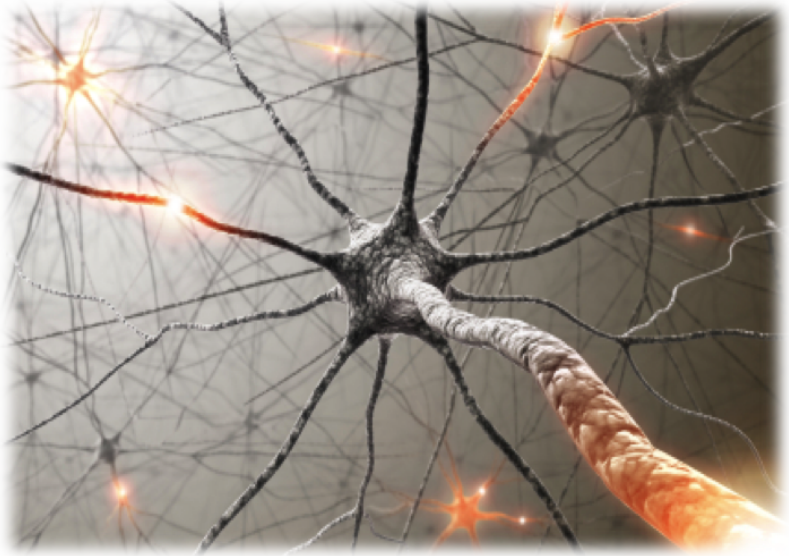
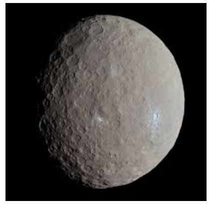
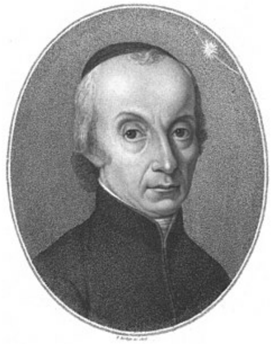
Machine learning: linear regression



The discovery of Ceres





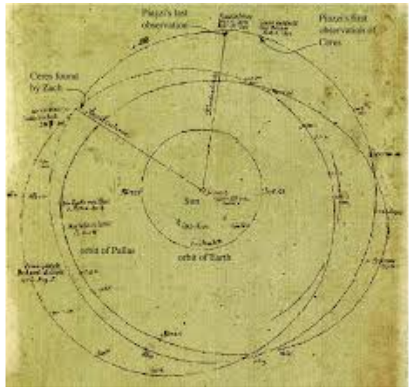
1801: astronomer Piazzi discovered Ceres, made 19 observations of location before it was obscured by the sun

Time	Right ascension	Declination
Jan 01, 20:43:17.8	50.91	15.24
Jan 02, 20:39:04.6	50.84	15.30
...	...	...
Feb 11, 18:11:58.2	53.51	18.43

When and where will Ceres be observed again?

Gauss's triumph

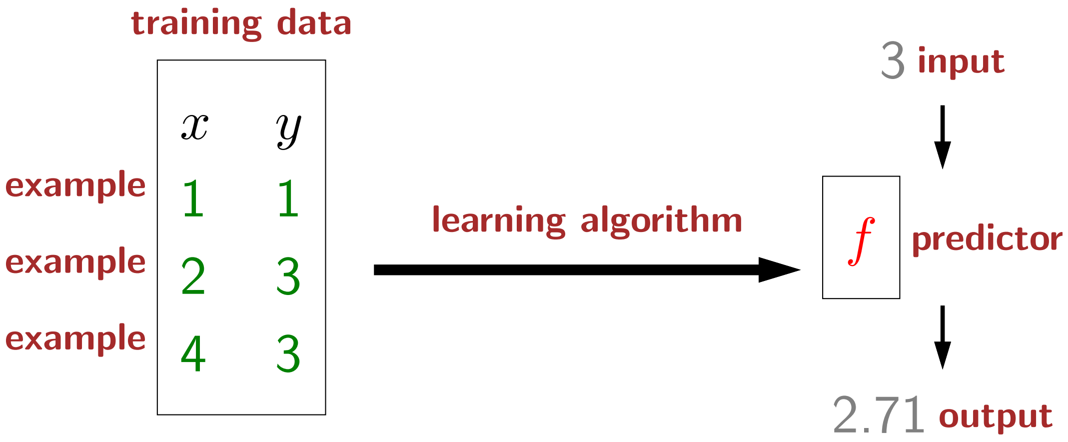
September 1801: Gauss took Piazzi's data and created a model of Ceres's orbit, makes prediction

