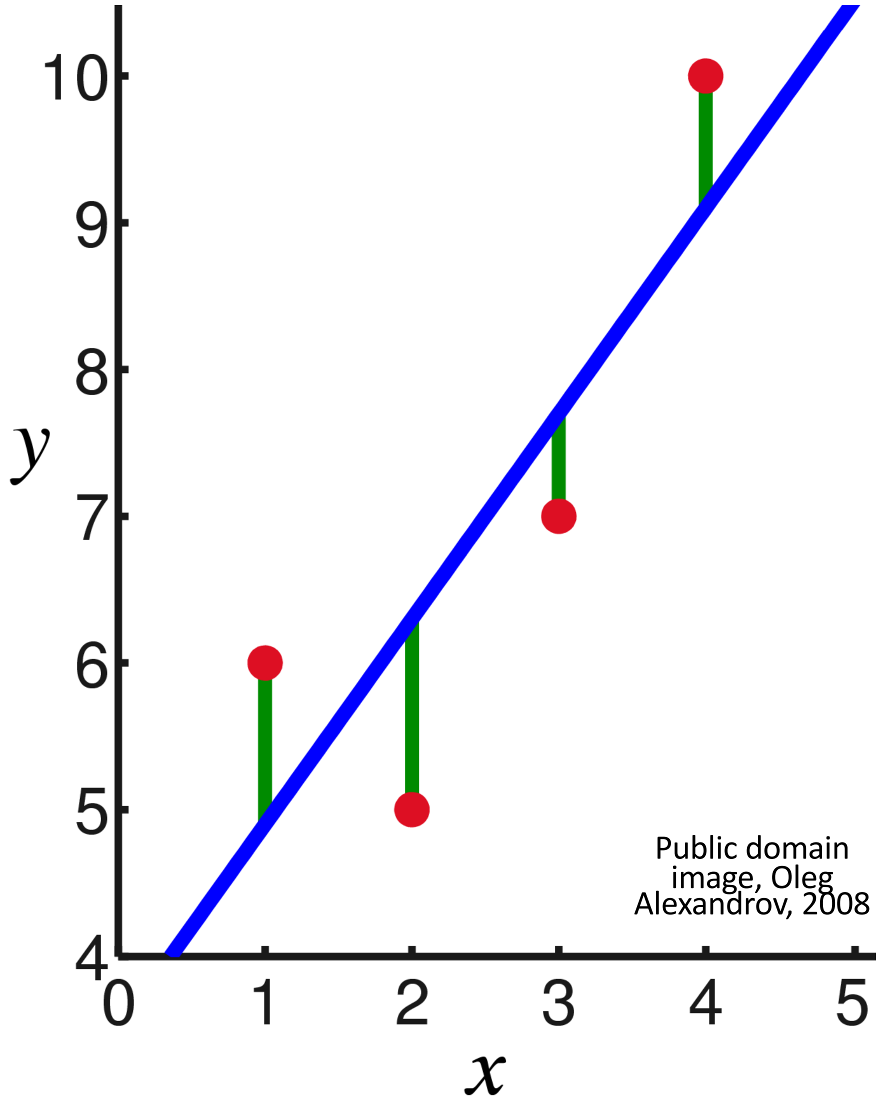
Error

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.

