Logistic / Linear Regression

COMP3314 — Lecture 3

Lingpeng Kong
Department of Computer Science, The University of Hong Kong

Based on: Probabilistic Machine Learning by Kevin Murphy
Slides from: Saw Shier Nee with special thanks!

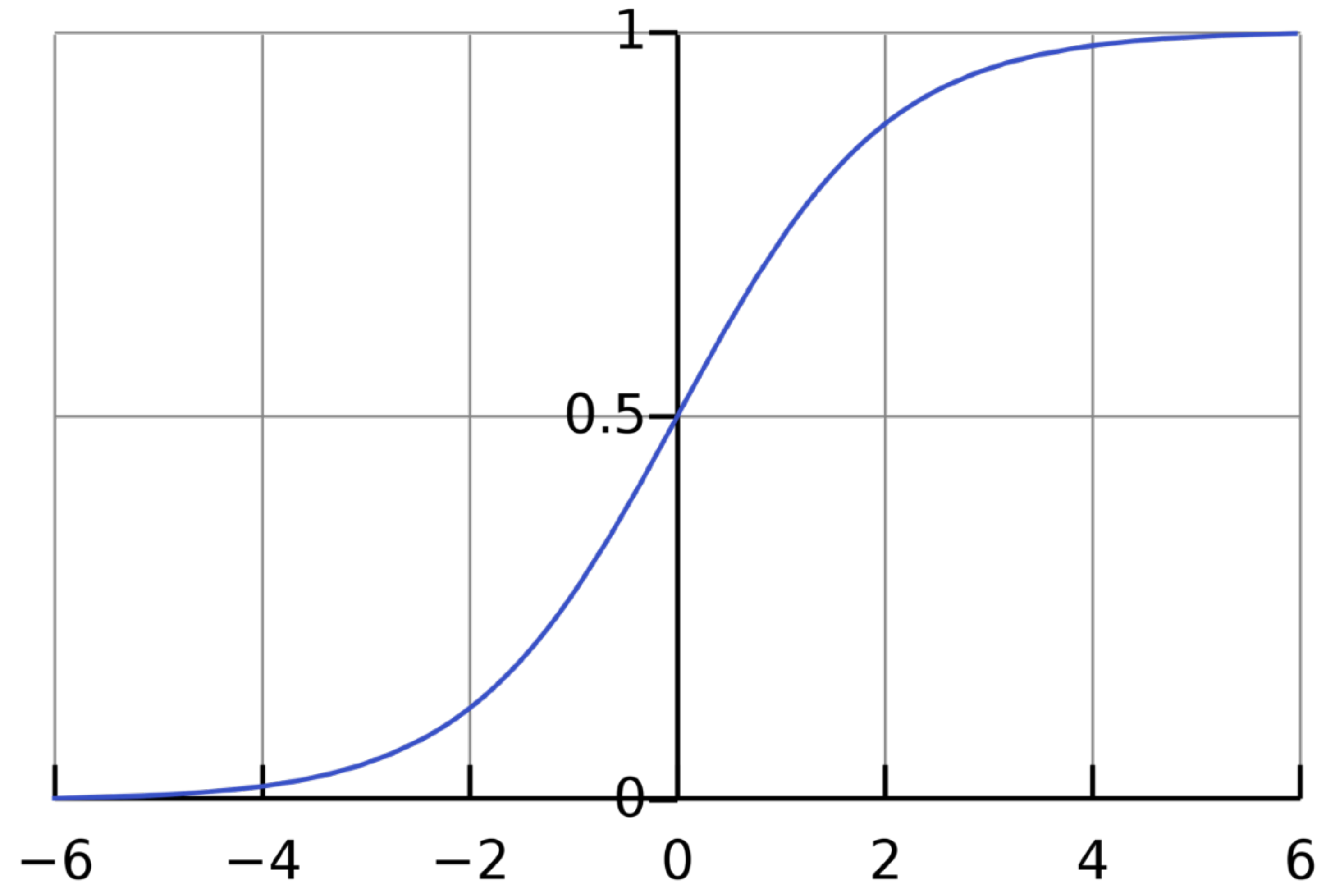
Binary Logistic Regression

$$p(y|\boldsymbol{x};\boldsymbol{\theta}) = \mathrm{Ber}(y|\boldsymbol{\sigma}(\boldsymbol{w}^\mathsf{T}\boldsymbol{x} + b))$$

sigmoid function
(map into 0~1)

$$p(y=1|\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\sigma}(a) = \frac{1}{1+e^{-a}}$$

