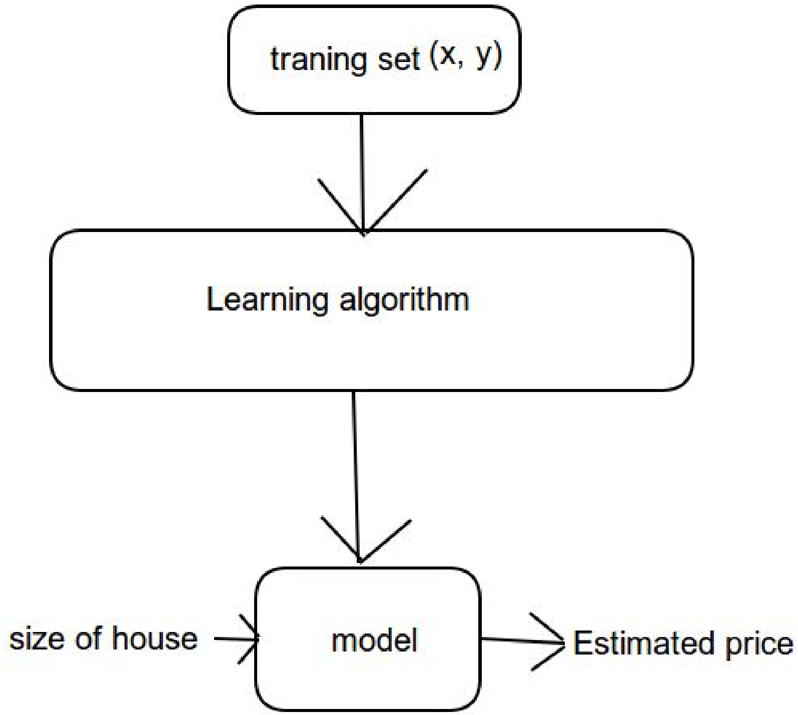
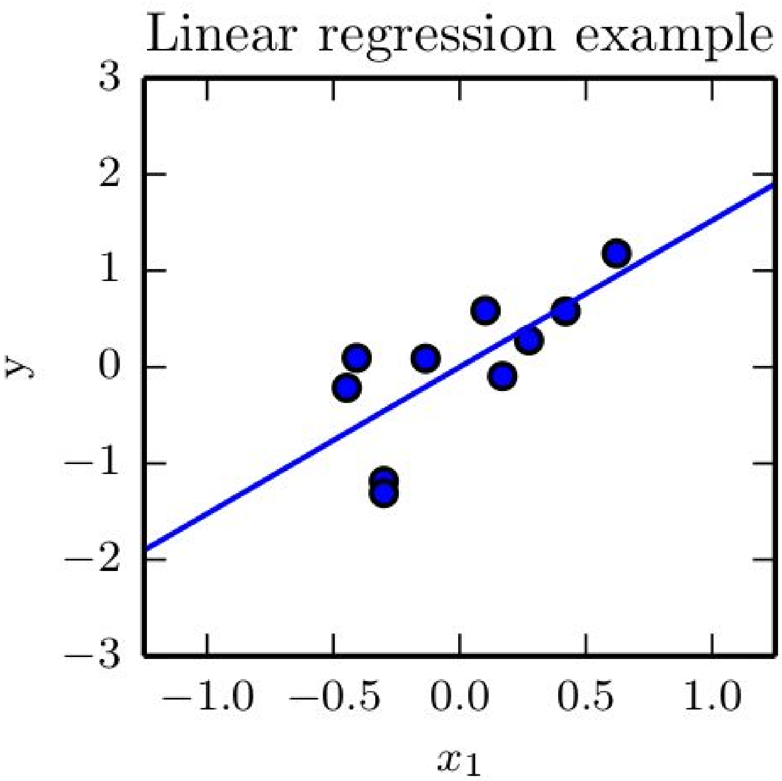
Tom Mitchell:
A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.