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



Mean squared error

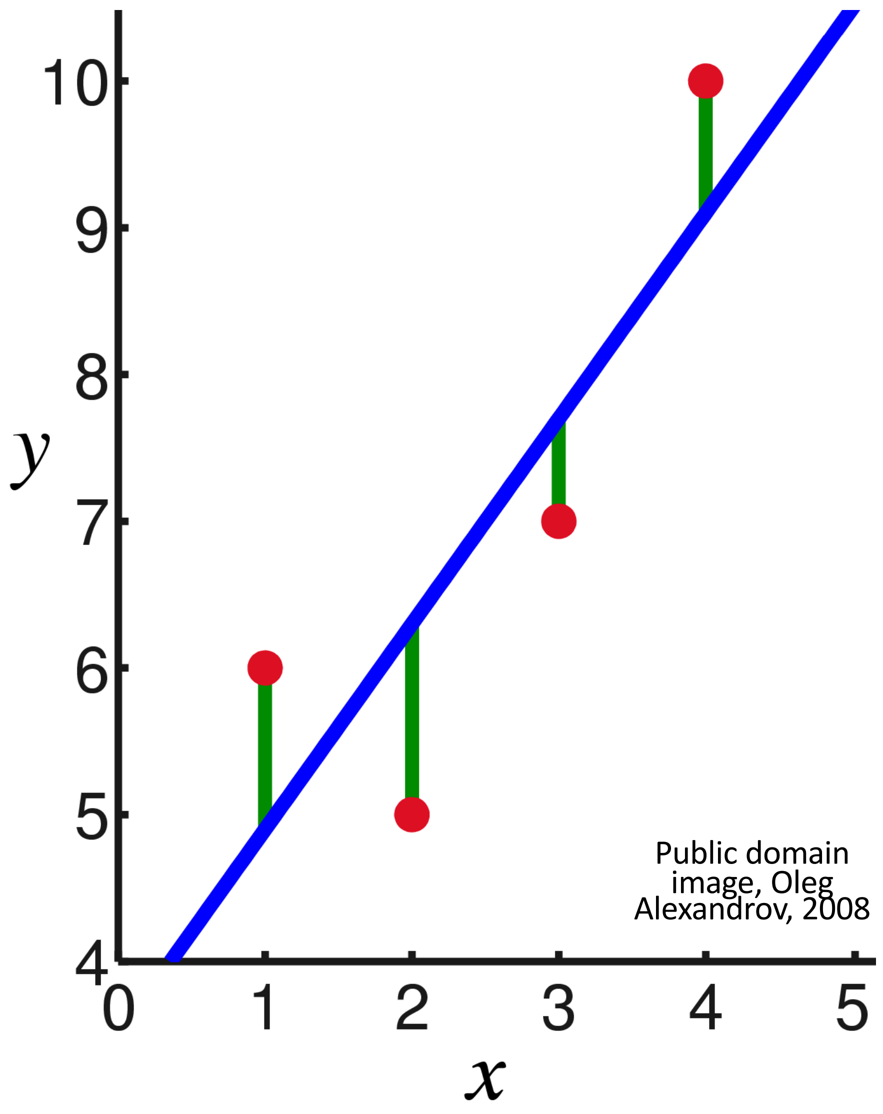
One useful criterion (not the only useful criterion, but perhaps the most common) of “minimizing the error” is to minimize the mean squared error:

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

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

Outline

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

Vector notation

In the general case, we will have vector inputs, $\boldsymbol{x}_i = \left[x_{i,1}, \dots, x_{i,m}\right]^T$, and a weight vector $\boldsymbol{w} = [w_1, \dots, w_m]^{\boldsymbol{T}}$, and the linear regression output will be

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i = \sum_{j=1}^{m} w_j x_{i,j}$$

Then the mean-squared error is

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

Vector notation special case: [slope,offset]

Notice that the scalar case is a special case of the vector notation. If we write $\boldsymbol{x}_i = [x_i, 1]^T$, and a weight vector $\boldsymbol{w} = [a, b]^{\boldsymbol{T}}$, then the linear regression output is:

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

MSE as a dot product

Incidentally, we can write the sum-squared-error as a dot product:

$$\sum_{i=1}^{n}(\boldsymbol{w}^T\boldsymbol{x}_i - y_i)^2 = [\boldsymbol{w}^T\boldsymbol{x}_1 - y_1, \dots, \boldsymbol{w}^T\boldsymbol{x}_n - y_n]\begin{bmatrix}\boldsymbol{x}_1^T\boldsymbol{w} - y_1\\ \vdots \\ \boldsymbol{x}_n^T\boldsymbol{w} - y_n\end{bmatrix}$$

$$= (\boldsymbol{w}^T[\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] - [y_1, \dots, y_n])\left(\begin{bmatrix}\boldsymbol{x}_1^T\\ \vdots \\ \boldsymbol{x}_n^T\end{bmatrix}\boldsymbol{w} - \begin{bmatrix}y_1\\ \vdots \\ y_n\end{bmatrix}\right)$$

MSE as a dot product

…so the MSE becomes…

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

…where…

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_n^T \end{bmatrix}, \boldsymbol{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

