REGRESSION

Regression

Relationships among several quantities

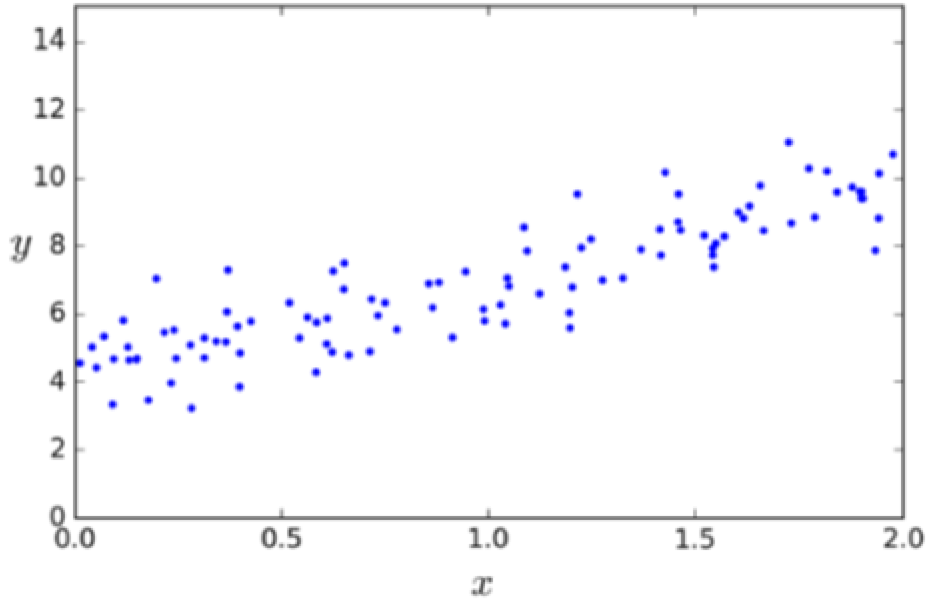
build a model that predicts the value of one variable as a function of other variables

simplest relation between two variables x and y is the linear equation

Regression coefficients

$$y = \beta_0 + \beta_1 x$$

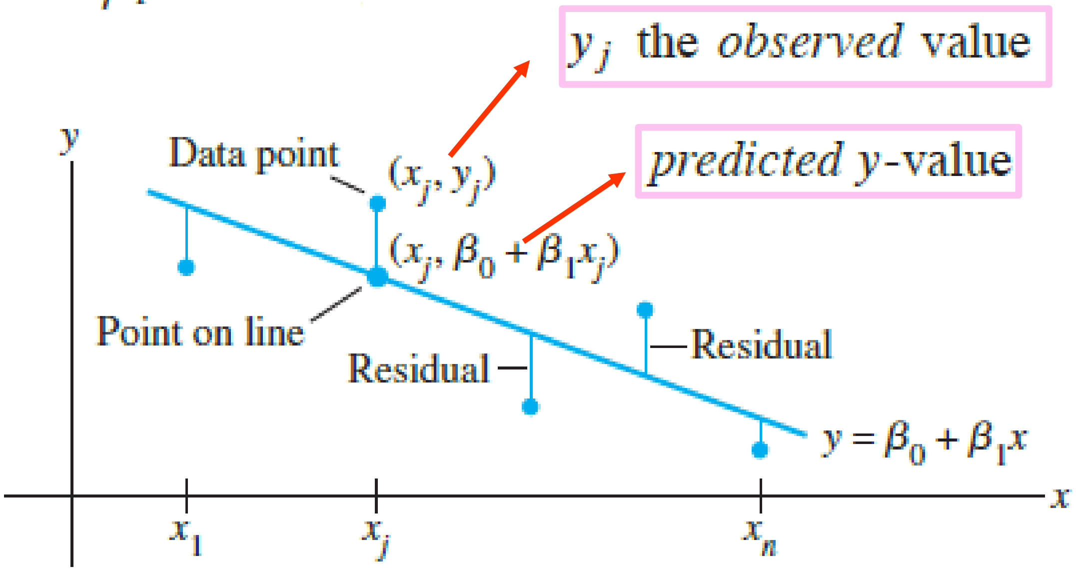
$$(x_1, y_1), \ldots, (x_n, y_n)$$



Regression of y on x

Regression

Suppose β_0 and β_1 are fixed



Regression

If the data points were on the line, the parameters would satisfy the equations

Predicted y-value		Observed y-value
$\beta_0 + \beta_1 x_1$	$=$	y_1
$\beta_0 + \beta_1 x_2$	$=$	y_2
$\vdots$		$\vdots$
$\beta_0 + \beta_1 x_n$	$=$	y_n

if the data points don't lie on a line

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$\boxed{X\boldsymbol{\beta} = \mathbf{y}}$$