traning set (x, y)
Learning algorithm
size of house
model
Estimated price

Linear Regression



$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$$

$\boldsymbol{x} \in \mathbb{R}^n$

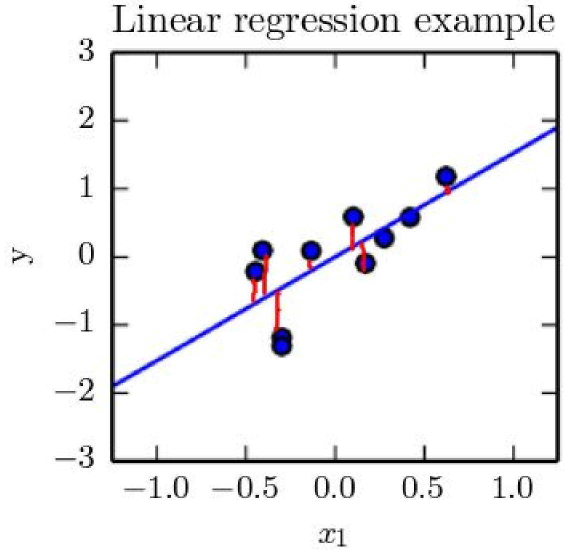
$y \in \mathbb{R}$

$\boldsymbol{w} \in \mathbb{R}^n$ is a vector of **parameters**

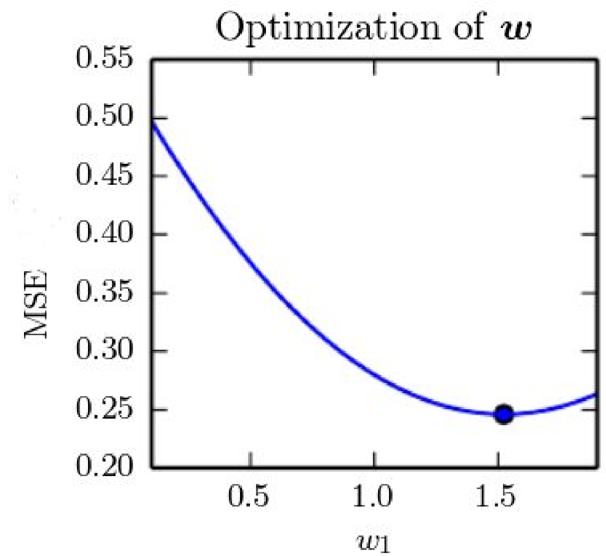
Cost function

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$$

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$



$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$



Normal Equation

Design matrix

N – number of traning examples
N = number of features

X = m * (n+1)
Y = n

$$\nabla_{\boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

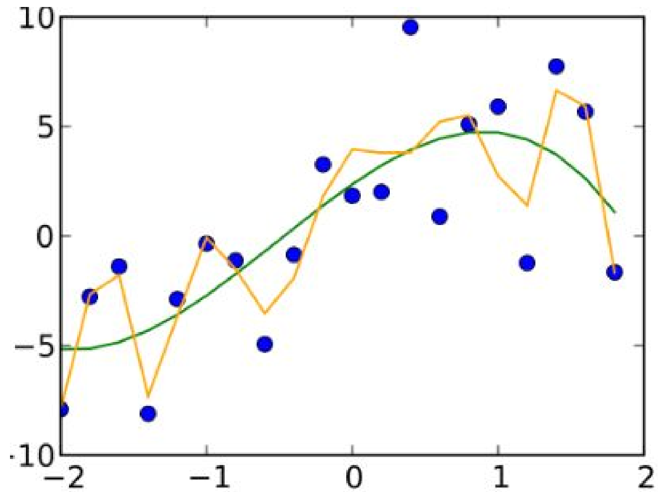
$$\cdot\ \boldsymbol{w} = \left(\boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{X}^{(\mathrm{train})}\right)^{-1} \boldsymbol{X}^{(\mathrm{train})\top} \boldsymbol{y}^{(\mathrm{train})}$$