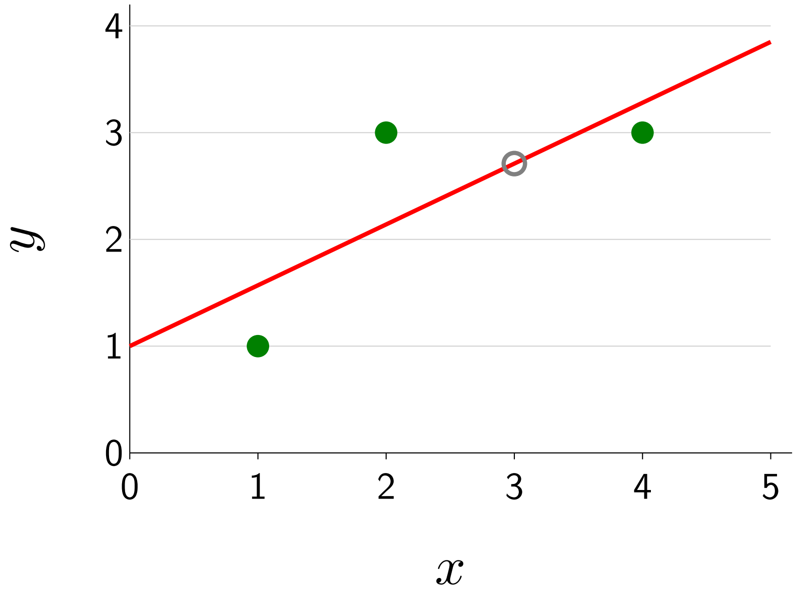


December 7, 1801: Ceres located within 1/2 degree of Gauss's prediction, much more accurate than other astronomers

Method: least squares linear regression

Linear regression framework





Design decisions:

Which predictors are possible? **hypothesis class**

How good is a predictor? **loss function**

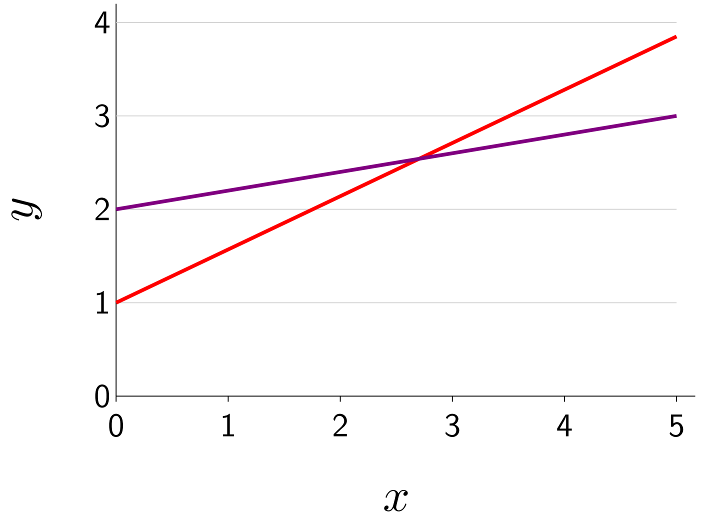
How do we compute the best predictor? **optimization algorithm**

Hypothesis class: which predictors?

$f(x) = 1 + 0.57x$

$f(x) = 2 + 0.2x$

$f(x) = w_1 + w_2 x$



Vector notation:

weight vector $\mathbf{w} = [w_1, w_2]$

feature extractor $\phi(x) = [1, x]$ **feature vector**

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x) \text{ score}$$

$f_{\mathbf{w}}(3) = [1, 0.57] \cdot [1, 3] = 2.71$

Hypothesis class:

$$\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^2\}$$

Loss function: how good is a predictor?