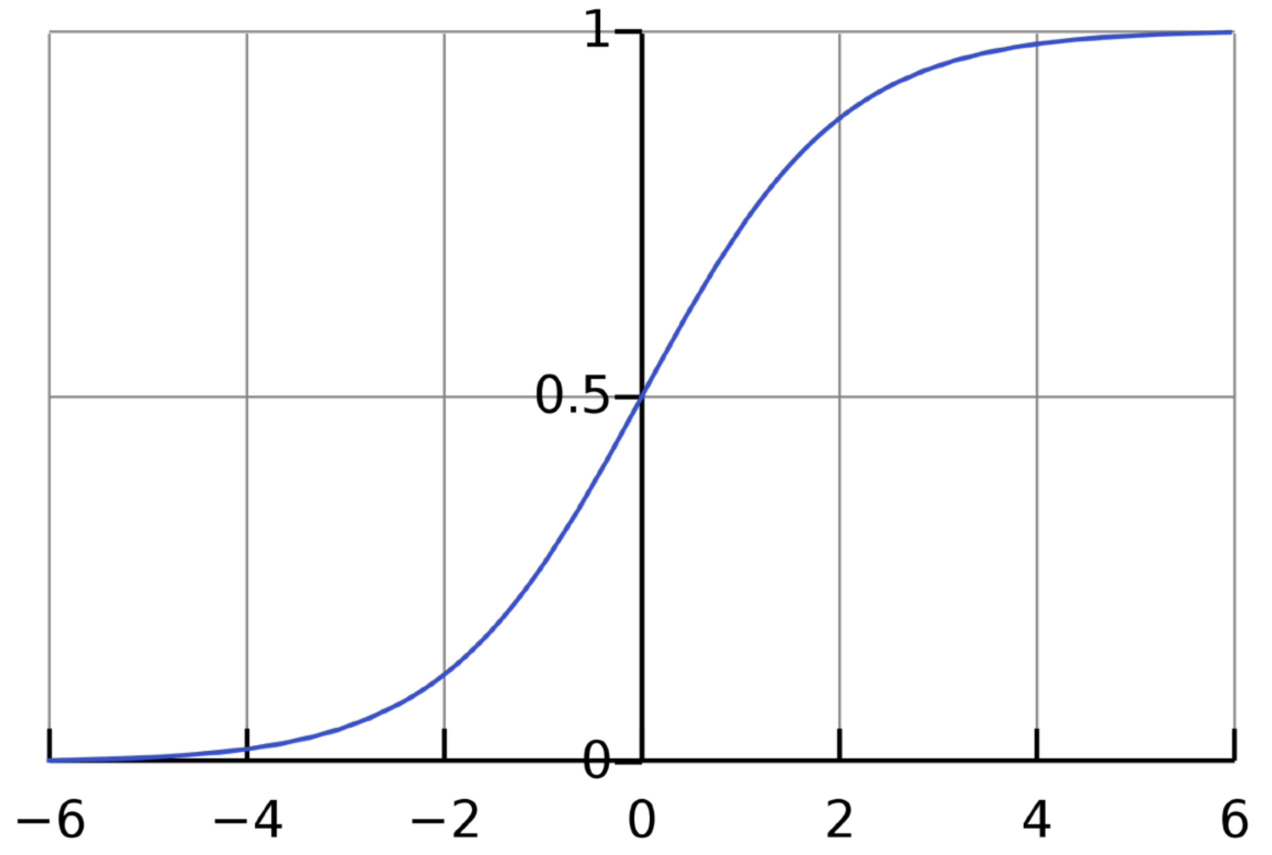
a: logit



sigmoid function

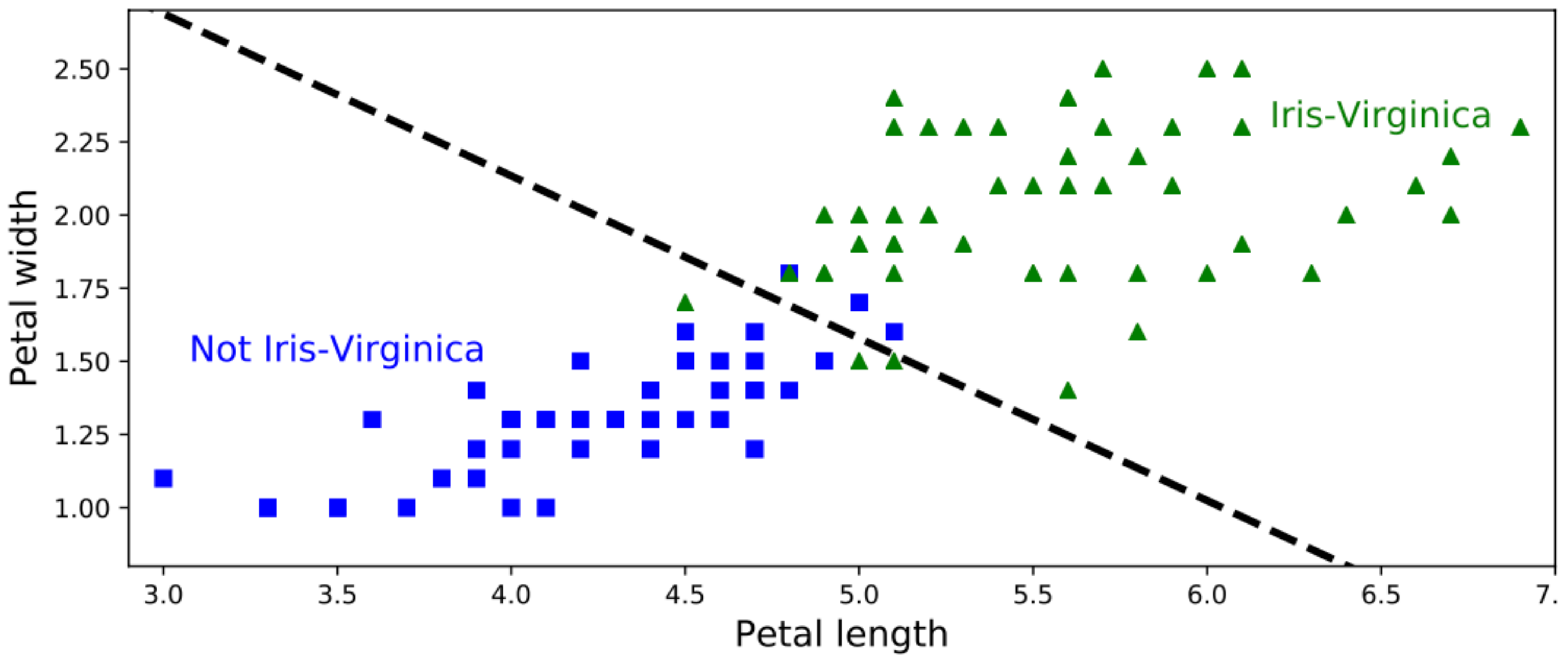
Linear Classifiers

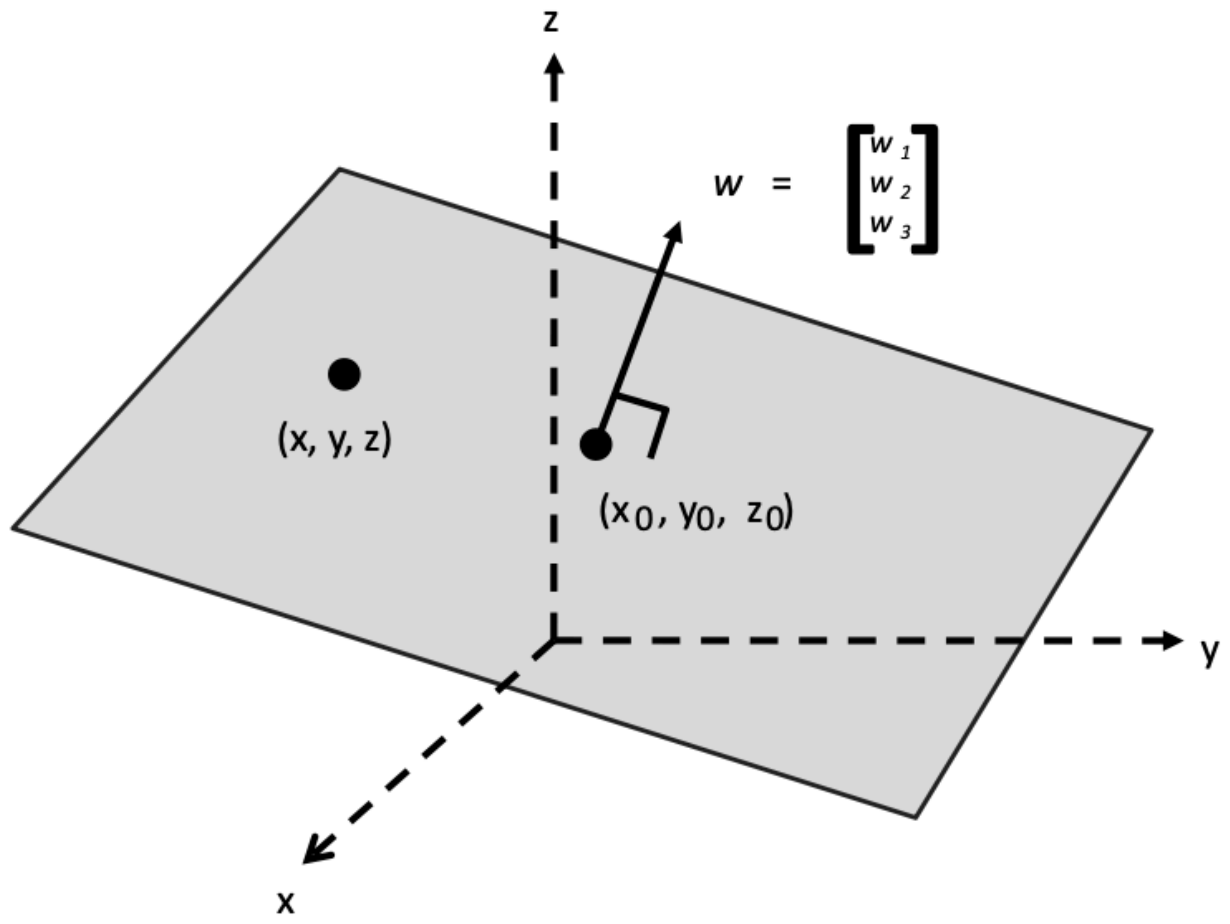
$$f(\boldsymbol{x}) = \mathbb{I}\left(p(y=1|\boldsymbol{x}) > p(y=0|\boldsymbol{x})\right) = \mathbb{I}\left(\log \frac{p(y=1|\boldsymbol{x})}{p(y=0|\boldsymbol{x})} > 0\right) = \mathbb{I}\left(a > 0\right)$$



$$a = \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x} + b$$

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = b + \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x} = b + \sum_{d=1}^{D} w_d x_d$$





linear hyperplane

Nonlinear Classifiers

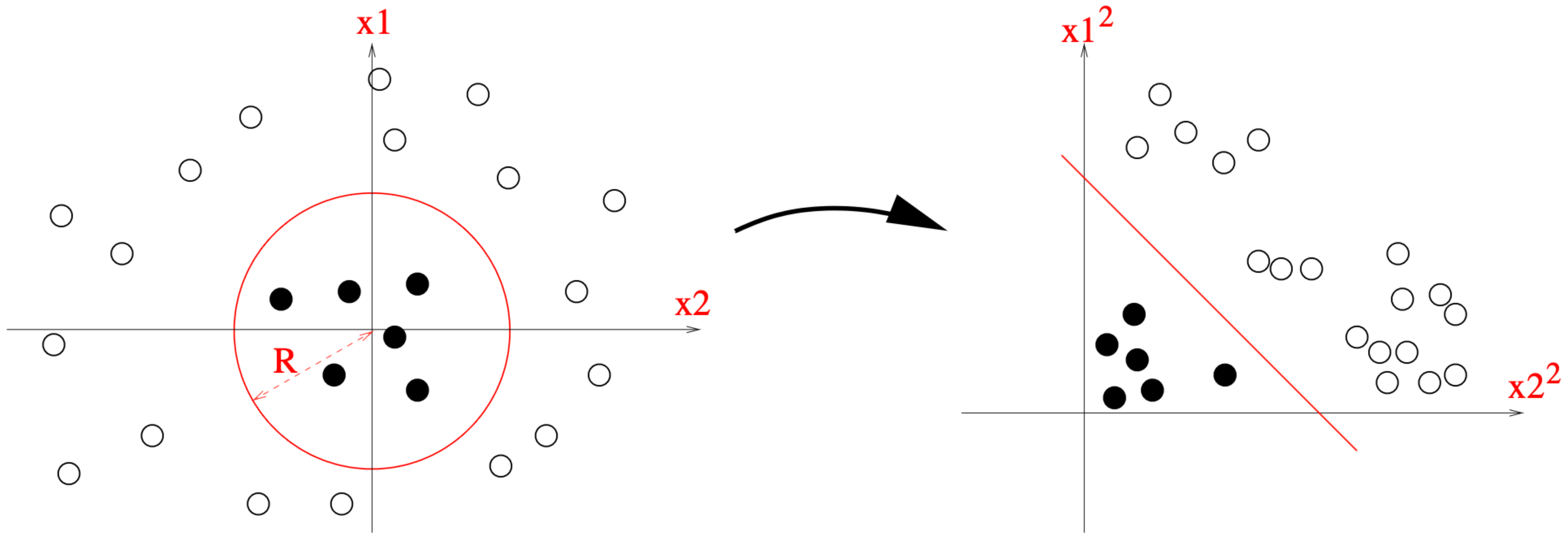
$$f(\boldsymbol{x};\boldsymbol{\theta}) = b + \boldsymbol{w}^\top \boldsymbol{x} = b + \sum_{d=1}^{D} w_d x_d$$

preprocessing the inputs / transform the feature vector

$$\phi(x_1, x_2) = [1, x_1^2, x_2^2]$$

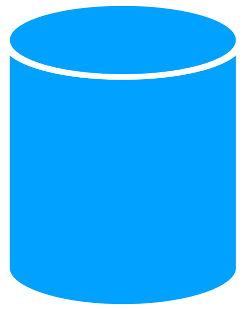
say $\boldsymbol{w} = [-R^2, 1, 1]$

we have $\boldsymbol{w}^\top \phi(\boldsymbol{x}) = x_1^2 + x_2^2 - R^2$



Training Objective

$$p(\mathcal{D} \mid \boldsymbol{\theta}) = p(\boldsymbol{y}_1 \mid \boldsymbol{x}_1) \times p(\boldsymbol{y}_2 \mid \boldsymbol{x}_2) \times \ldots \times p(\boldsymbol{y}_n \mid \boldsymbol{x}_n)$$

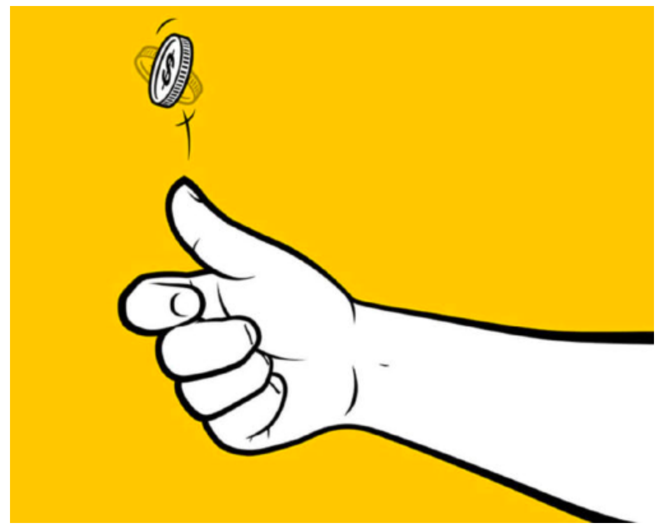


$$\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$$

training dataset

Why's that?

iid — independent and identically distributed random variables



$$\hat{\theta} = \arg\max_{\theta} P(D \mid \theta)$$

MLE

$$\hat{\theta} = \arg\max_{\theta} P(\theta \mid D) = \arg\max_{\theta} \frac{P(D \mid \theta) P(\theta)}{P(D)}$$

MAP

Maximum Likelihood Estimation

$$p(\mathcal{D} \mid \boldsymbol{\theta}) \longrightarrow \arg\min_{\boldsymbol{\theta}}[-\log P(\mathcal{D} \mid \boldsymbol{\theta})]$$

$$\mathrm{NLL}(\boldsymbol{w}) = -\frac{1}{N}\log p(\mathcal{D}|\boldsymbol{w}) = -\frac{1}{N}\log\prod_{n=1}^{N}\mathrm{Ber}(y_n|\mu_n)$$

negative log likelihood

$$\mu_n = \boldsymbol{\sigma}(a_n) \longrightarrow p(y=1 \mid \boldsymbol{x})$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\log[\mu_n^{y_n} \times (1-\mu_n)^{1-y_n}]$$