Training Set/ Test Set

Make traning error small

Make the gap between traning error and test error small

nonlinear functions & model capacity

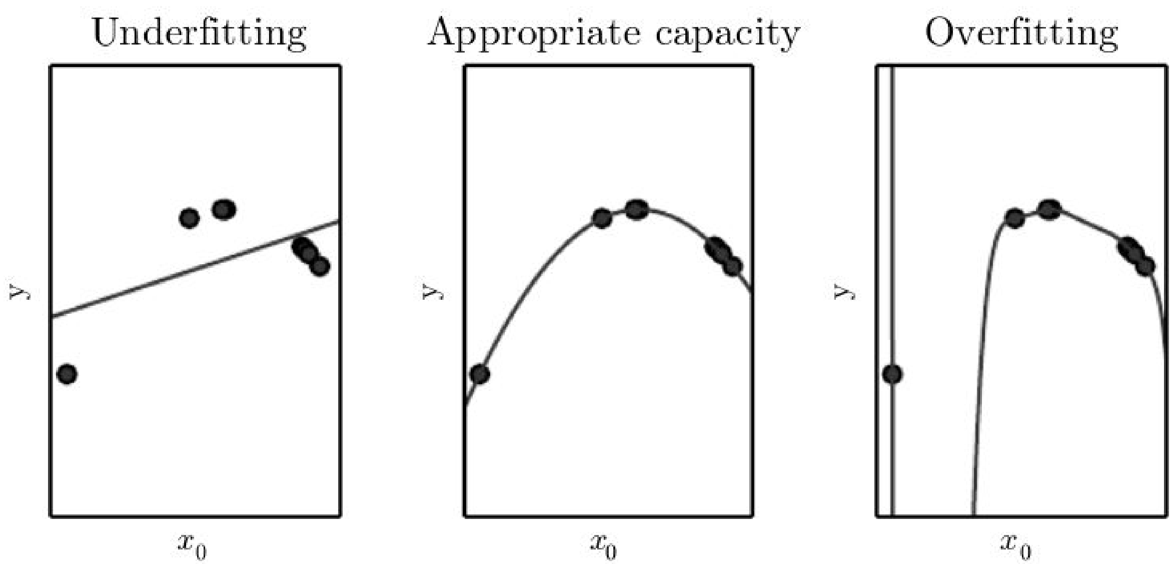


choose the degree of polynomial

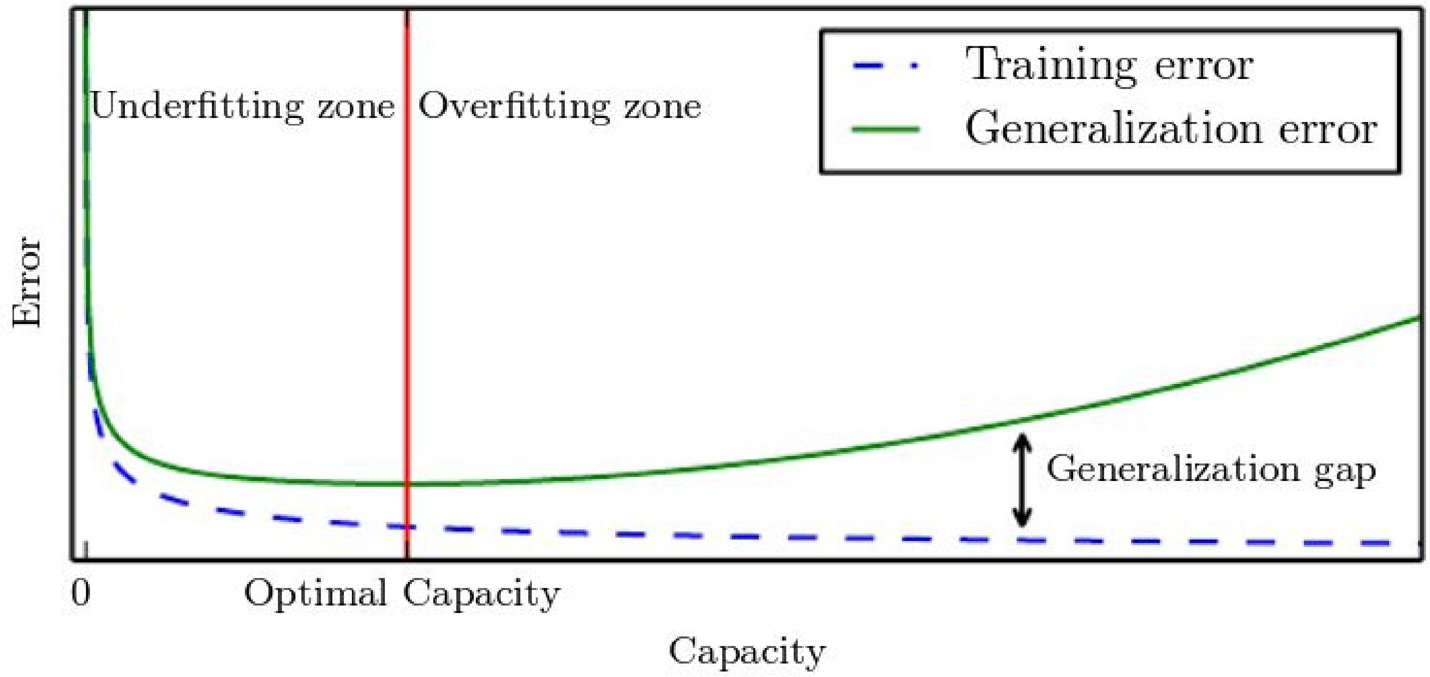
$$\hat{y} = b + wx.$$