Regression

There are several ways to measure how “close” the line is to the data

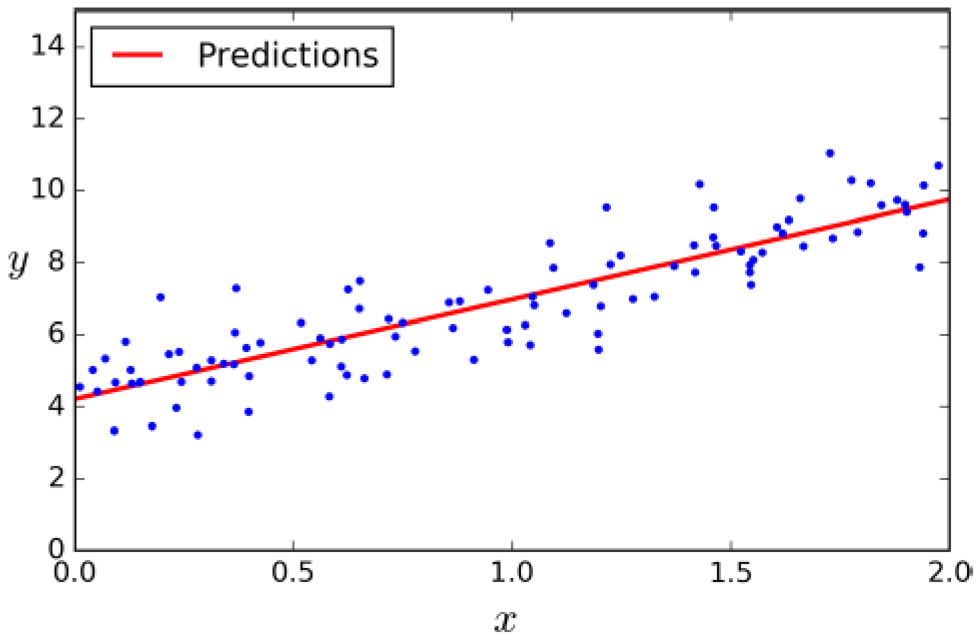
The usual choice is to add the squares of the residuals