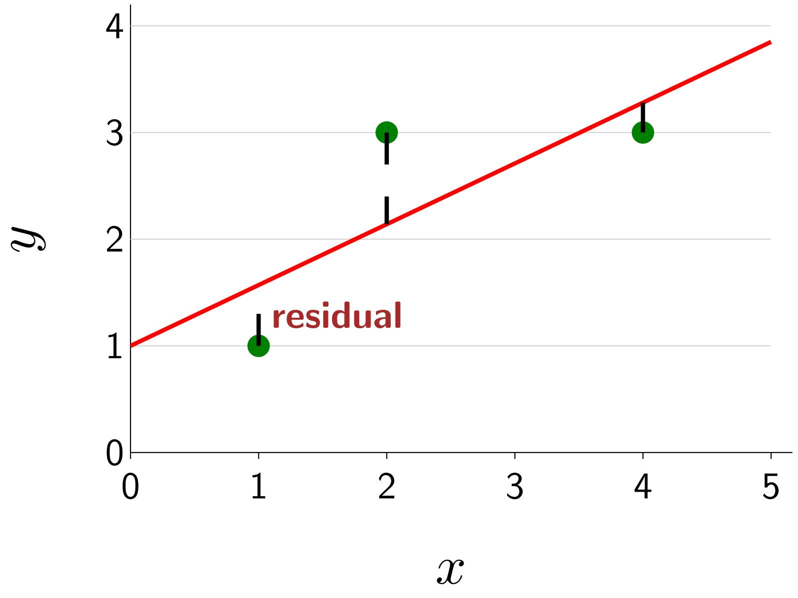
$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$

$\mathbf{w} = [1, 0.57]$

$\phi(x) = [1, x]$

training data $\mathcal{D}_{\text{train}}$

x	y
1	1
2	3
4	3



$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$ **squared loss**

$\text{Loss}(1, 1, [1, 0.57]) = ([1, 0.57] \cdot [1, 1] - 1)^2 = 0.32$

$\text{Loss}(2, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 2] - 3)^2 = 0.74$

$\text{Loss}(4, 3, [1, 0.57]) = ([1, 0.57] \cdot [1, 4] - 3)^2 = 0.08$

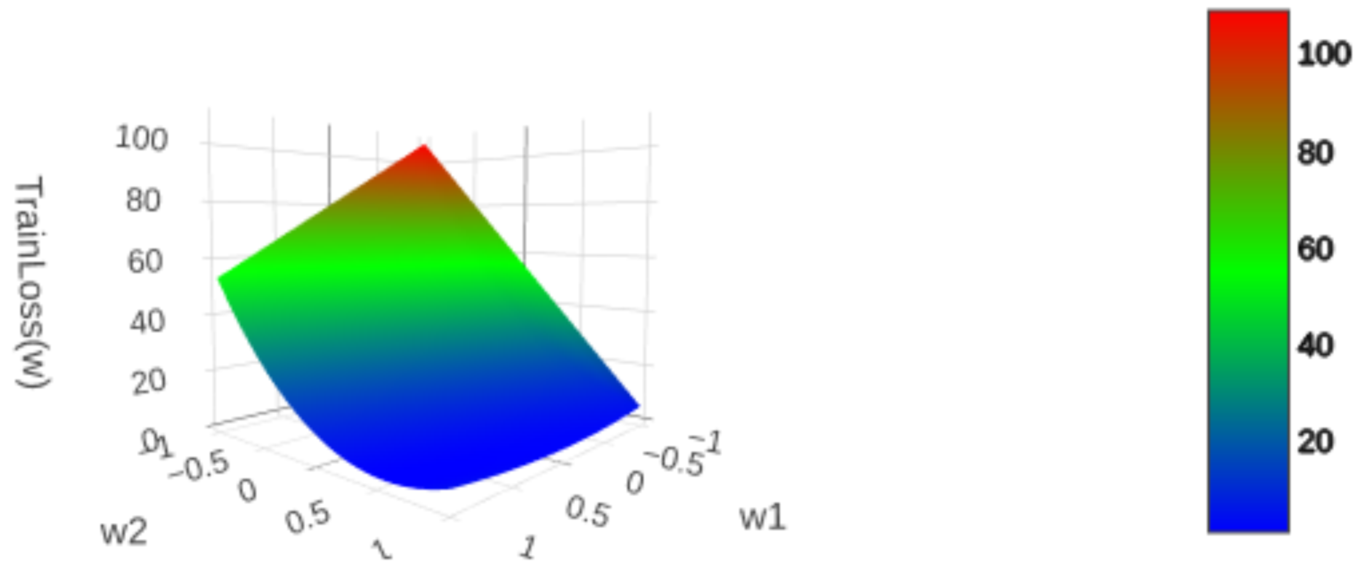
$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w})$