$$= -\frac{1}{N}\sum_{n=1}^{N}[y_n\log\mu_n + (1-y_n)\log(1-\mu_n)]$$

$$= \frac{1}{N}\sum_{n=1}^{N}\mathbb{H}(y_n, \mu_n)$$

$$\mathbb{H}(p,q) = -\left[p\log q + (1-p)\log(1-q)\right]$$

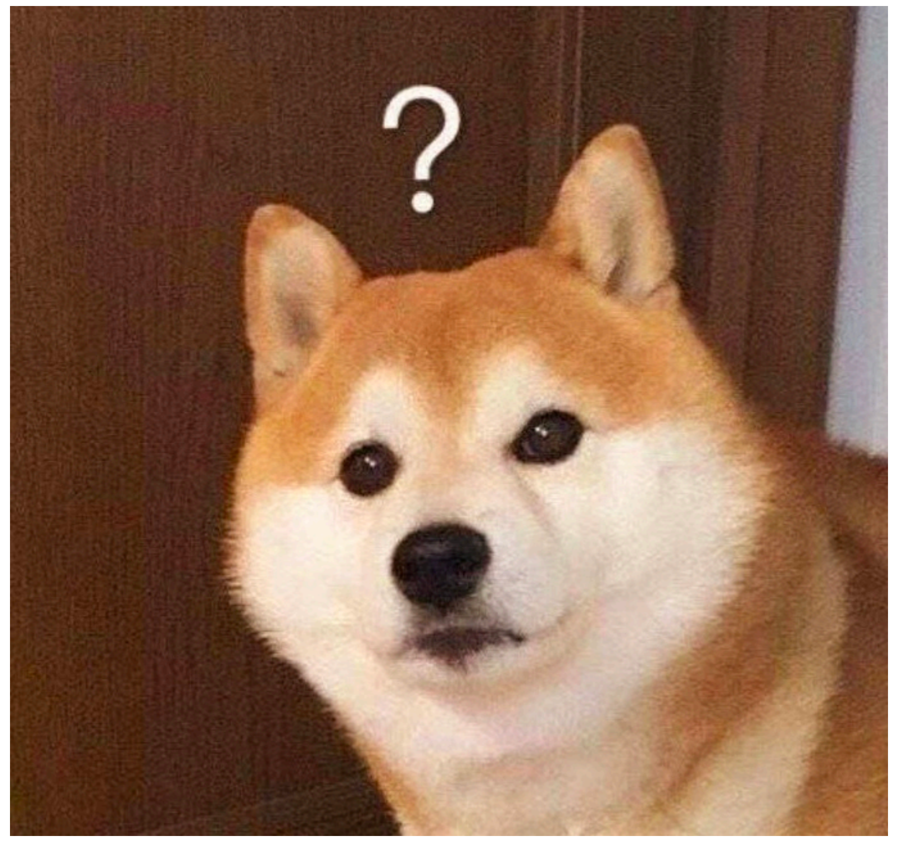
binary cross entropy

Maximum Likelihood Estimation

MLE: Choose θ that maximizes the probability of observed data

$$\hat{\theta} = \arg\max_{\theta} P(D \mid \theta)$$

$$= \arg\max_{\theta} \log P(D \mid \theta)$$

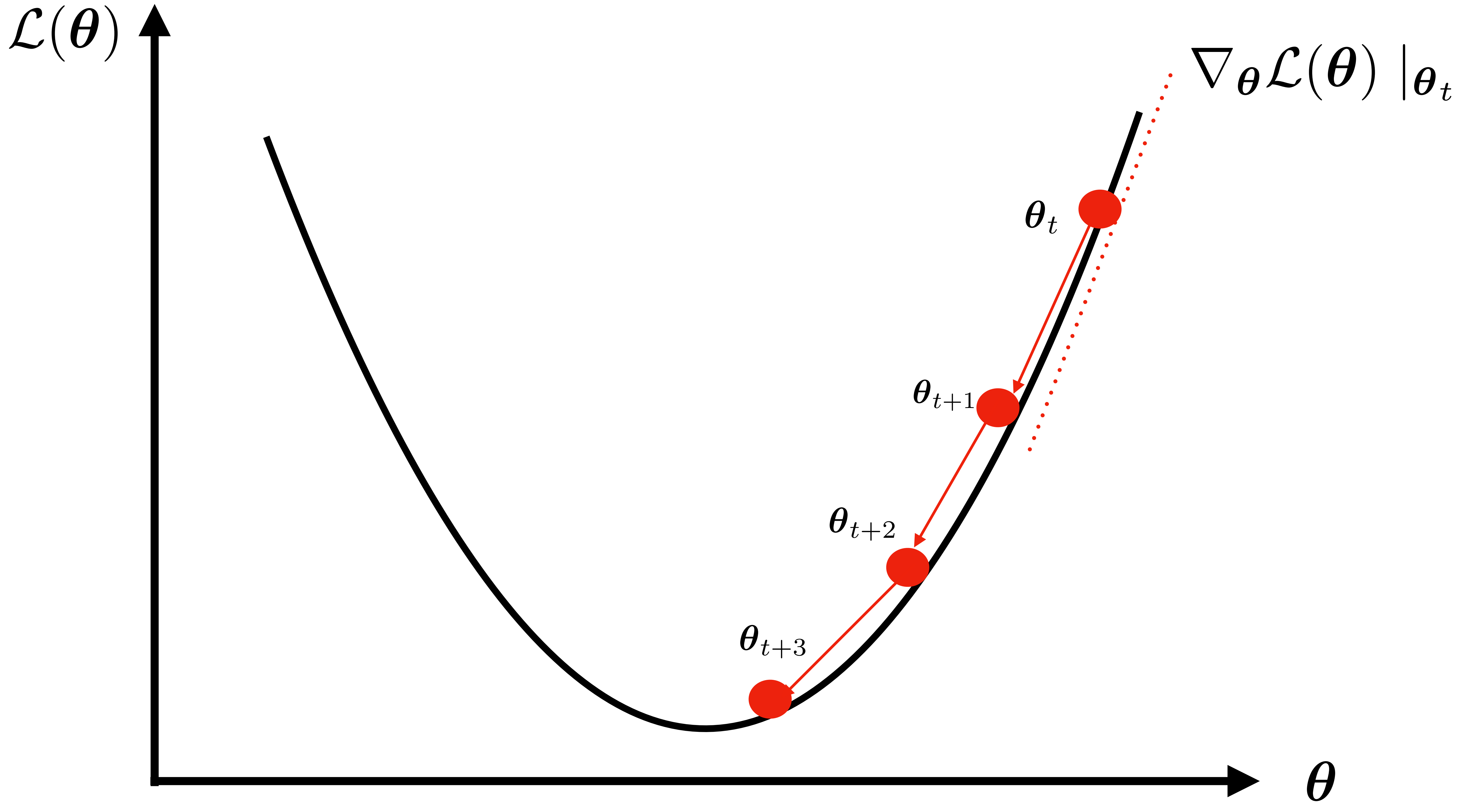


Set derivative to zero

$$\frac{\partial \log P(D \mid \theta)}{\partial \theta} = 0$$

What if this is too difficult to compute?
Is there a more general way?

Gradient Descent



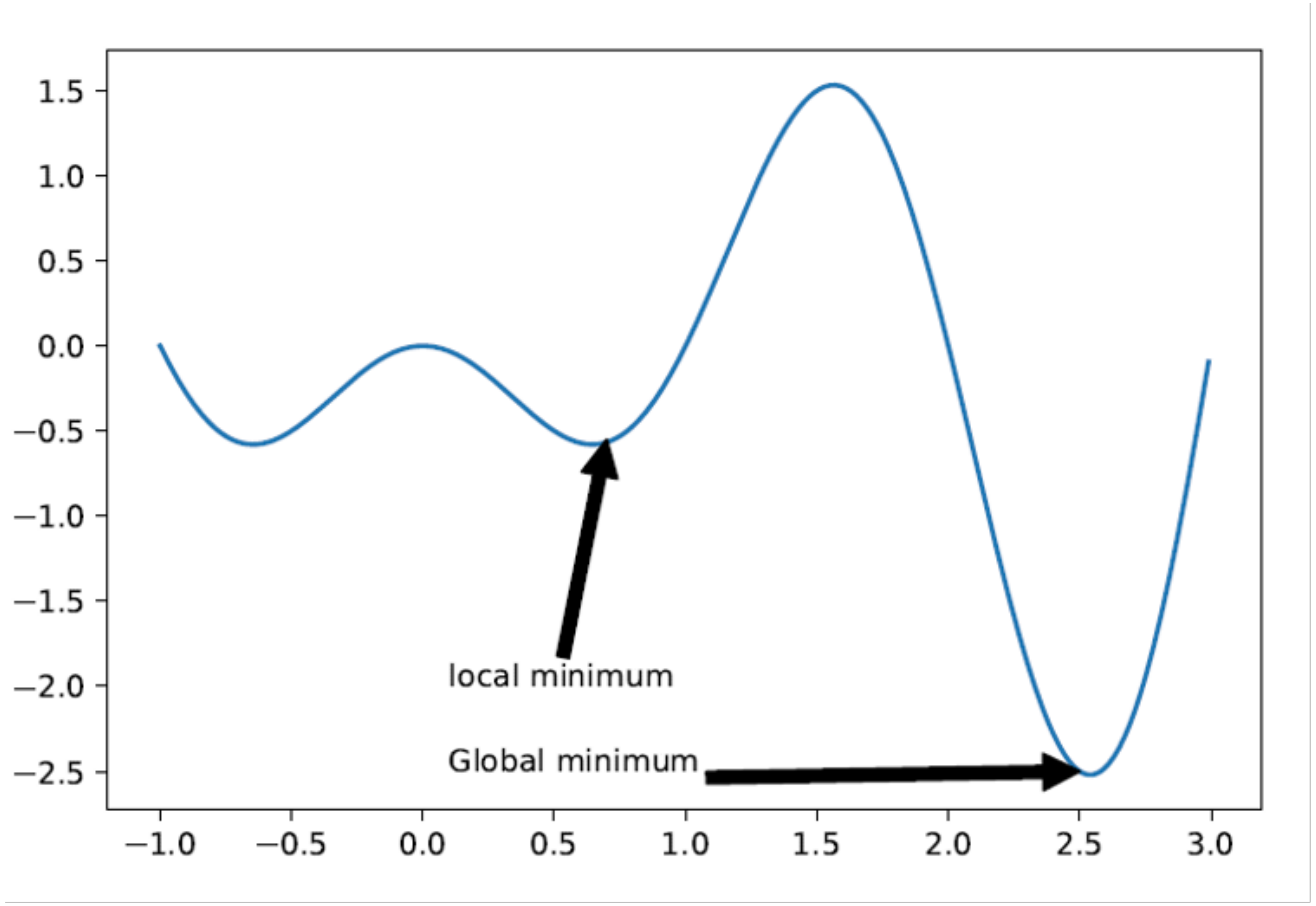
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}_t}$$

negative gradient (descent direction)

step size
(learning rate)

Global / Local Optimum

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$



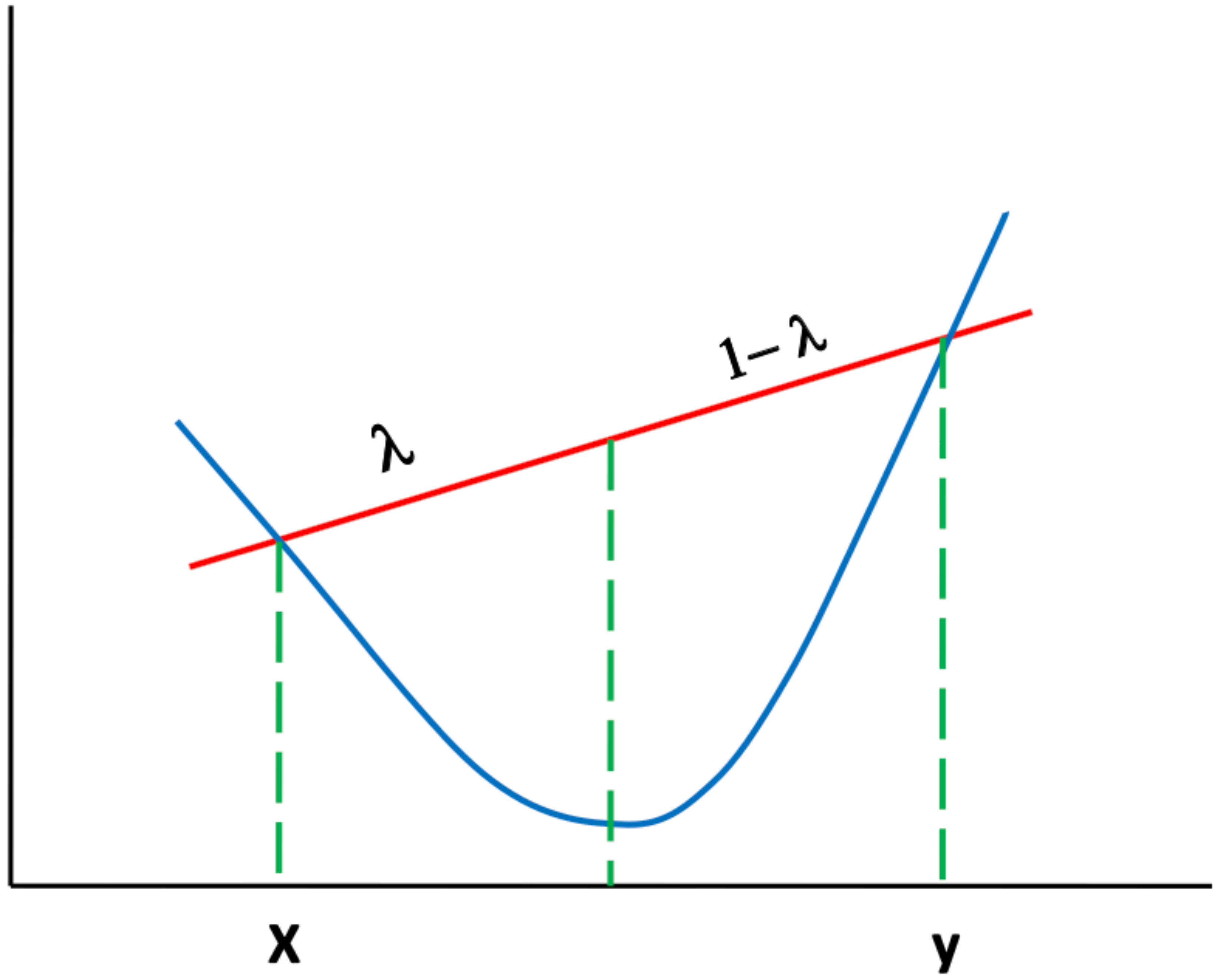
Convex Function

We say f is a **convex function** if its **epigraph** (the set of points above the function, illustrated in Figure 8.4a) defines a convex set. Equivalently, a function $f(\boldsymbol{x})$ is called convex if it is defined on a convex set and if, for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{S}$, and for any $0 \leq \lambda \leq 1$, we have