least-squares line is the that minimizes the sum of the squares of the residuals

$$residual = \epsilon = y - X\beta$$

$$\min \|X\beta - y\|_2^2$$

Cost function

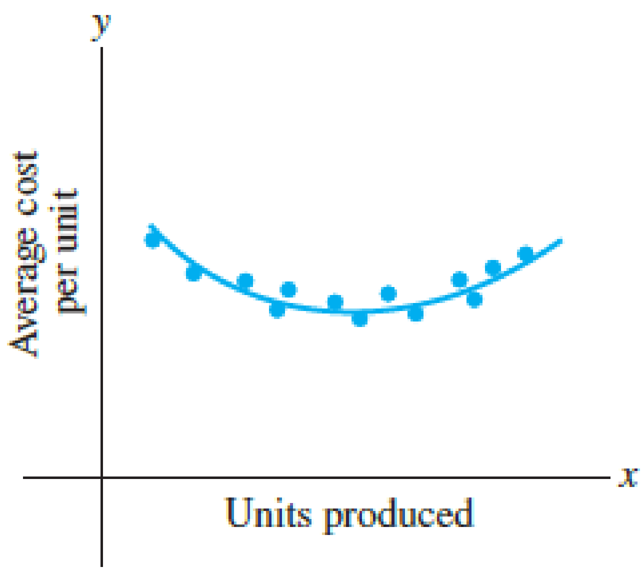


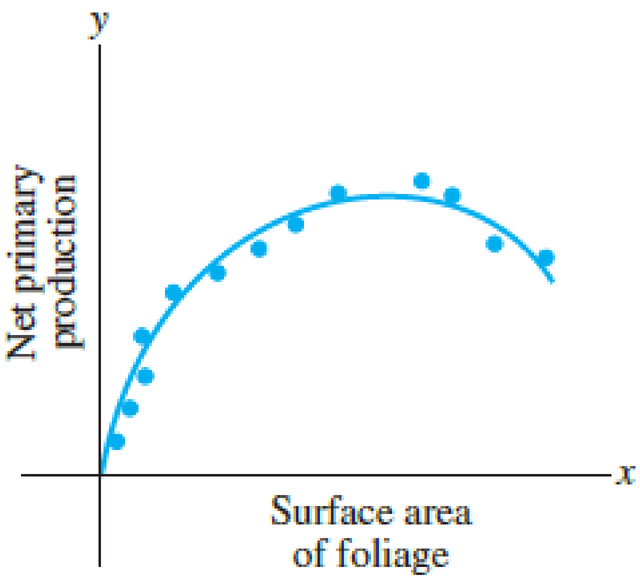
Regression(Curve fitting)

data points $(x_1, y_1), \ldots, (x_n, y_n)$ on a scatter plot do not lie close to any line.

some other functional relationship between x and y

$$y = \beta_0 f_0(x) + \beta_1 f_1(x) + \cdots + \beta_k f_k(x)$$





Regression(Curve fitting)

Example

$$\boxed{y = \beta_0 + \beta_1 x + \beta_2 x^2}$$

$(x_1, y_1), \ldots, (x_n, y_n)$

$$\begin{aligned}
y_1 &= \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \epsilon_1 \\
y_1 &= \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \epsilon_1 \\
y_2 &= \beta_0 + \beta_1 x_2 + \beta_2 x_2^2 + \epsilon_2 \\
\vdots & \qquad\qquad\quad \vdots \\
y_n &= \beta_0 + \beta_1 x_n + \beta_2 x_n^2 + \epsilon_n
\end{aligned}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\mathbf{y} = X \boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$$residual = \epsilon = y - X\beta$$

$$\min \|X\beta - y\|_2^2$$

Multiple Regression

Multiple Regression

We have n features and we want to predict y based on them

$$\boldsymbol{x_1, x_2, \dots, x_m} \Rightarrow \boldsymbol{y}$$

$$\boldsymbol{y = \beta_0 + \beta_1 x_1 + \beta_2\, x_2 + \cdots + \beta_m x_m}$$

$$\boldsymbol{y = \theta_0 + \theta_1 x_1 + \theta_2\, x_2 + \cdots + \theta_m x_m}$$

$$\boldsymbol{y = \omega_0 + \omega_1 x_1 + \omega_2\, x_2 + \cdots + \omega_m x_m}$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_m^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_1^{(n)} & \dots & x_m^{(n)} \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

$$\boxed{\begin{gathered} \boldsymbol{residual = \epsilon = y - X\beta} \\ \boldsymbol{\min \|X\beta - y\|_2^2} \end{gathered}}$$

Multiple Regression

$$\boldsymbol{y} = \boldsymbol{\beta_0} + \boldsymbol{\beta_1 f_1(x_1)} + \boldsymbol{\beta_2 f_2(x_2)} + \cdots + \boldsymbol{\beta_m f_m(x_m)}$$

$$X = \begin{bmatrix} 1 & f_1(x_1^{(1)}) & \dots & f_m(x_m^{(1)}) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f_1(x_1^{(n)}) & \dots & f_m(x_m^{(n)}) \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix}$$