$$\hat{y} = b + w_1 x + w_2 x^2.$$

Overfitting/underfitting

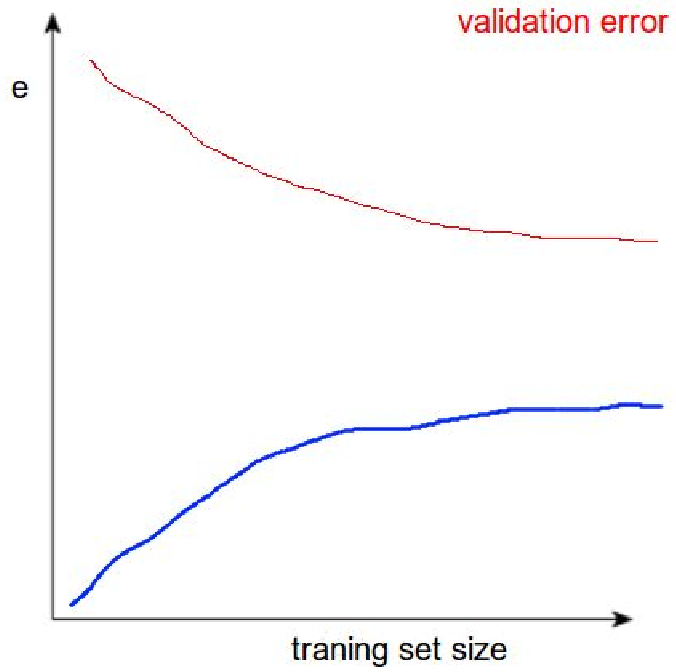


Overfitting/underfitting & Capacity

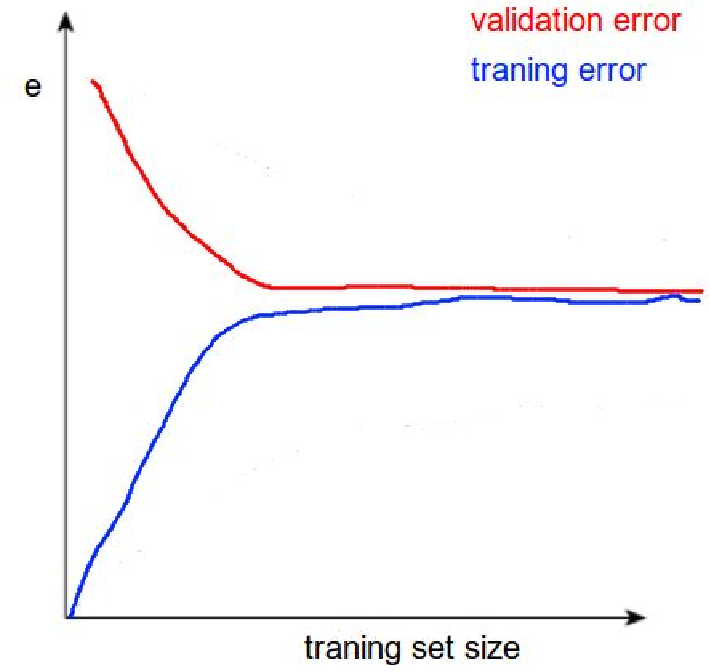


Learning curves

High variance



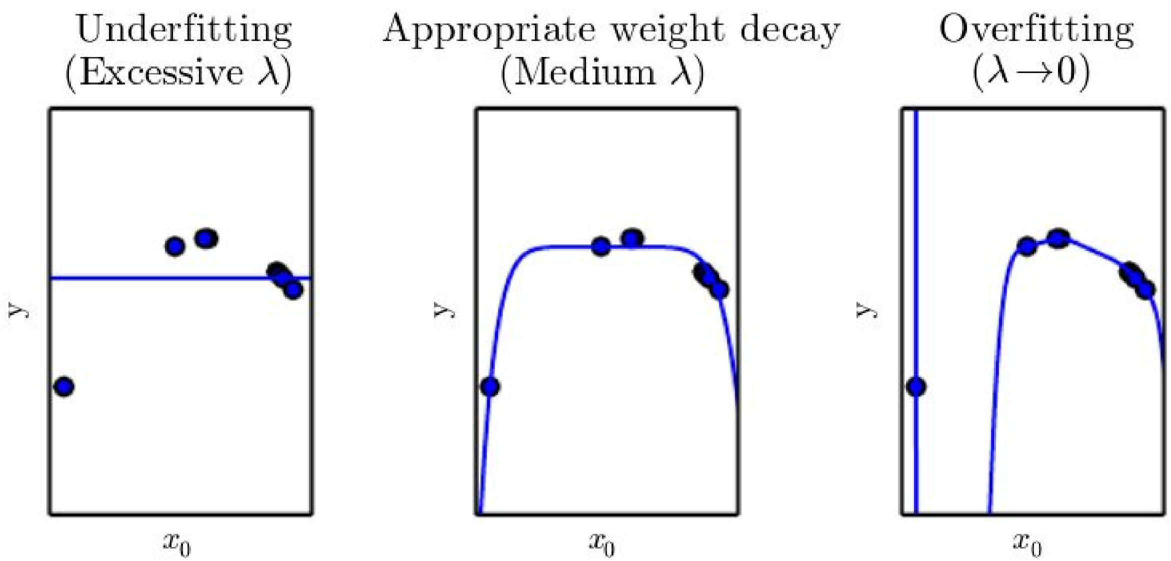
High bias



No Free Lunch Theorem