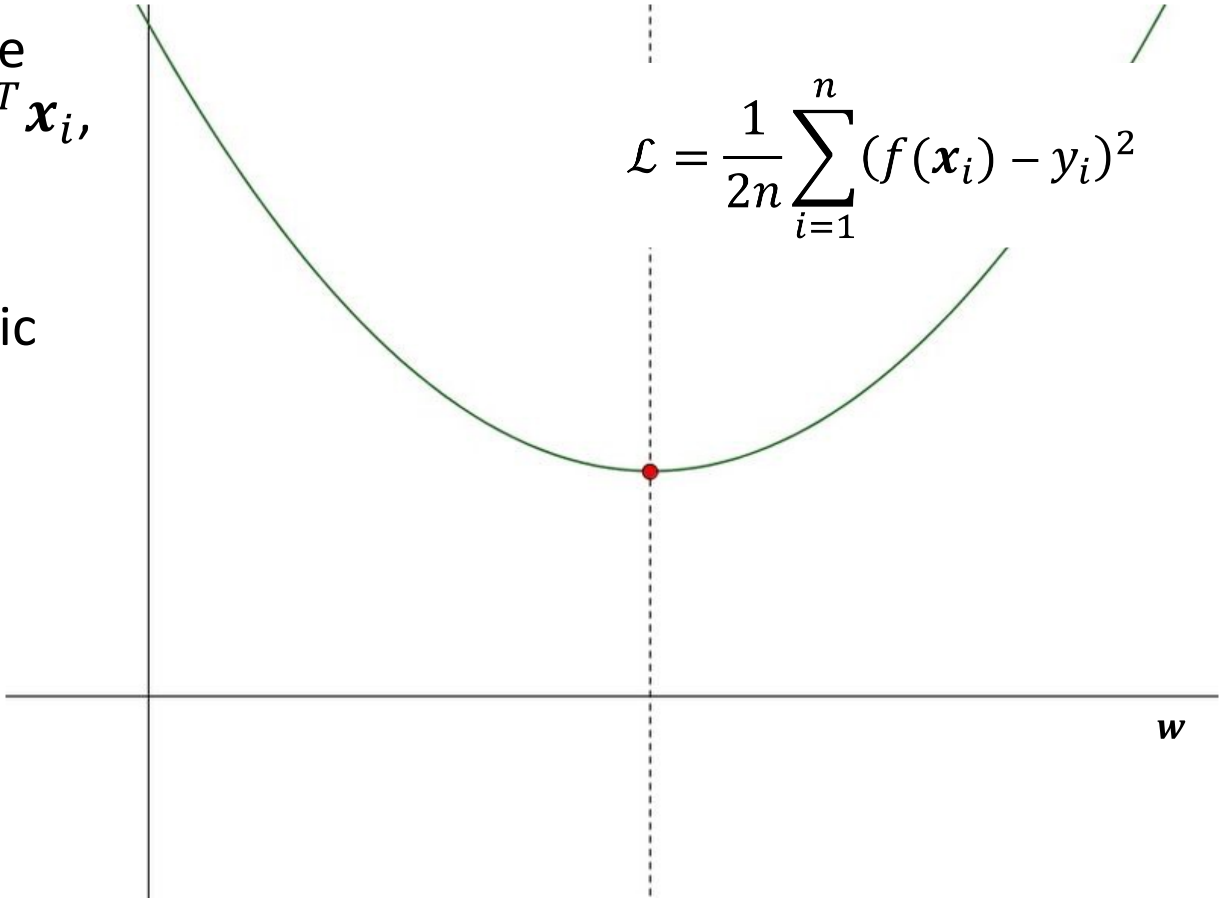
Outline

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

MSE = Parabola

Notice that MSE is a non-negative quadratic function of $f(\boldsymbol{x}_i) = \boldsymbol{w}^T \boldsymbol{x}_i$, 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.