Least Square Problem

$$\boldsymbol{residual = \epsilon = y - X\beta}$$

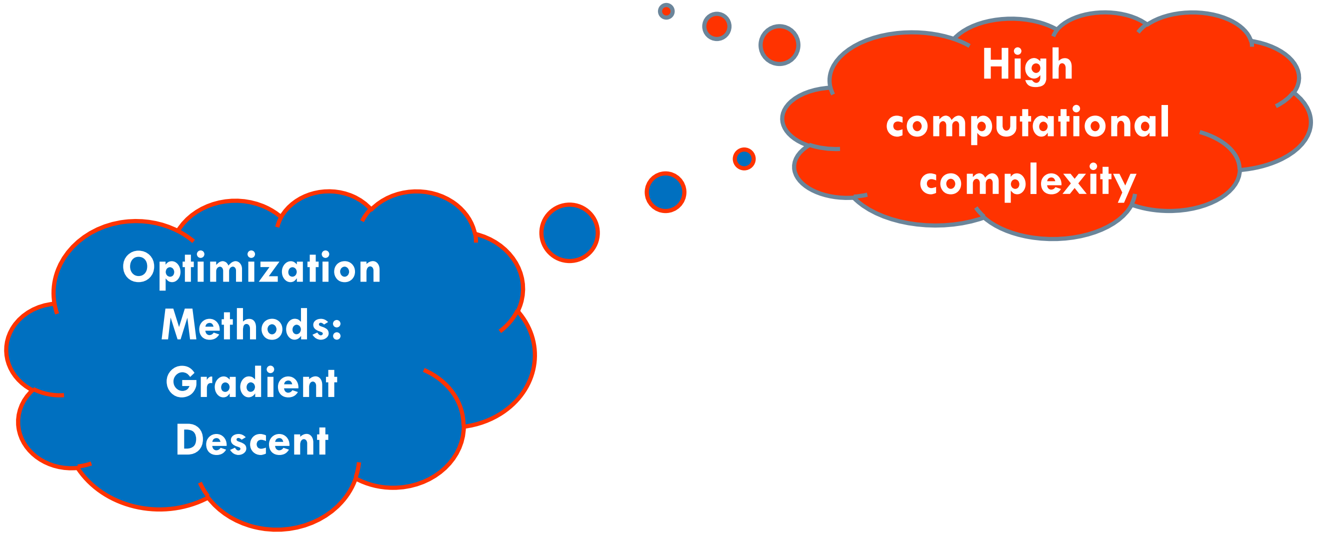
$$\mathbf{min}\ \|\boldsymbol{X\beta - y}\|_2^2$$

Solving Least Square Problem

least-squares solution is a solution of the normal equations

$$\boxed{\min \|X\beta - y\|_2^2}$$

$$X^T X \beta = X^T \mathbf{y}$$



Optimization algorithms

x_0

$\Downarrow$

generate a sequence of iterates $\{x_k\}_{k=0}^{\infty}$

$\Downarrow$

terminate : no more progress or a solution point with sufficient accuracy

$$\boldsymbol{x_{k+1} = x_k + \alpha p_k}$$

direction

Step length or Learning Rate

α?

p_k?

Optimization algorithms

direction

Descent methods

✓any descent direction is guaranteed to produce a decrease in f , provided that the step length is sufficiently small

Need definition of Gradient

Gradient

$$f : \mathbf{R}^n \to \mathbf{R}$$

f is
real-valued

$$\nabla f(x) \quad \Rightarrow \quad \nabla f(x)_i = \frac{\partial f(x)}{\partial x_i}, \quad i = 1, \dots, n.$$

Gradient: example

Example:

quadratic function

$$f : \mathbf{R}^n \to \mathbf{R}$$

$$f(x) = (1/2)x^T P x + q^T x + r$$

$P \in \mathbf{S}^n$, $q \in \mathbf{R}^n$, and $r \in \mathbf{R}$

$$Df(x) = x^T P + q^T$$

$$\nabla f(x) = Px + q$$

Directional Derivative

$$\nabla_p f(x) = \langle \nabla f(x), p \rangle$$

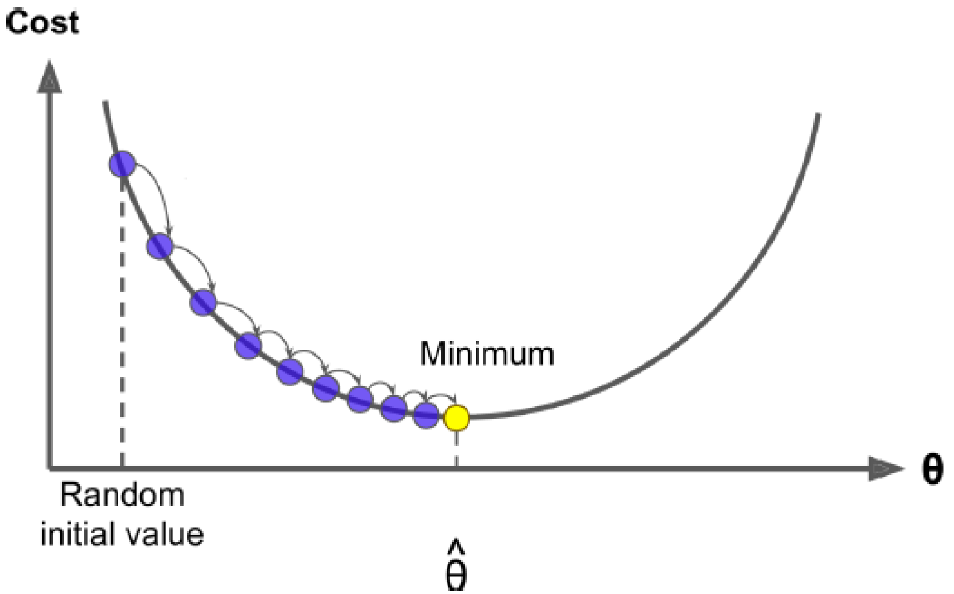
$\nabla_p f(x) < 0$ $\Rightarrow$ P is a descent direction