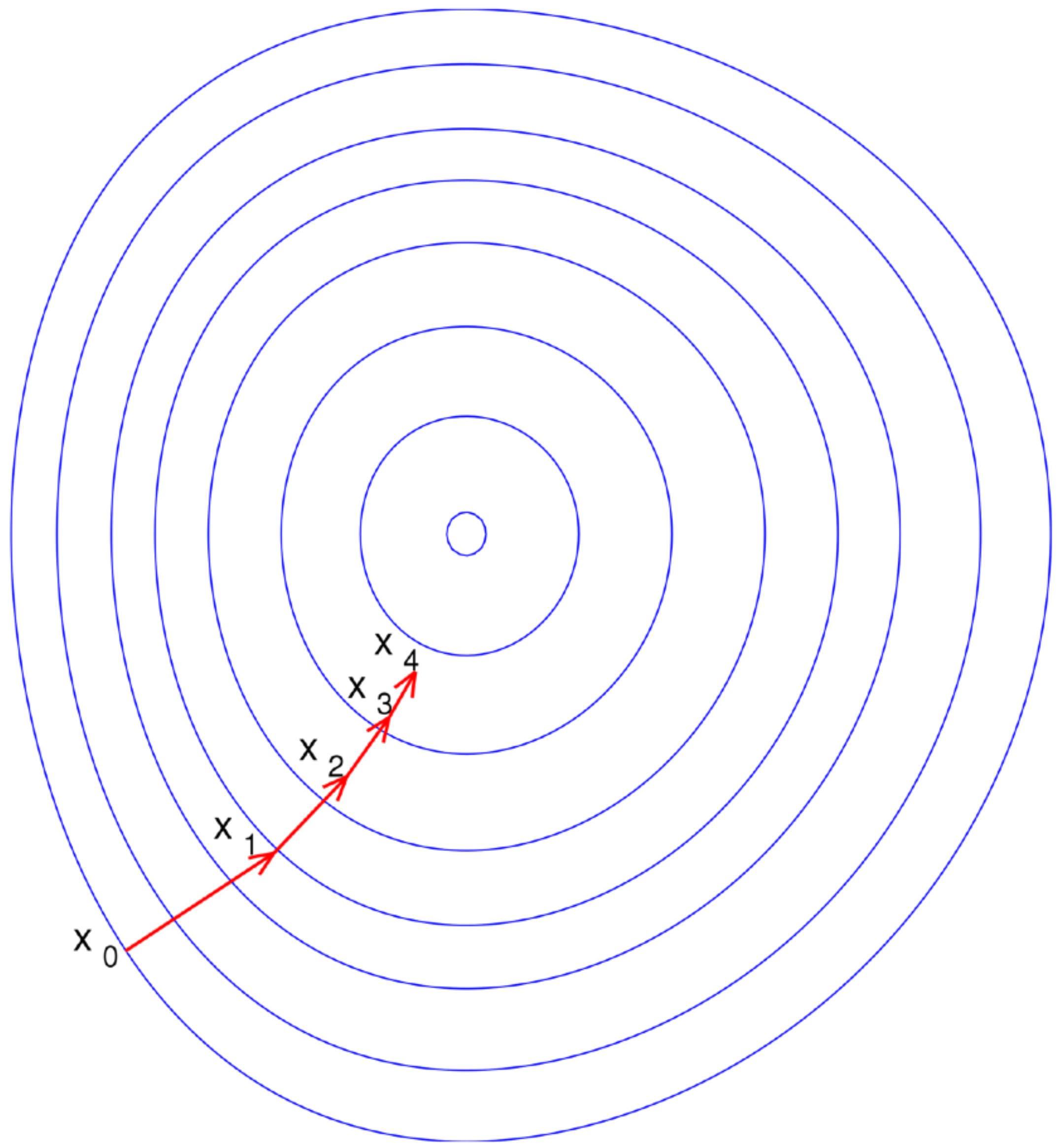
$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}) \tag{8.7}$$

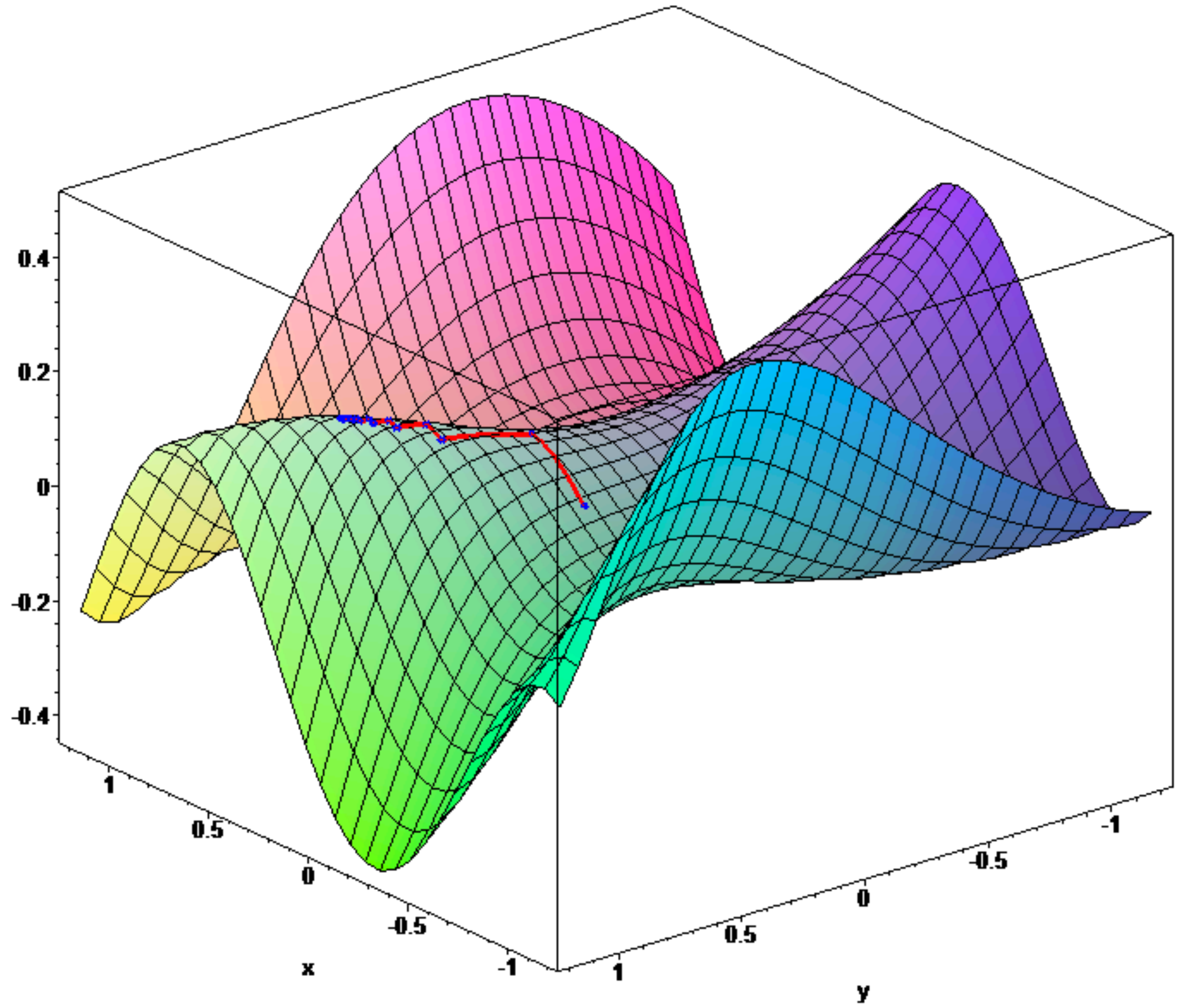
Y-axis value that fell between the domain from x and y

Straight line between (x, f(x)) and (y, f(y)) as a function of λ



Gradient Descent





fog in the mountains

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}_t}$$

negative gradient (descent direction)

step size
(learning rate)

Optimizing the Objective

$$\nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{w}) = \mathbf{0}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_t}$$

Let's assume we have two types of feature values here. (+1/-1)

$$\nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mu_n - y_n) \boldsymbol{x}_n$$

— textbook 10.2.3.3 for a detailed derivation

$$\mu_n = \boldsymbol{\sigma}(a_n) \longrightarrow p(y = 1 \mid \boldsymbol{x})$$

$$\arg\min_{\boldsymbol{\theta}} [-\log P(\mathcal{D} \mid \boldsymbol{\theta})] \longrightarrow \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$$

Optimization:

The core problem in machine learning is parameter estimation (aka model fitting).

This requires solving an optimization problem, where we try to find the values for a set of variables, $\theta \in \Theta$, that minimize a scalar-valued loss function or cost function, $L(\theta)$.

Optimizing the Objective

$$\nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{w}) = \mathbf{0}$$

$$\nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} (\mu_n - y_n) \boldsymbol{x}_n$$

— textbook 10.2.3.3 for a detailed derivation

$$\mu_n = \boldsymbol{\sigma}(a_n) \longrightarrow p(y = 1 \mid \boldsymbol{x})$$

gradient weights each input $\boldsymbol{x}_n$ by its error, and then average the results

batch approach — takes into consideration all the training examples

Stochastic Gradient Descent

$$\text{NLL}(\boldsymbol{w}) = -\frac{1}{N}\sum_{n=1}^{N}\left[y_n \log \mu_n + (1 - y_n)\log(1 - \mu_n)\right]$$

$$\nabla_{\boldsymbol{w}}\text{NLL}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mu_n - y_n)\boldsymbol{x}_n$$

$$\mu_n = \boldsymbol{\sigma}(a_n) \longrightarrow p(y = 1 \mid \boldsymbol{x})$$

stochastic gradient descent: use a mini batch of size 1

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla_{\boldsymbol{w}}\text{NLL}(\boldsymbol{w}_t) = \boldsymbol{w}_t - \eta_t(\mu_n - y_n)\boldsymbol{x}_n$$

Perceptron Algorithm

stochastic gradient descent:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla_{\boldsymbol{w}} \mathrm{NLL}(\boldsymbol{w}_t) = \boldsymbol{w}_t - \eta_t(\mu_n - y_n)\boldsymbol{x}_n \qquad \mu_n = \boldsymbol{\sigma}(a_n) \longrightarrow p(y = 1 \mid \boldsymbol{x})$$

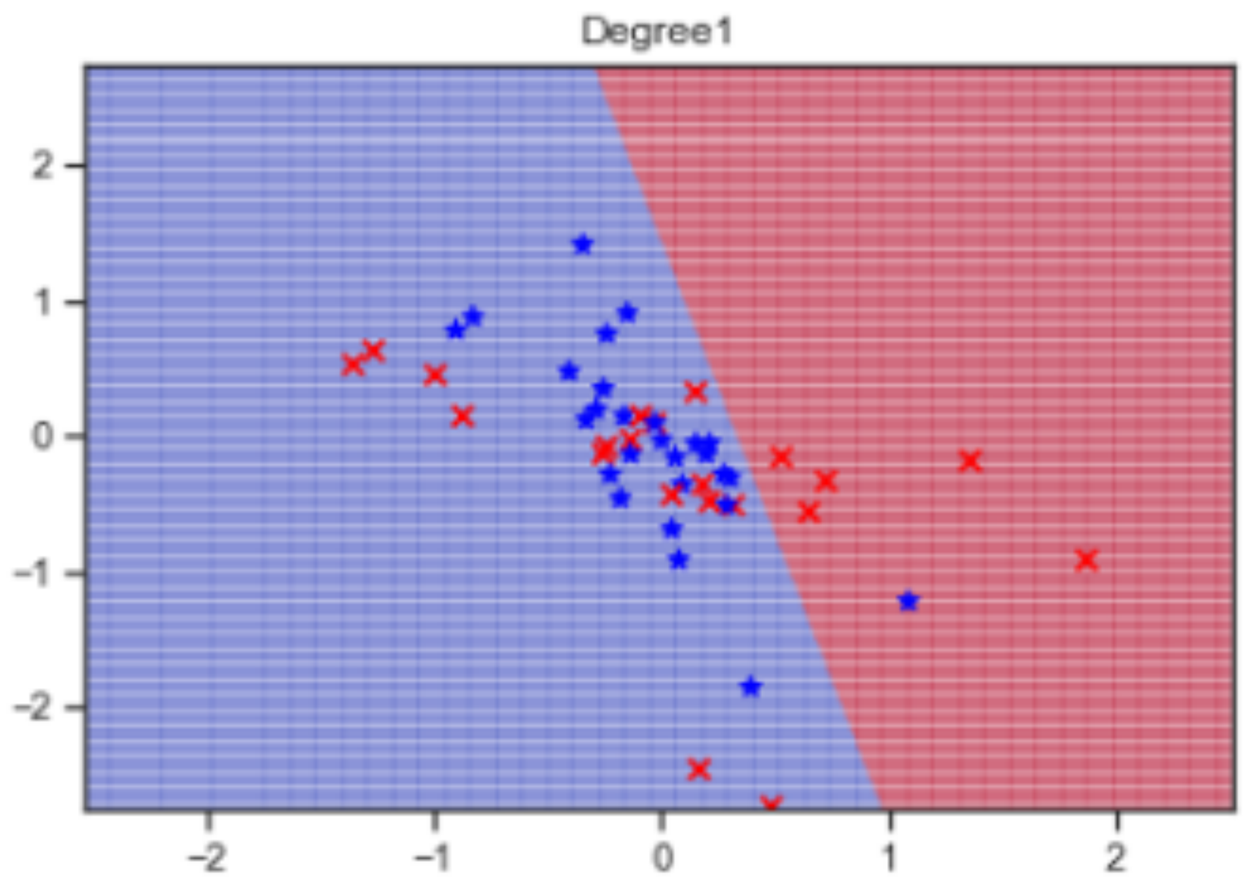
perceptron algorithm:

$$f(\boldsymbol{x}_n; \boldsymbol{\theta}) = \mathbb{I}\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n + b > 0\right)$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t(\hat{y}_n - y_n)\boldsymbol{x}_n$$