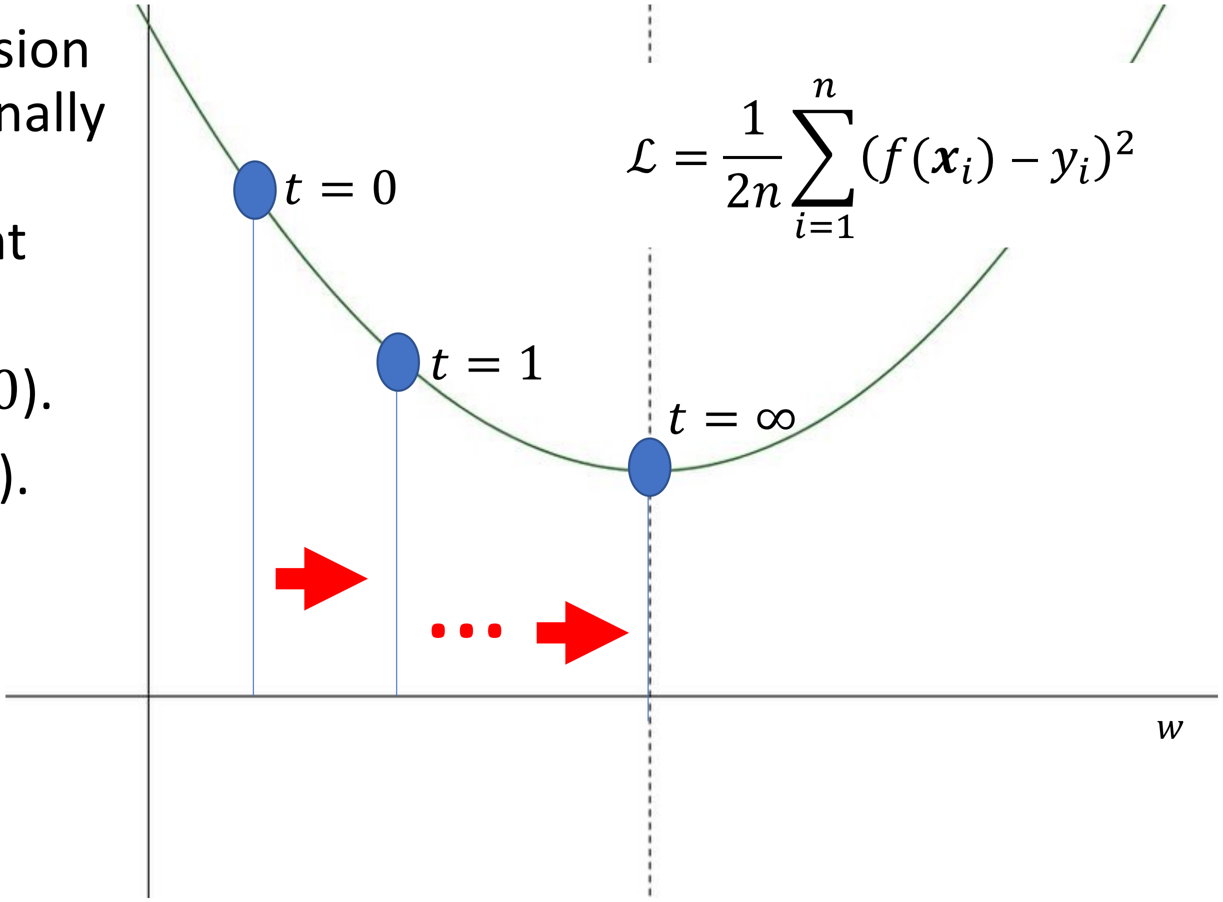


$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} (f(\boldsymbol{x}_i) - y_i)^2$$

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 ($t = \infty$).



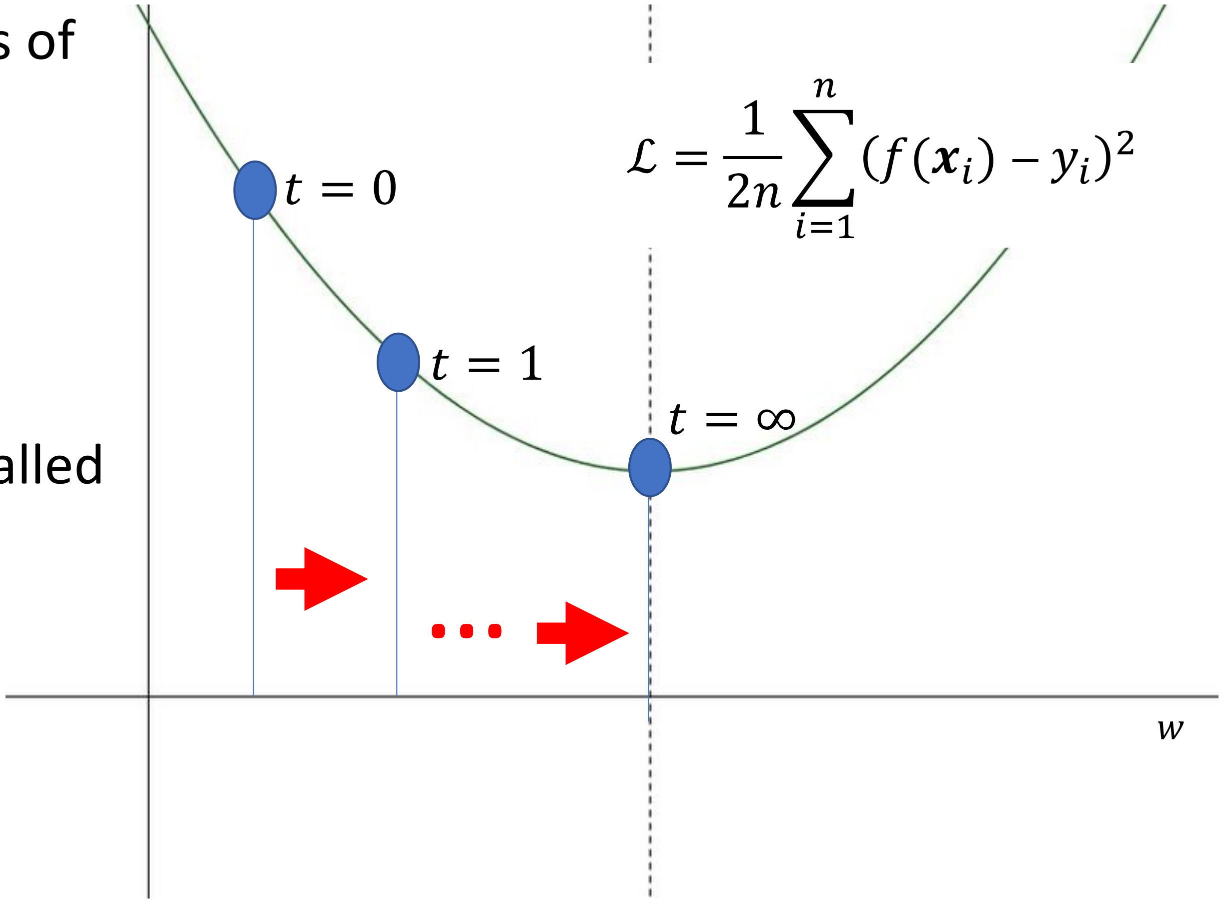
$t = 0$
$t = 1$
$t = \infty$
$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2$
w