Gradient Descent

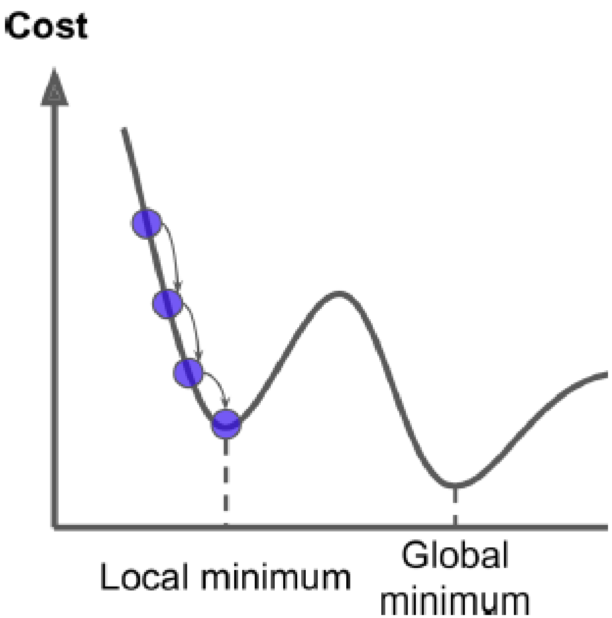
steepest descent direction

✓steepest descent direction $-\nabla f_k$ is the most obvious choice for search direction for a line search method.

✓choose the step length α in a variety of ways $$x_{k+1} = x_k + \alpha_k(-\nabla f(x_k))$$