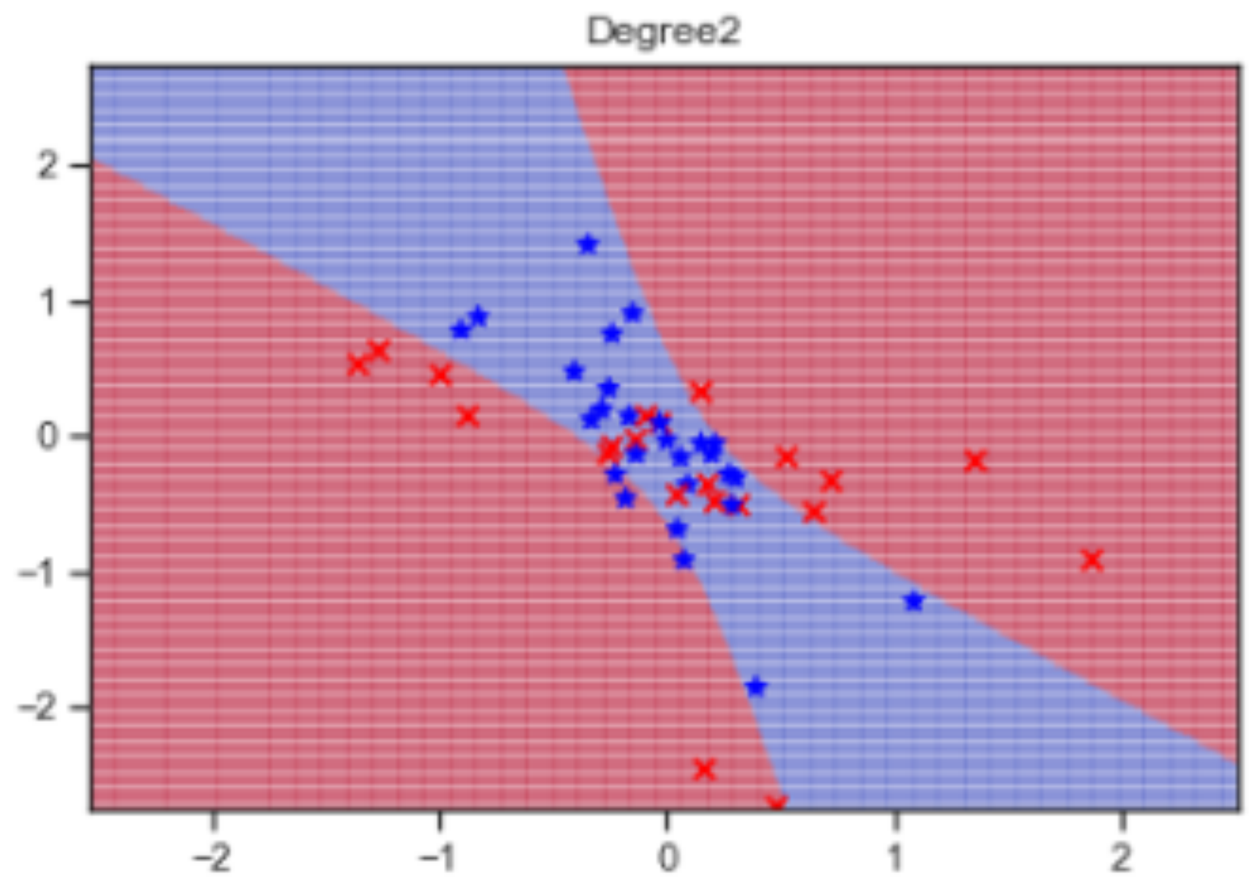
replace the soft probabilities $\mu_n = p(y_n = 1|\boldsymbol{x}_n)$ with hard labels $\hat{y}_n = f(\boldsymbol{x}_n)$

Overfitting



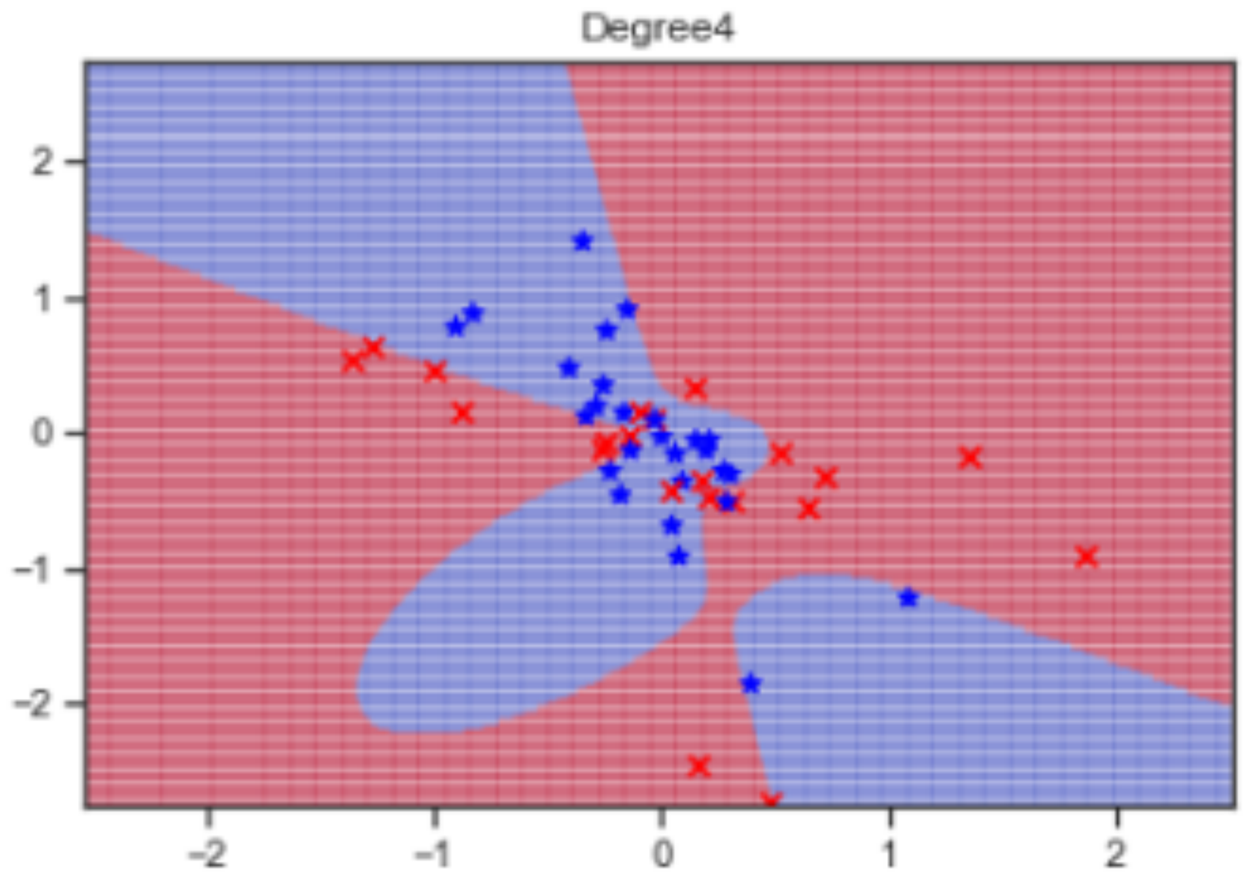
$\hat{w} = [0.51291712, 0.11866937]$

(a)



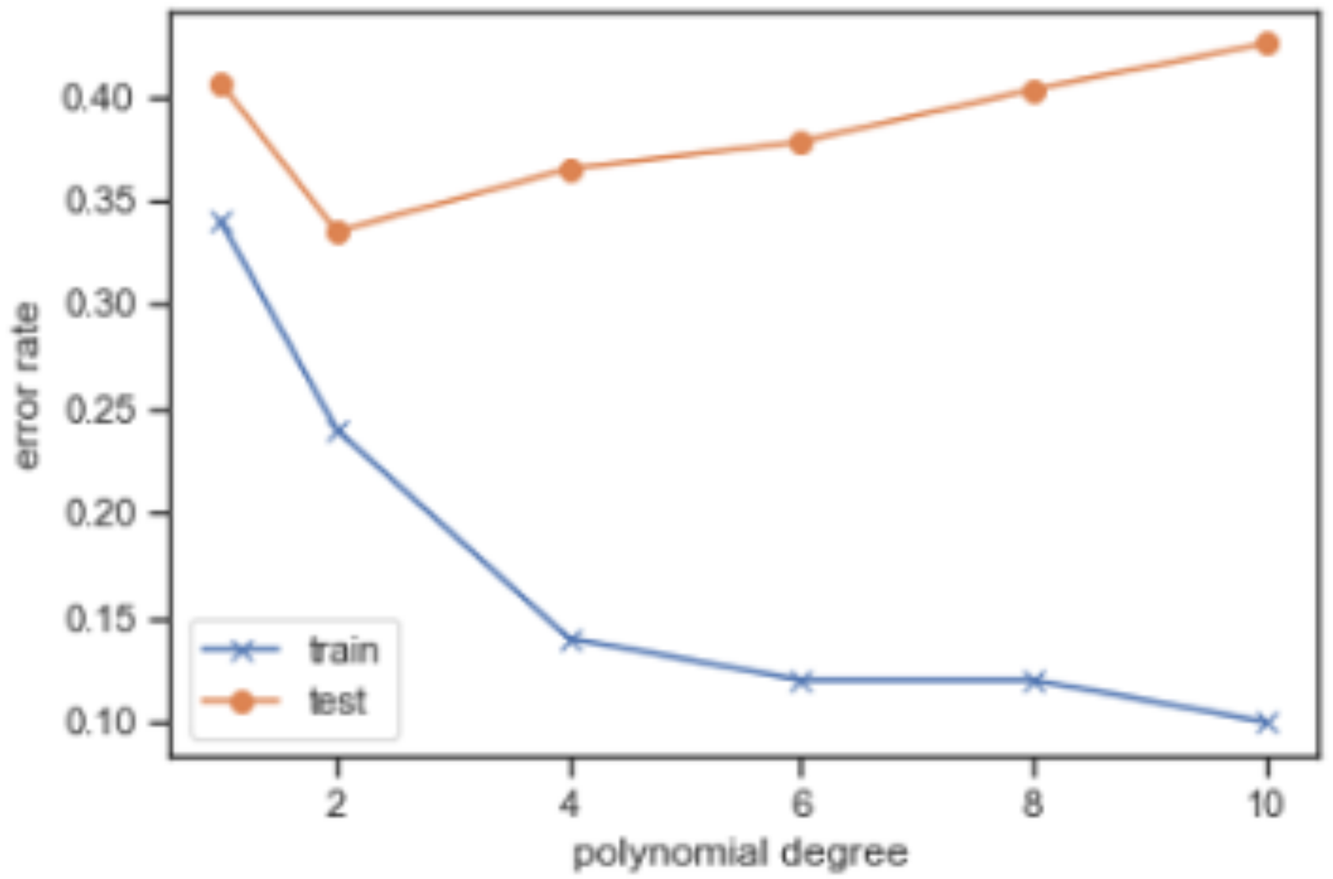
$\hat{w} = [2.27510513, 0.05970325, 11.84198867, 15.40355969, 2.51242311]$

(b)



$\hat{w} = [-3.07813766, \cdots, -59.03196044, 51.77152431, 10.25054164]$

(c)



(d)

See any trend?

As degree increases, w increase / decrease?