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," and $\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}}$ is the ***gradient***.



$\mathcal{L} = \frac{1}{2n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i) - y_i)^2$
$t = 0$
$t = 1$
$t = \infty$
…
w

Gradient

The gradient of a scalar with respect to a vector is defined as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_1} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial w_m} \end{bmatrix}$$

Finding the gradient

Remember that MSE is

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

We can work out that the gradient is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = \frac{1}{2n}\left((\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})^T\boldsymbol{X} + \left(\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})\right)^T\right)^T$$

…or…

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

Gradient descent

Each step of gradient descent updates w as

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

- $\boldsymbol{w}$ is an m-vector
- Finding $\boldsymbol{X}\boldsymbol{w}$ requires $m \times n$ multiplications
- Finding $\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y})$ requires another $m \times n$ multiplications
- If n is very large, each step of gradient descent requires many multiplications!

Outline

- Linear regression
- Vector notation
- 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:

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

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

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:

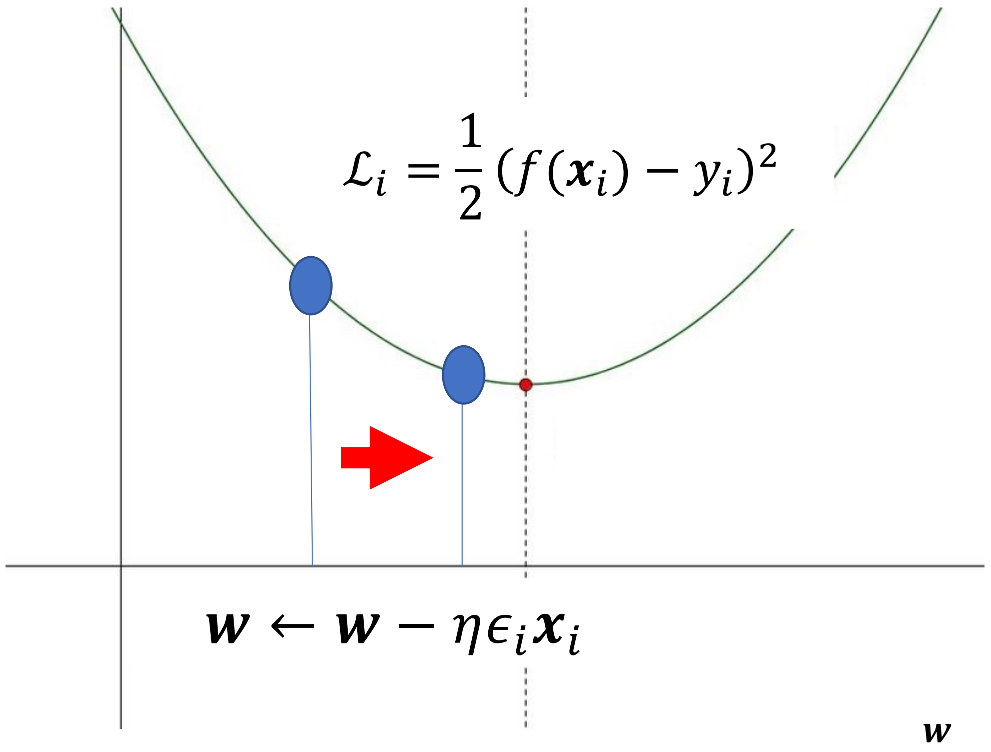
$$\mathcal{L}_i = \frac{1}{2}\epsilon_i^2, \qquad \epsilon_i = \boldsymbol{w}^T\boldsymbol{x}_i - y_i$$