No machine learning algorithm is universally any better than any other

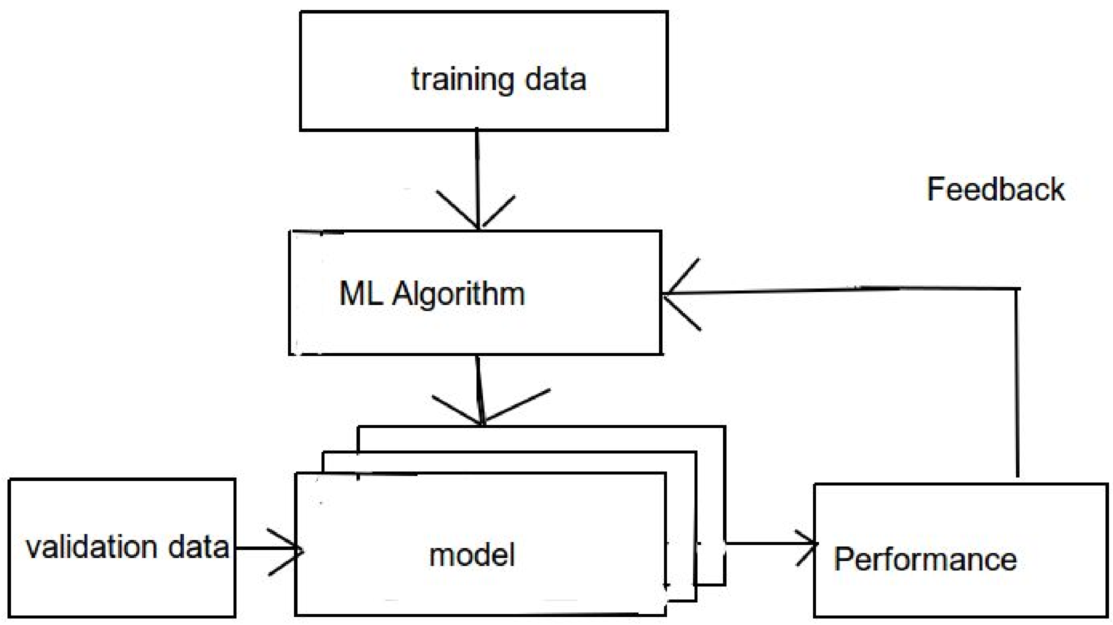
Regularization



$$J(\boldsymbol{w}) = \mathrm{MSE}_{\text{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w},$$

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not itstraining error.