Overfitting

Reduce Overfitting:

Do not let the weight to grow too big

Add regularizer to the objective function as penalty

$$\mathcal{L}(\boldsymbol{w}) = \text{NLL}(\boldsymbol{w}) + \lambda ||\boldsymbol{w}||_2^2$$

L2 regularization / weight decay

Standardization

$$\text{standardize}(x_{nd}) = \frac{x_{nd} - \hat{\mu}_d}{\hat{\sigma}_d}$$

$$\hat{\mu}_d = \frac{1}{N} \sum_{n=1}^{N} x_{nd}$$

$$\hat{\sigma}_d^2 = \frac{1}{N} \sum_{n=1}^{N} (x_{nd} - \hat{\mu}_d)^2$$

Overfitting

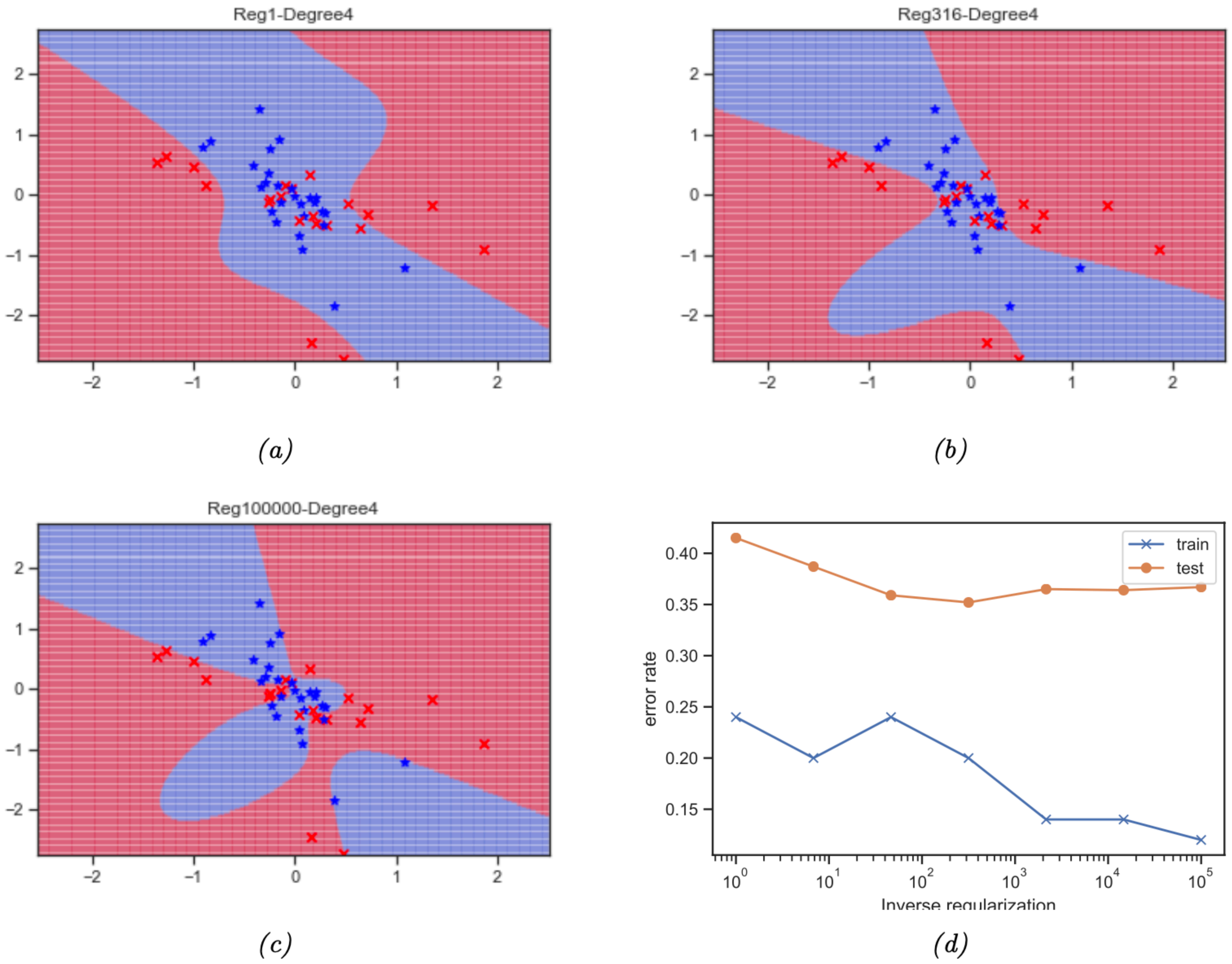


Figure 10.6: Weight decay with variance C applied to two-class, two-dimensional logistic regression problem with a degree 4 polynomial. (a) $C = 1$. (b) $C = 316$. (c) $C = 100,000$. (d) Train and test error vs C. Generated by code at figures.probml.ai/book1/10.6.

$$\mathcal{L}(\boldsymbol{w}) = \mathrm{NLL}(\boldsymbol{w}) + \lambda||\boldsymbol{w}||_2^2$$

$$\lambda = \frac{1}{C}$$

Recap: Bayesian Learning

$$P(\theta \mid D) = \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

posterior

$$\arg\max_{\theta} P(\theta \mid D) = \arg\max_{\theta} \frac{P(D \mid \theta)P(\theta)}{P(D)}$$

$$= \arg\max_{\theta} P(D \mid \theta)P(\theta)$$

$$P(D \mid \theta)$$

likelihood

$$P(\theta)$$

prior

Overfitting

$$\mathcal{L}(\boldsymbol{w}) = \text{NLL}(\boldsymbol{w}) + \lambda ||\boldsymbol{w}||_2^2$$

MLE:

$$p(\mathcal{D} \mid \boldsymbol{\theta}) \longrightarrow \arg\min_{\boldsymbol{\theta}} [-\log P(\mathcal{D} \mid \boldsymbol{\theta})]$$

MAP:

$$p(\boldsymbol{\theta} \mid \mathcal{D}) \longrightarrow \arg\min_{\boldsymbol{\theta}} [-[\log p(\boldsymbol{\theta}) + \log p(\mathcal{D} \mid \boldsymbol{\theta})]]$$

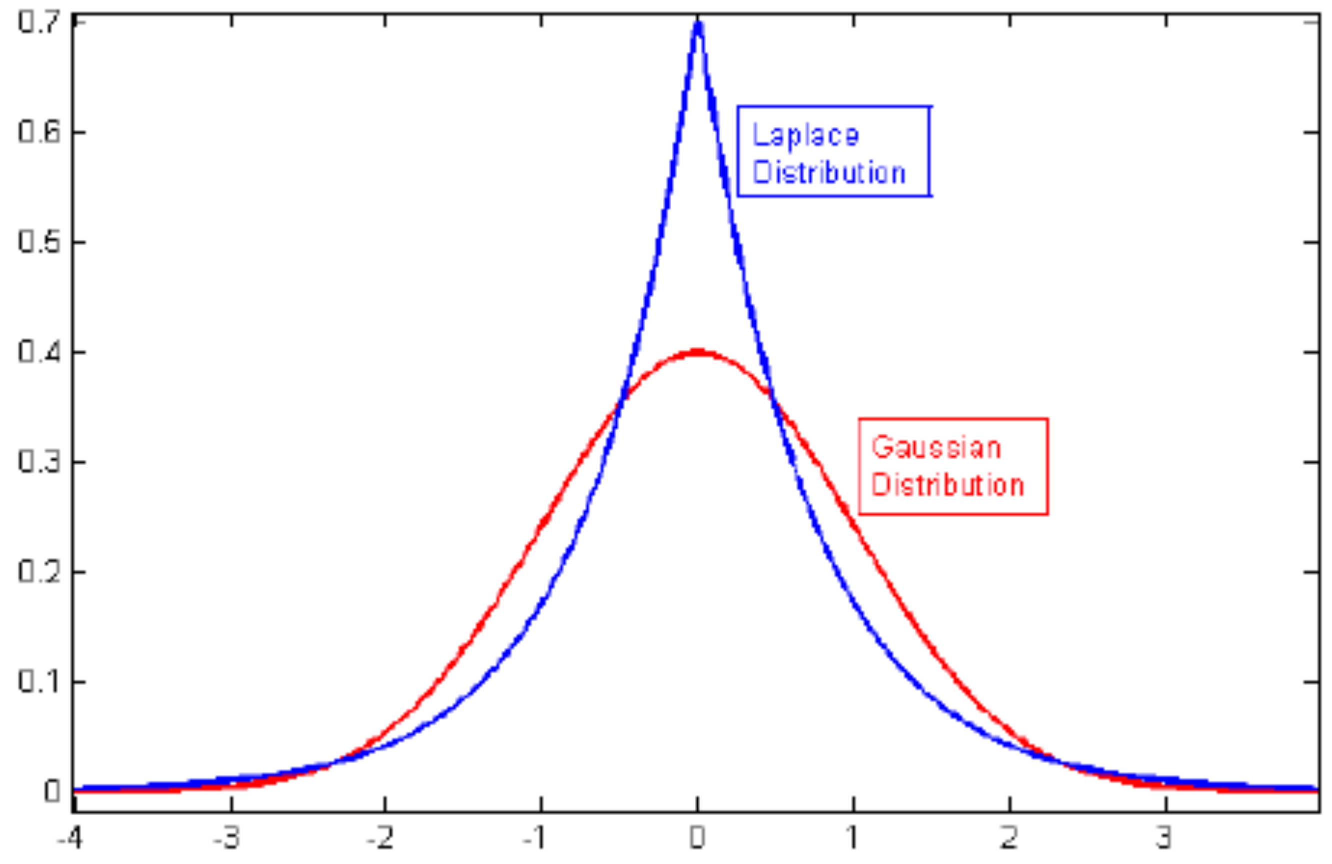
$$\nabla_{\boldsymbol{w}} \text{PNLL}(\boldsymbol{w}) = \boldsymbol{g}(\boldsymbol{w}) + 2\lambda \boldsymbol{w}$$

Guassian:

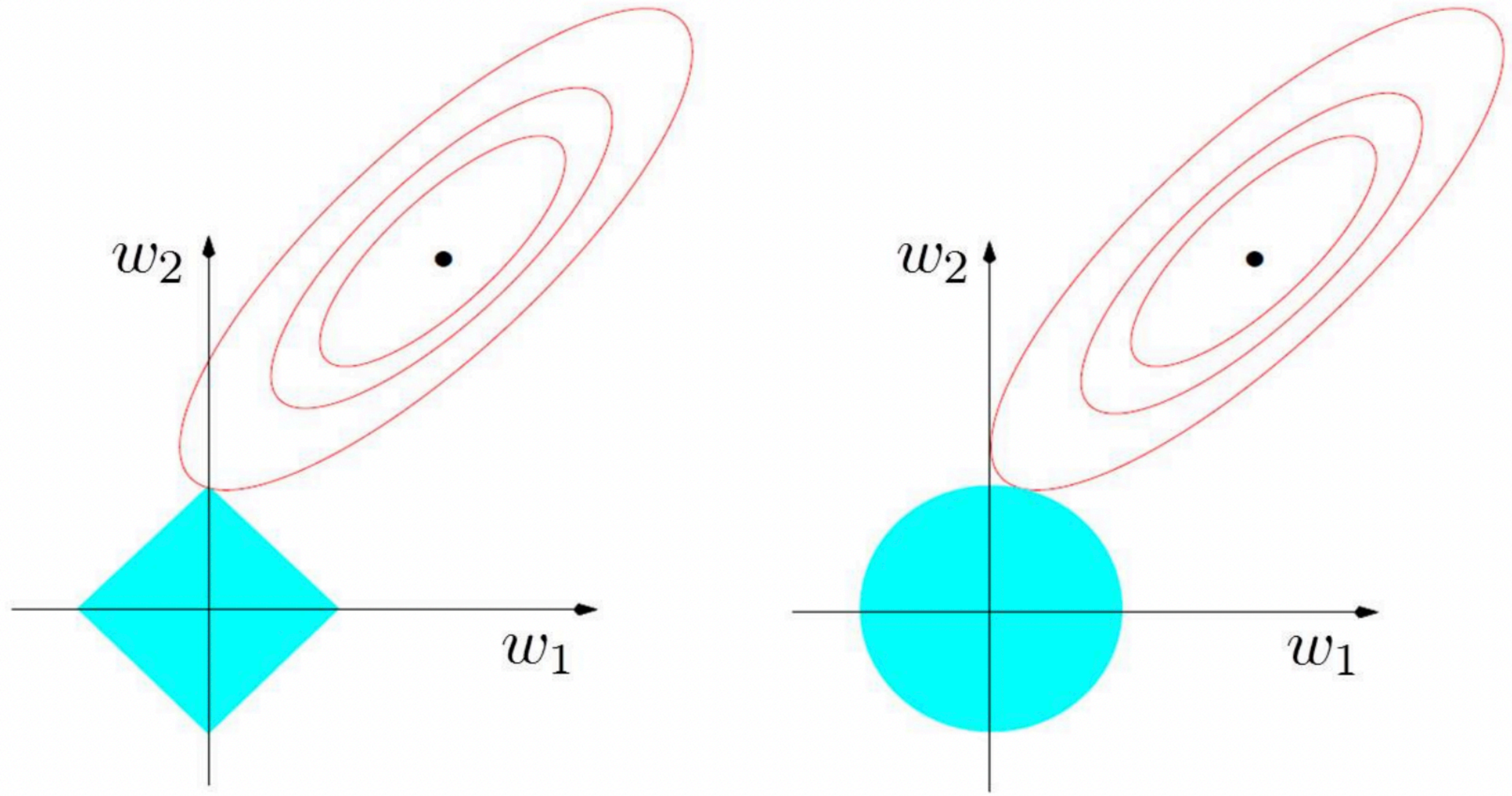
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Laplace:

$$f(x|\mu, b) = \frac{1}{2b} \exp\left(-\frac{|x-\mu|}{b}\right)$$



Regularizers and Sparsity



$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2 \quad \text{subject to } \|\mathbf{w}\|_1 \leq \tau$$