$\text{TrainLoss}([1, 0.57]) = 0.38$

Loss function: visualization

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} (f_{\mathbf{w}}(x) - y)^2$$

$$\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$$

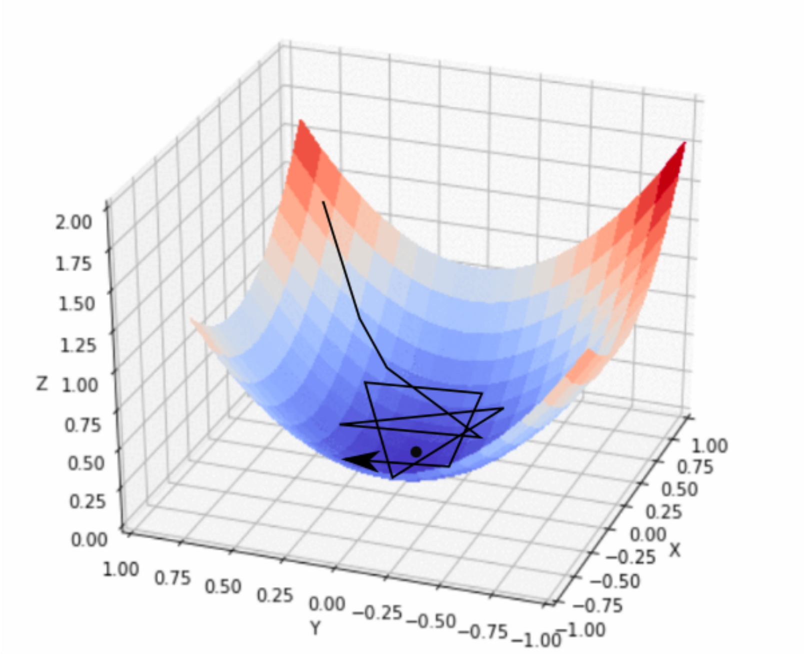


Optimization algorithm: how to compute best?

Goal: $\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

Definition: gradient

The gradient $\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$ is the direction that increases the training loss the most.

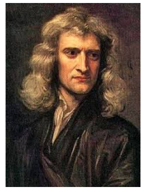


Algorithm: gradient descent

Initialize $\mathbf{w} = [0, \dots, 0]$

For $t = 1, \dots, T$: **epochs**

$$\mathbf{w} \leftarrow \mathbf{w} - \underbrace{\eta}_{\text{step size}} \underbrace{\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})}_{\text{gradient}}$$



Computing the gradient

Objective function:

$$\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} (\mathbf{w}\cdot\phi(x) - y)^2$$

Gradient (use chain rule):

$$\nabla_{\mathbf{w}}\text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} 2(\underbrace{\mathbf{w}\cdot\phi(x) - y}_{\text{prediction}-\text{target}})\phi(x)$$

Gradient descent example

training data $\mathcal{D}_{\text{train}}$

x	y
1	1
2	3
4	3

$$\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} 2(\mathbf{w}\cdot\phi(x) - y)\phi(x)$$