Hyperparameters and Validation set



Test set – 20%

Validation set – 20%

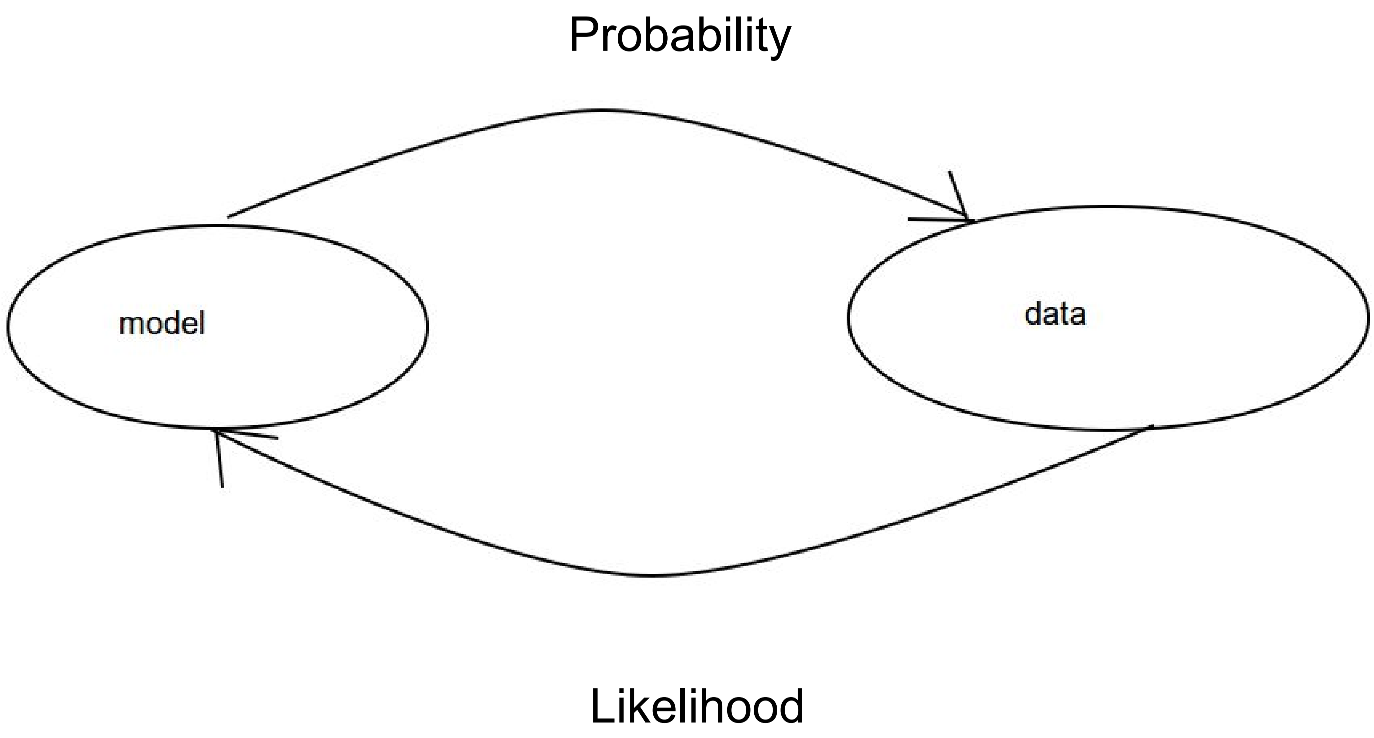
Traning set – 60%

Frequentist

Frequentist assume that the true parameter value θ is fixed but unknown. They learn true value by repeating experiment over and over again.

$$\mathrm{plim}_{m\to\infty}\hat{\theta}_m = \theta.$$

Likelihood



$$\mathcal{L}(\theta|x) = P(x|\theta).$$

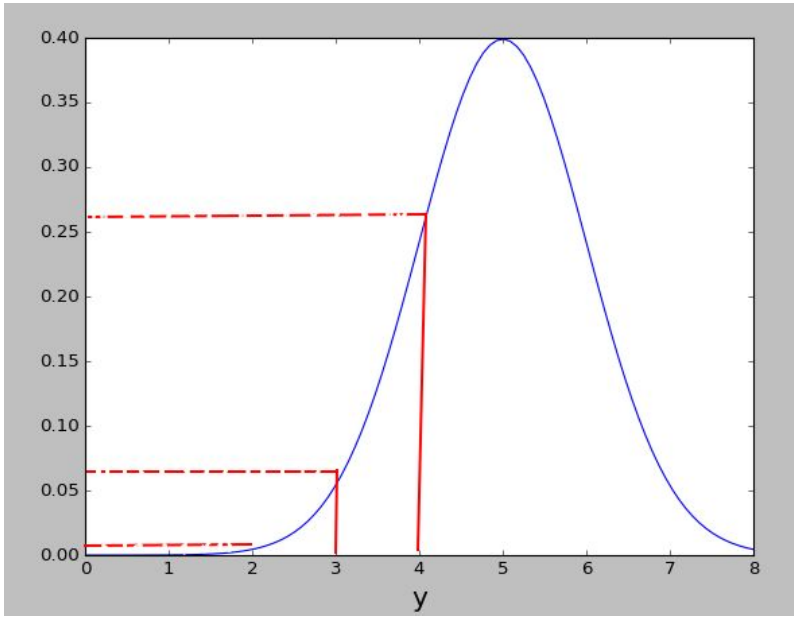
Maximum Likelihood Estimation

normally distributed three data points y1 = 2, y2 = 3 y3 = 4
unknown mean θ and variance 1 , Y indep.

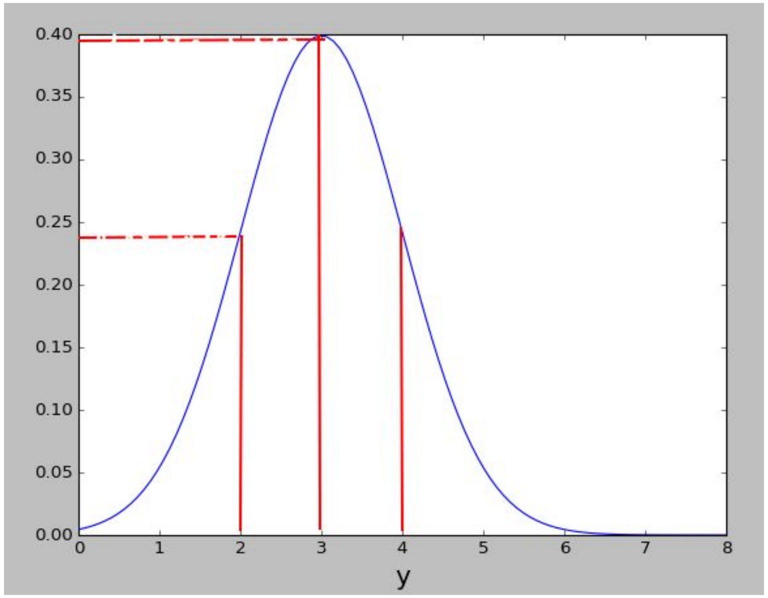
P(y1, y2, y3 | θ) = P(y1, θ) * P(y2, θ) * P(y3, θ)

θ − that maximizes the likelihood ?