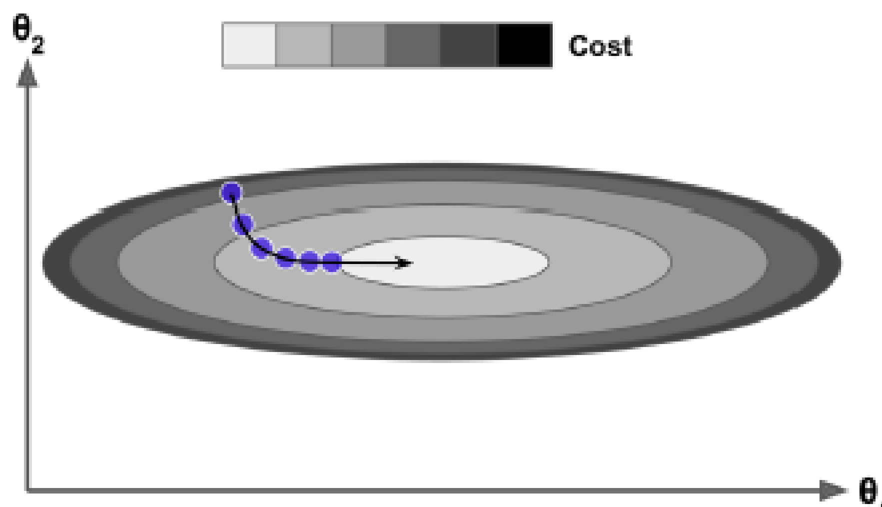
Gradient Descent



$$f(X) = \|X\beta - y\|_2^2$$

Convex function

$$\nabla f(x) = X^T X\beta - X^T y$$

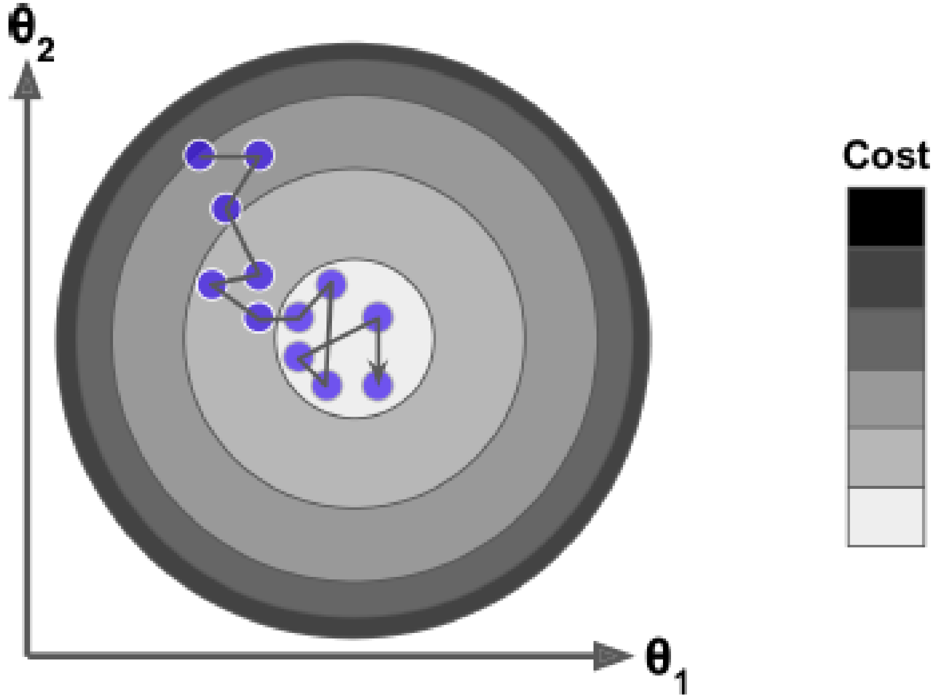


Gradient Descent

- ✓ training a model means searching for a combination of model parameters that minimizes a cost function
- ✓ search in the model's parameter space
- ✓ more parameters a model has, more dimensions this space has, and the harder search
- ✓ Batch Gradient Descent uses the whole training set to compute the gradients at every step, which makes it very slow when the training set is large.

- ✓ Stochastic Gradient Descent : picks a random instance in the training set at every step and computes the gradients based only on that single instance.
- ✓ is much less regular than Batch Gradient Descent
- ✓ instead of gently decreasing until it reaches the minimum, the cost function will bounce up and down, decreasing only on average.