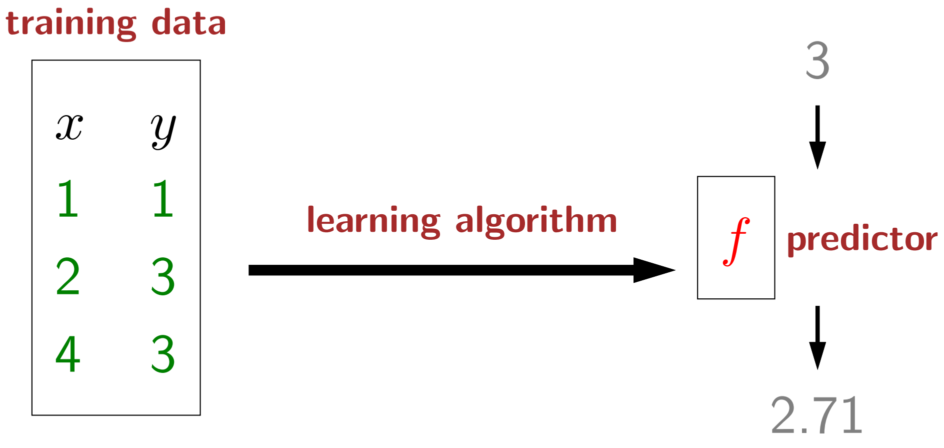
Gradient update: $\mathbf{w} \leftarrow \mathbf{w} - 0.1\nabla_{\mathbf{w}}\text{TrainLoss}(\mathbf{w})$

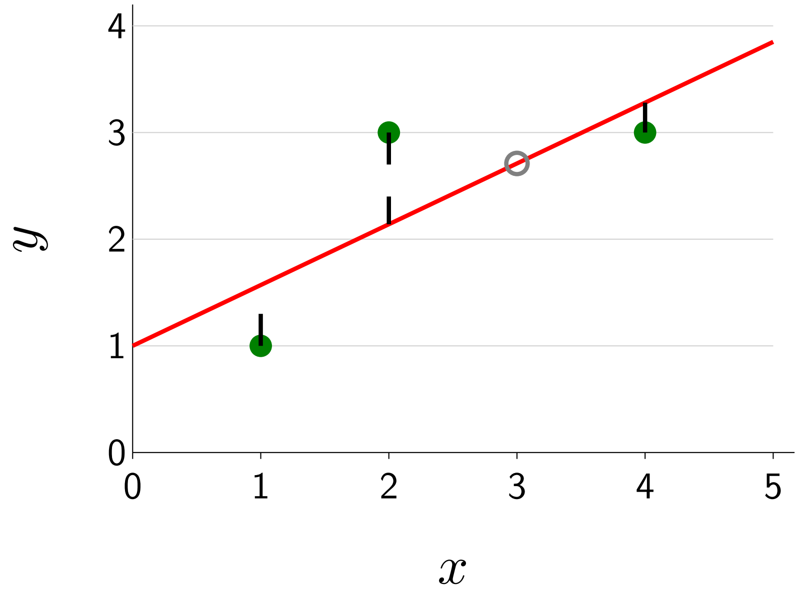
t	$\nabla_{\mathbf{w}}\text{TrainLoss}(\mathbf{w})$	$\mathbf{w}$
		$[0, 0]$
1	$\underbrace{\frac{1}{3}(2([0,0]\cdot[1,1]-1)[1,1]+2([0,0]\cdot[1,2]-3)[1,2]+2([0,0]\cdot[1,4]-3)[1,4])}_{=[-4.67,-12.67]}$	$[0.47, 1.27]$
2	$\underbrace{\frac{1}{3}(2([0.47,1.27]\cdot[1,1]-1)[1,1]+2([0.47,1.27]\cdot[1,2]-3)[1,2]+2([0.47,1.27]\cdot[1,4]-3)[1,4])}_{=[2.18,7.24]}$	$[0.25, 0.54]$
...	...	...
200	$\underbrace{\frac{1}{3}(2([1,0.57]\cdot[1,1]-1)[1,1]+2([1,0.57]\cdot[1,2]-3)[1,2]+2([1,0.57]\cdot[1,4]-3)[1,4])}_{=[0,0]}$	$[1, 0.57]$

Gradient descent in Python

[code]

Summary





Which predictors are possible?

Hypothesis class

Linear functions

$\mathcal{F} = \{f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)\}, \phi(x) = [1, x]$

How good is a predictor?

Loss function

Squared loss

$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(x) - y)^2$

How to compute best predictor?

Optimization algorithm

Gradient descent

$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla \textbf{TrainLoss}(\mathbf{w})$