P(y|θ=5)



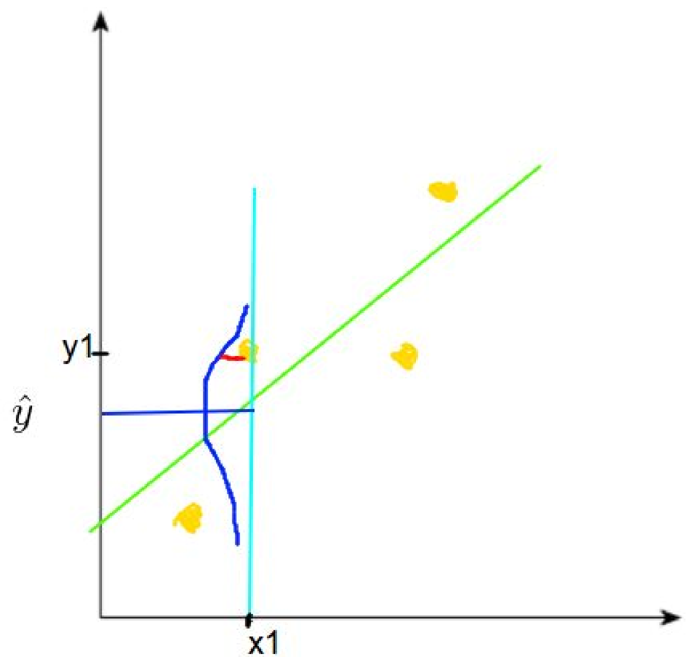
P(y|θ=3)



Linear Regression as Maximum Likelihood

Y is gaussian, mean = $\boldsymbol{w}^\top \boldsymbol{x}$ Variance = σ^2

$$p(y \mid \boldsymbol{x}) = \mathcal{N}(y; \hat{y}(\boldsymbol{x}; \boldsymbol{w}), \sigma^2).$$



Maximum Likelihood Estimation

think of the model as producing a conditional distribution p(y | x)

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{Y} \mid \boldsymbol{X}; \boldsymbol{\theta}). \tag{5.62}$$

If the examples are assumed to be i.i.d., then this can be decomposed into

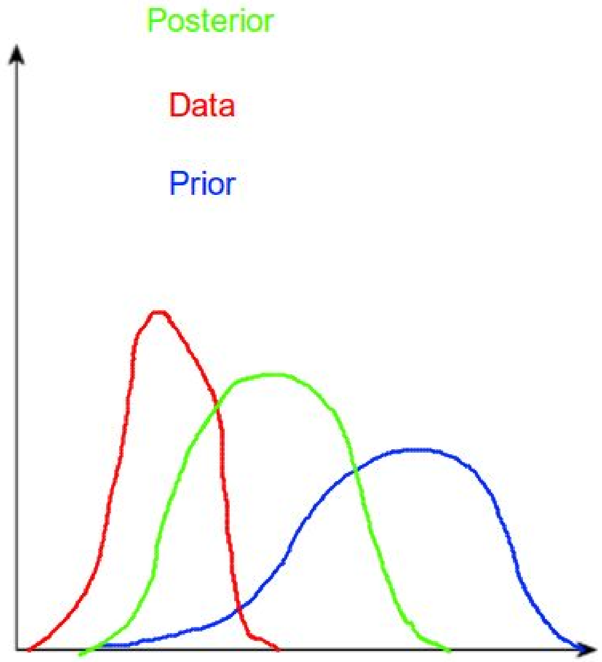
$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P(\boldsymbol{y}^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}). \tag{5.63}$$

$$\sum_{i=1}^{m} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= -m \log \sigma - \frac{m}{2} \log(2\pi) - \sum_{i=1}^{m} \frac{\left\| \hat{y}^{(i)} - y^{(i)} \right\|^2}{2\sigma^2},$$

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{m} \sum_{i=1}^{m} ||\hat{y}^{(i)} - y^{(i)}||^2,$$

Bayesian reasoning



Bayesian Statistics

The true parameter θ is unknown or uncertain and thus is represented as a random variable

$$p(\boldsymbol{\theta} \mid x^{(1)}, \ldots, x^{(m)}) = \frac{p(x^{(1)}, \ldots, x^{(m)} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(x^{(1)}, \ldots, x^{(m)})}$$

$p(\boldsymbol{\theta} \mid x^{(1)}, \ldots, x^{(m)})$ likelihood

$p(\boldsymbol{\theta})$ Before observing the data we represent out knowledge of θ using The prior probability distribution.