Stochastic Gradient Descent

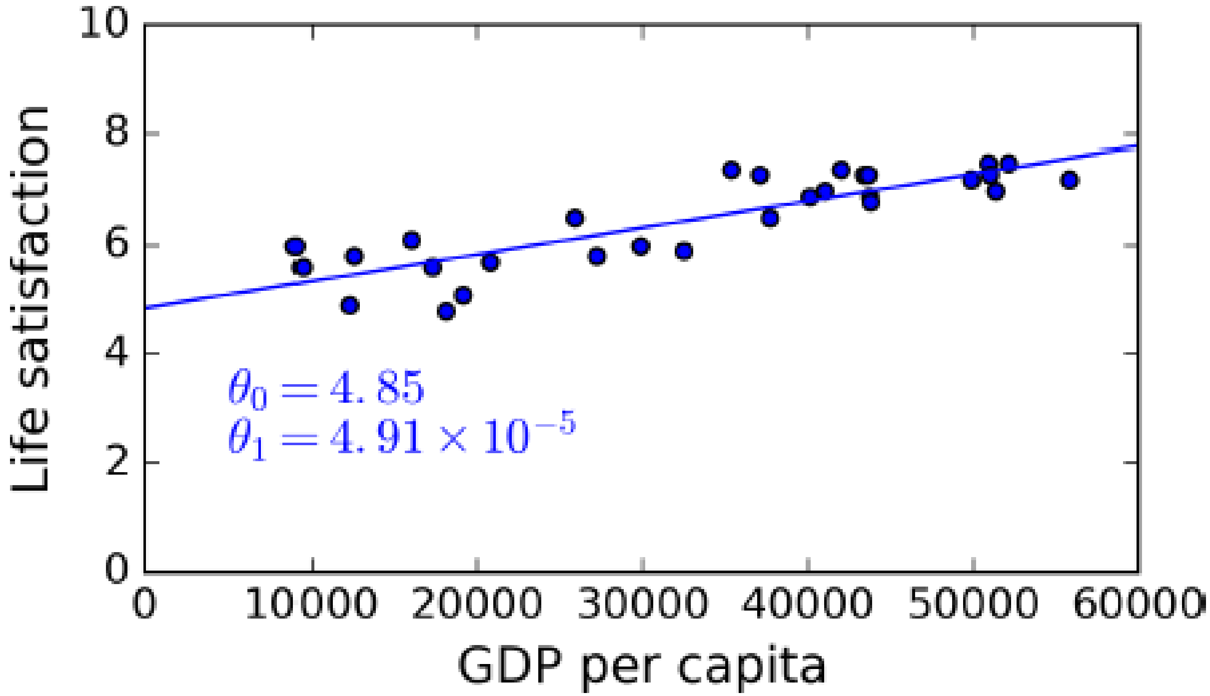


final parameter values are good, but not optimal

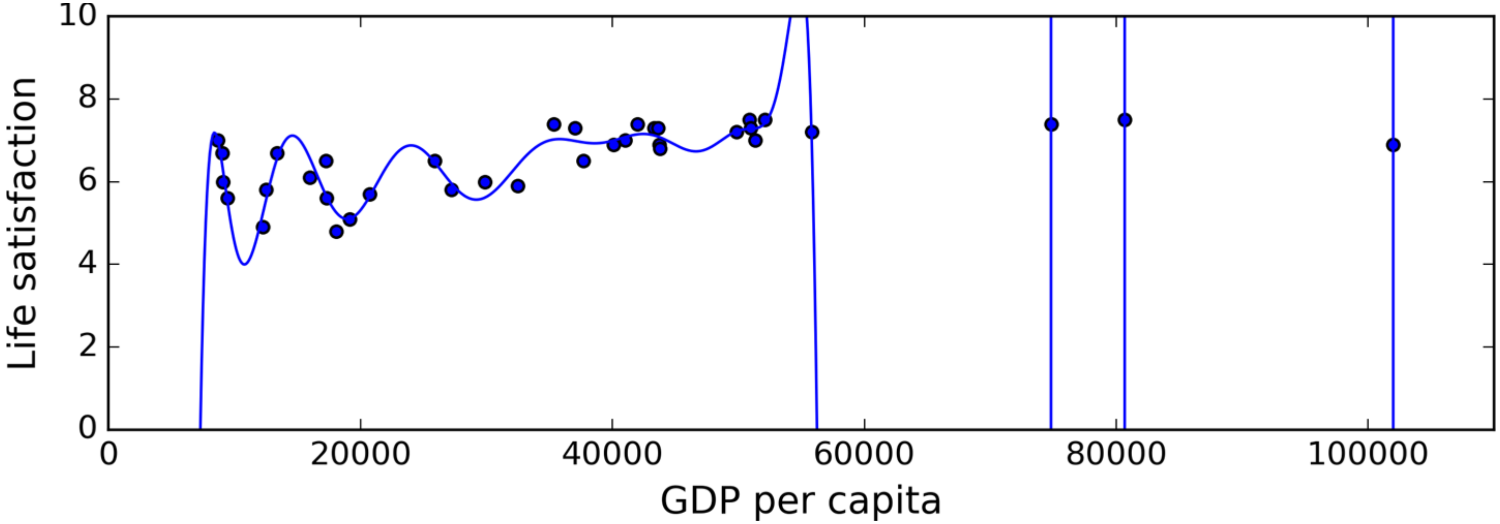
Overfitting and Underfitting

Overfitting

Overgeneralizing is something that we humans do all too often
machines can fall into the same trap
In Machine Learning this is called overfitting
model performs well on the training data, but it does not generalize well



Overfitting



Even though it performs much better on the training data than the simple linear model, would you really trust its predictions?

Overfitting

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data

- simplify the model
- gather more training data
- reduce the noise in the training data

Underfitting

- underfitting is the opposite of overfitting
- it occurs when your model is too simple to learn the underlying structure of the data
- more powerful model