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2 \quad \text{subject to } \|\mathbf{w}\|_2^2 \leq \tau$$

Maximum Entropy Classifiers

multinomial logistic regression

$$p(y = c|\boldsymbol{x}, \mathbf{W}) = \frac{\exp(\boldsymbol{w}_c^\mathsf{T}\boldsymbol{x})}{Z(\boldsymbol{w}, \boldsymbol{x})} = \frac{\exp(\boldsymbol{w}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c'=1}^{C}\exp(\boldsymbol{w}_{c'}^\mathsf{T}\boldsymbol{x})}$$

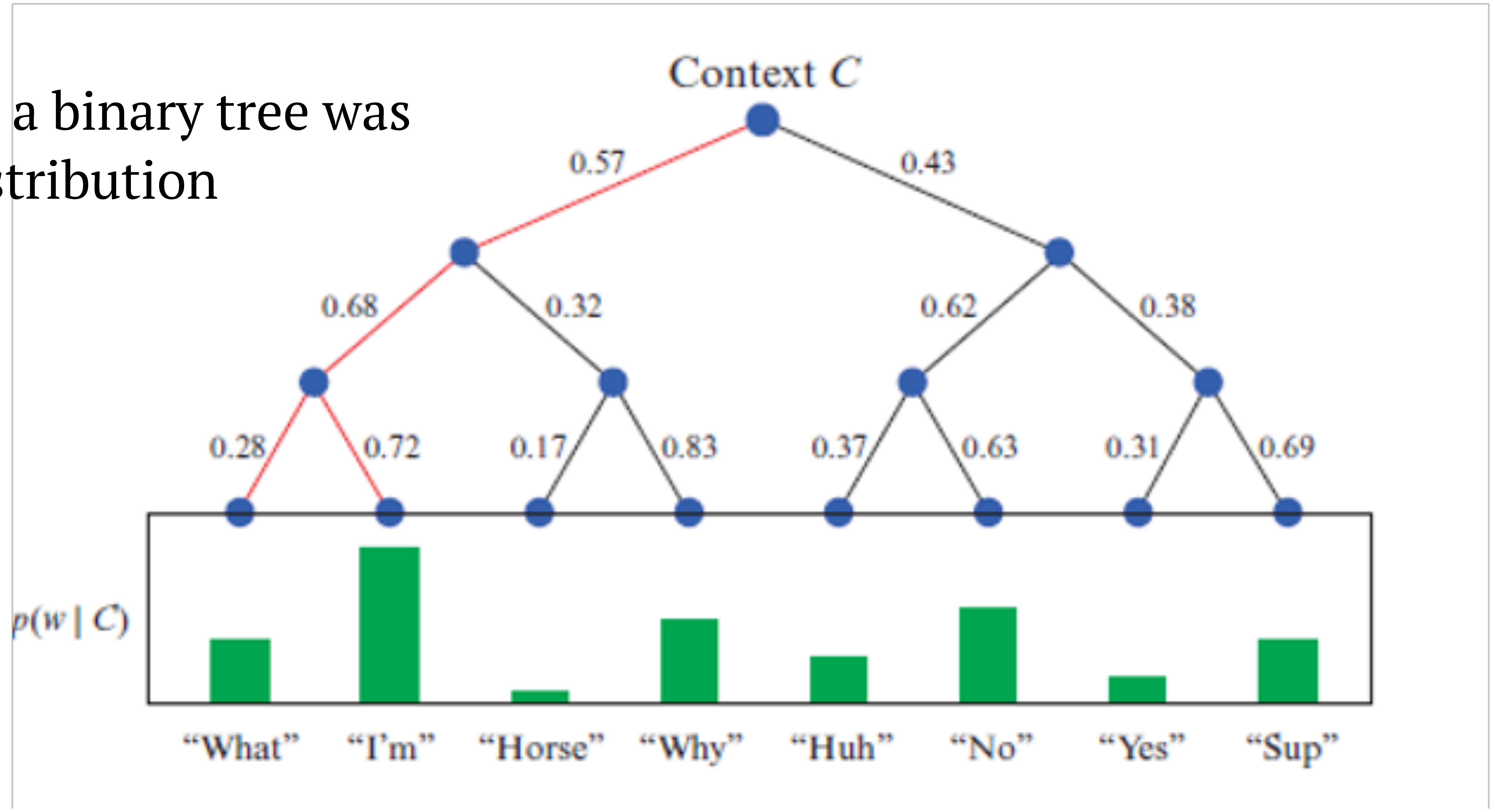
↑ softmax function

Handling Large Number of Classes

Using regular softmax function, when the number of classes, *C* increases, computational cost to compute H increases

To facilitate this, we can use hierarchy softmax

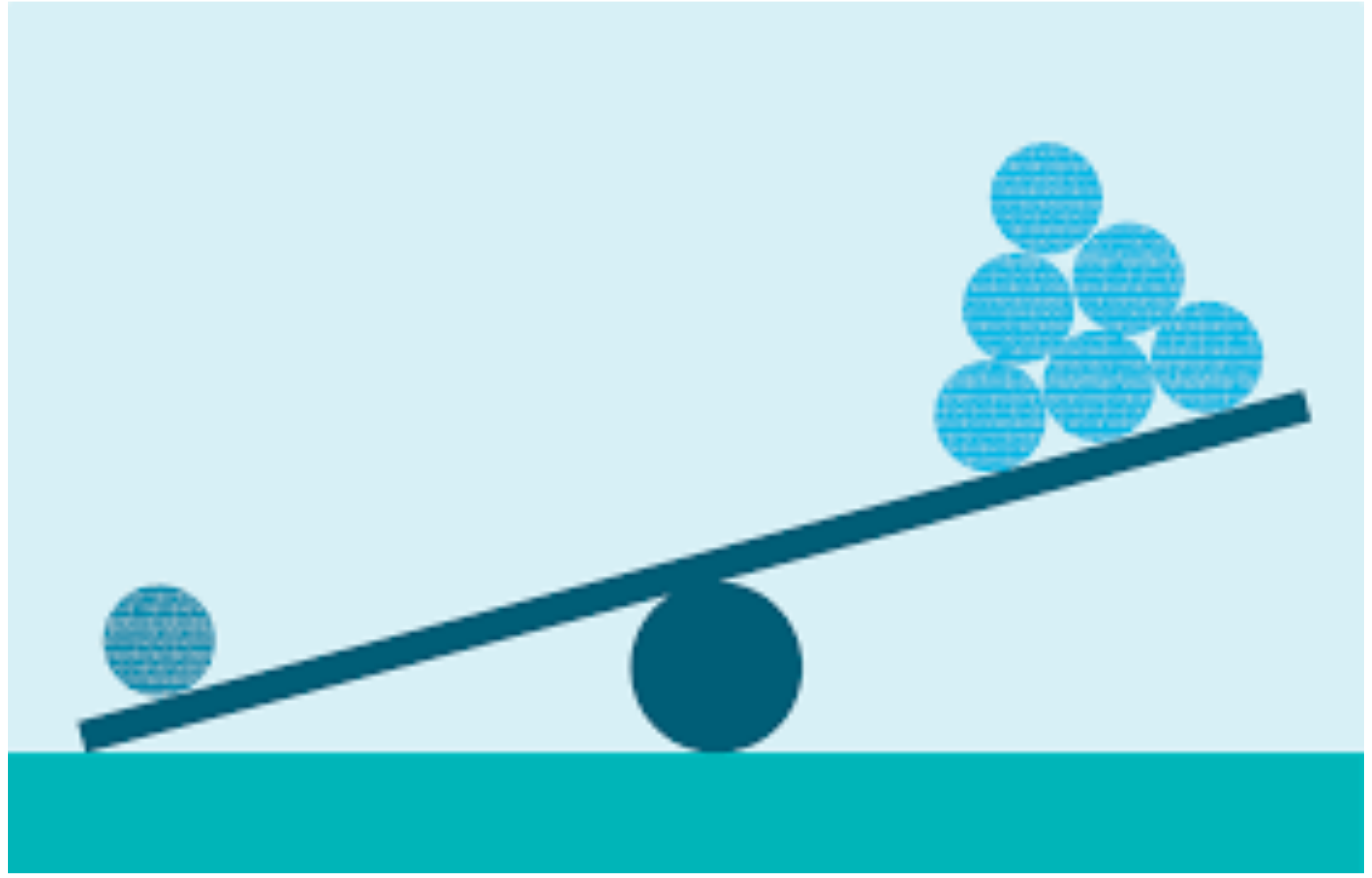
The idea behind decomposing the output layer to a binary tree was to reduce the complexity to obtain probability distribution



Handling Imbalance Class

More attention on more ‘common’ dataset
Less attention on ‘rare’ dataset

Approach
Resample the data – Oversample / Undersample



Linear Regression

Linear Regression

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + \boldsymbol{w}^\top \boldsymbol{x}, \sigma^2)$$

$$\boldsymbol{\theta} = (w_0, \boldsymbol{w}, \sigma^2)$$

Logistic Regression

$$p(y|\boldsymbol{x}; \boldsymbol{\theta}) = \text{Ber}(y|\boldsymbol{\sigma}(\boldsymbol{w}^\top \boldsymbol{x} + b))$$

If input is 1-D, simple linear regression

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

If input is N-D/output is N-D, multiple/multivariate linear regression

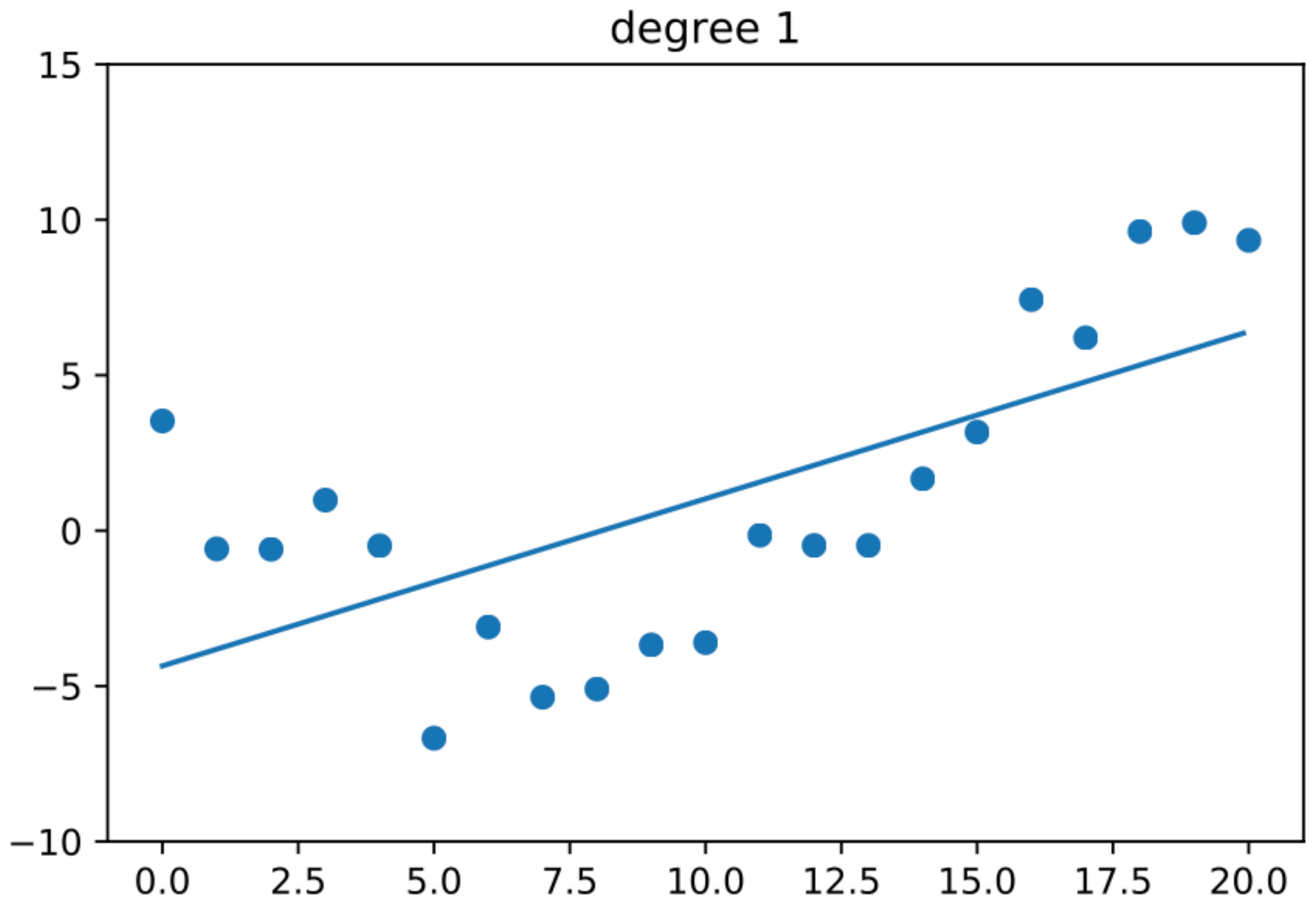
$$p(\boldsymbol{y}|\boldsymbol{x}, \mathbf{W}) = \prod_{j=1}^{J} \mathcal{N}(y_j|\boldsymbol{w}_j^\top \boldsymbol{x}, \sigma_j^2)$$