$p(x^{(1)}, \ldots, x^{(m)} \mid \boldsymbol{\theta})$ posterior

$p(x^{(1)}, \ldots, x^{(m)})$ constant

Bayesian Linear Regression

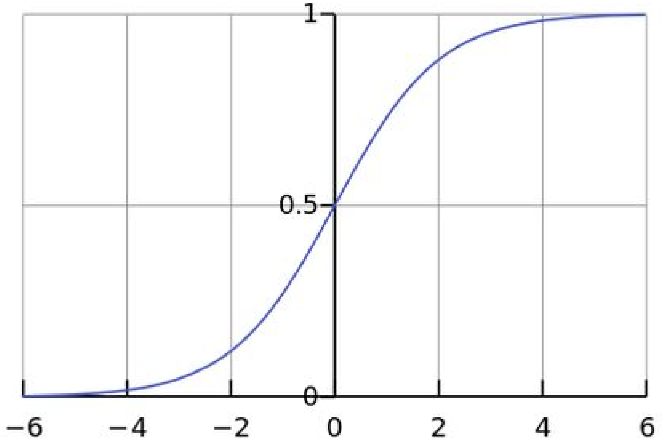
$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) \propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w})$$

Logistic Regression

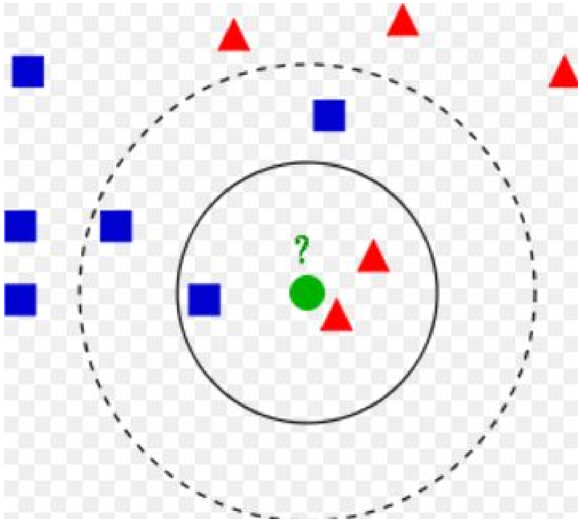
Y {0, 1}

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

$$p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = \sigma(\boldsymbol{\theta}^\top \boldsymbol{x}).$$



k
-nearest neighbors

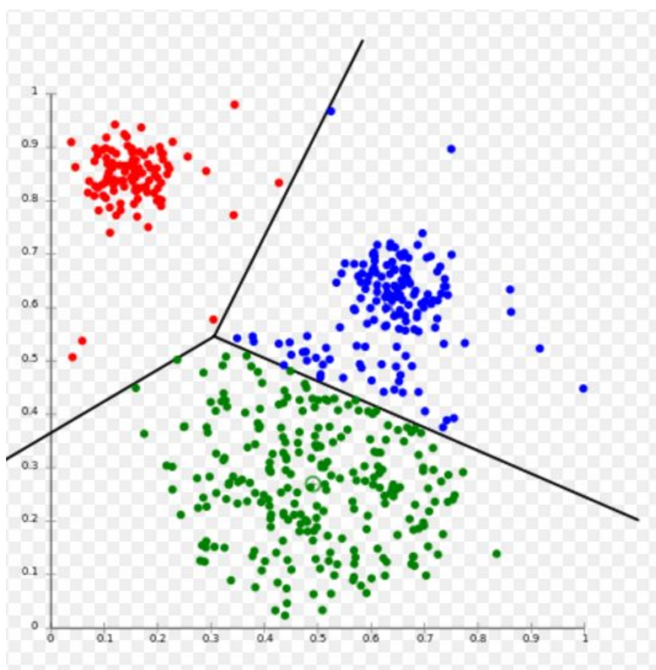


Unsupervised Learning Algorithms

initializing k different centroids $\{\boldsymbol{\mu}^{(1)}, \ldots, \boldsymbol{\mu}^{(k)}\}$

Loop

1) each training examples is assigned to cluster with nearest centroid
2) each centroid is updated to the mean of all training examples in that centroid



Stochastic Gradient Descent

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x},\mathrm{y}\sim\hat{p}_{\mathrm{data}}} L(\boldsymbol{x}, y, \boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{x}^{(i)}, y^{(i)}, \boldsymbol{\theta}).$$

Sample minibatch of examples drawn uniformly from training set.

Typical minibatch size from 1 to few hundred

Building a Machine Learning Algorithm

Data + cost function + an optimization procedure + model