… 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

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

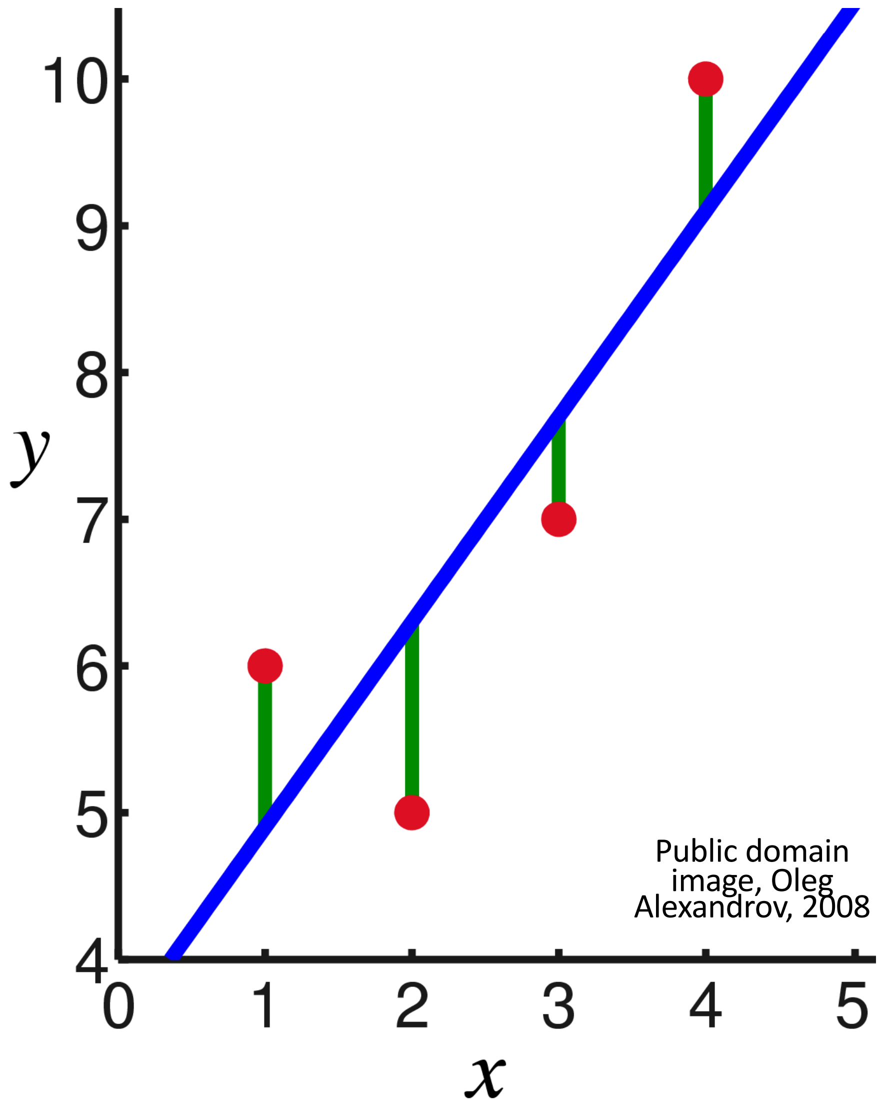

$\mathcal{L}_i = \frac{1}{2}(f(\boldsymbol{x}_i) - y_i)^2$
$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \epsilon_i \boldsymbol{x}_i$
$\boldsymbol{w}$

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** $f(x_i)$



Public domain image, Oleg Alexandrov, 2008

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

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