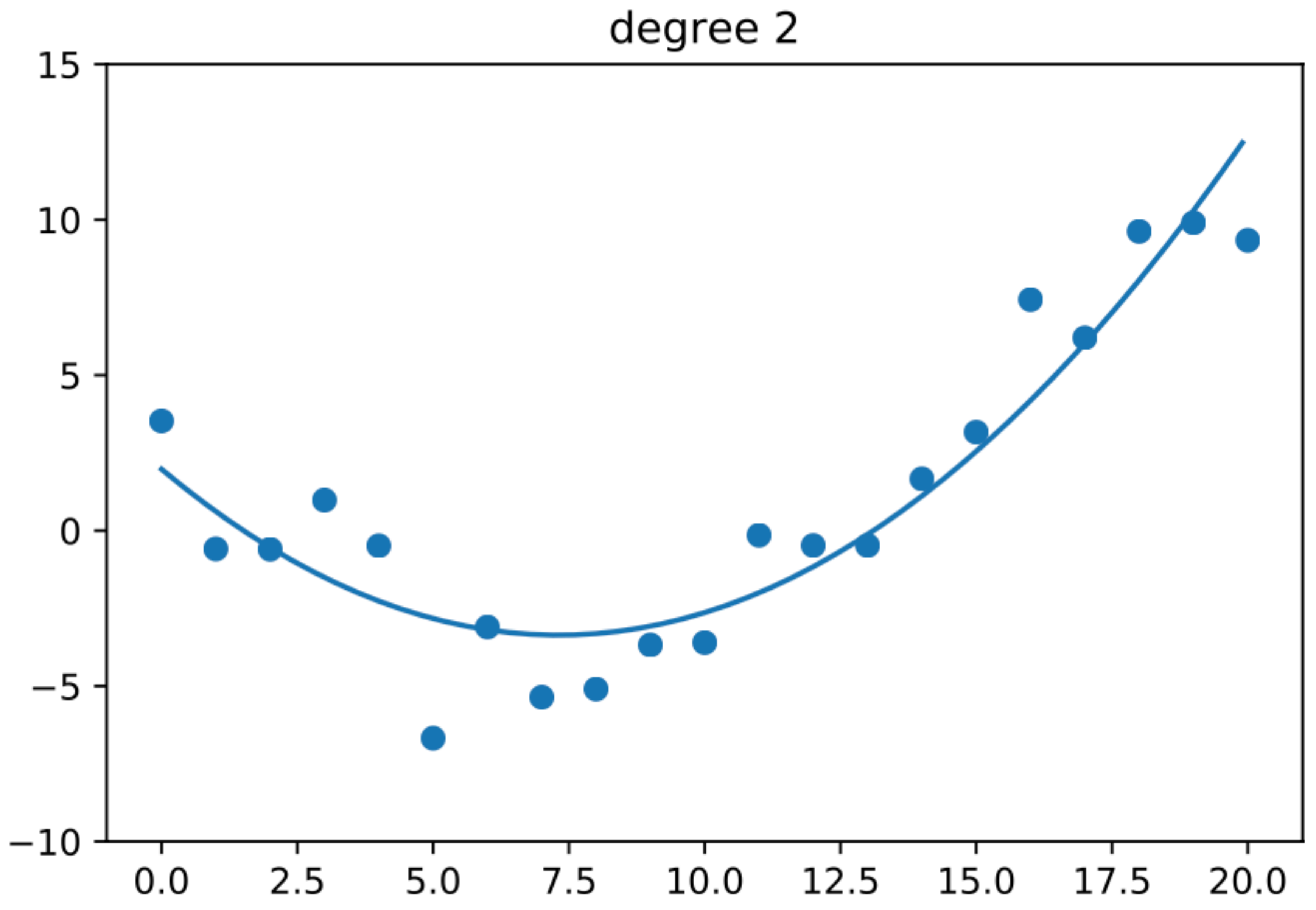
Linear Regression

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\boldsymbol{w}^\mathsf{T}\boldsymbol{\phi}(\boldsymbol{x}), \sigma^2)$$

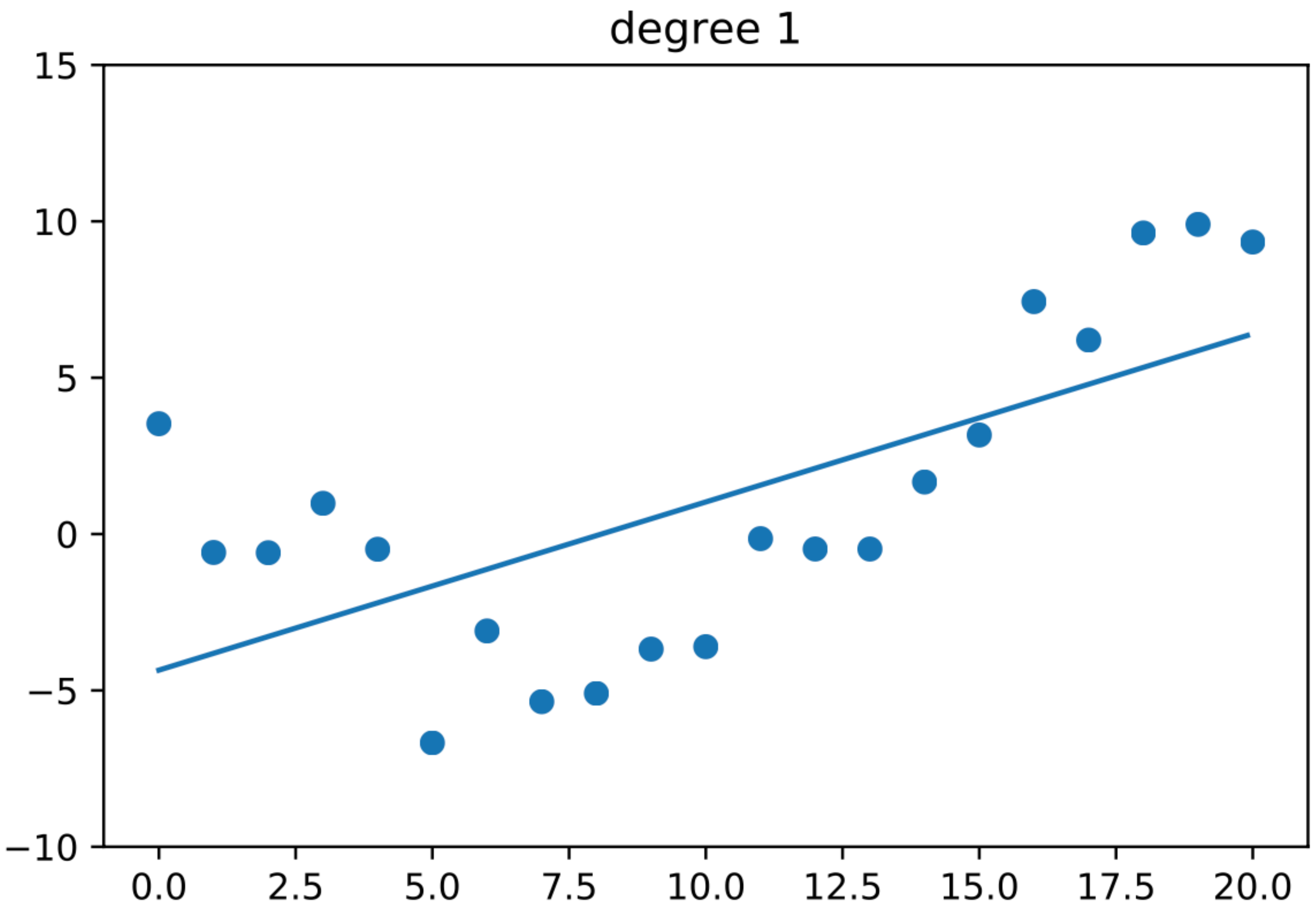
$\boldsymbol{\phi}(x) = [1, x, x^2, \ldots, x^d]$ Polynomial Regression

$\phi(\cdot)$ Feature Extractor





Linear Regression



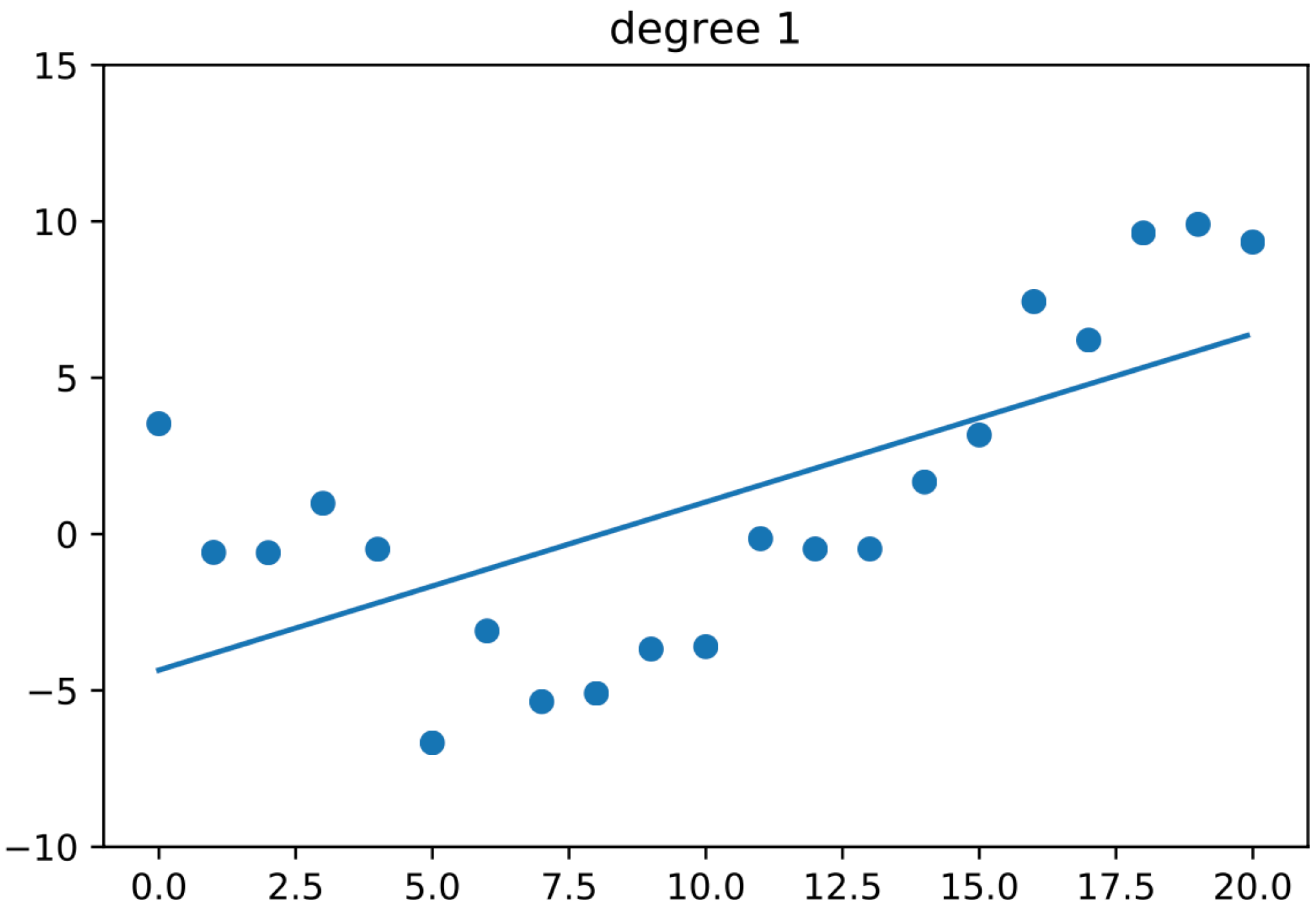
$$\mathrm{RSS}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n)^2 = \frac{1}{2}||\mathbf{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 = \frac{1}{2}(\mathbf{X}\boldsymbol{w} - \boldsymbol{y})^\mathsf{T}(\mathbf{X}\boldsymbol{w} - \boldsymbol{y})$$

residual sum of squares

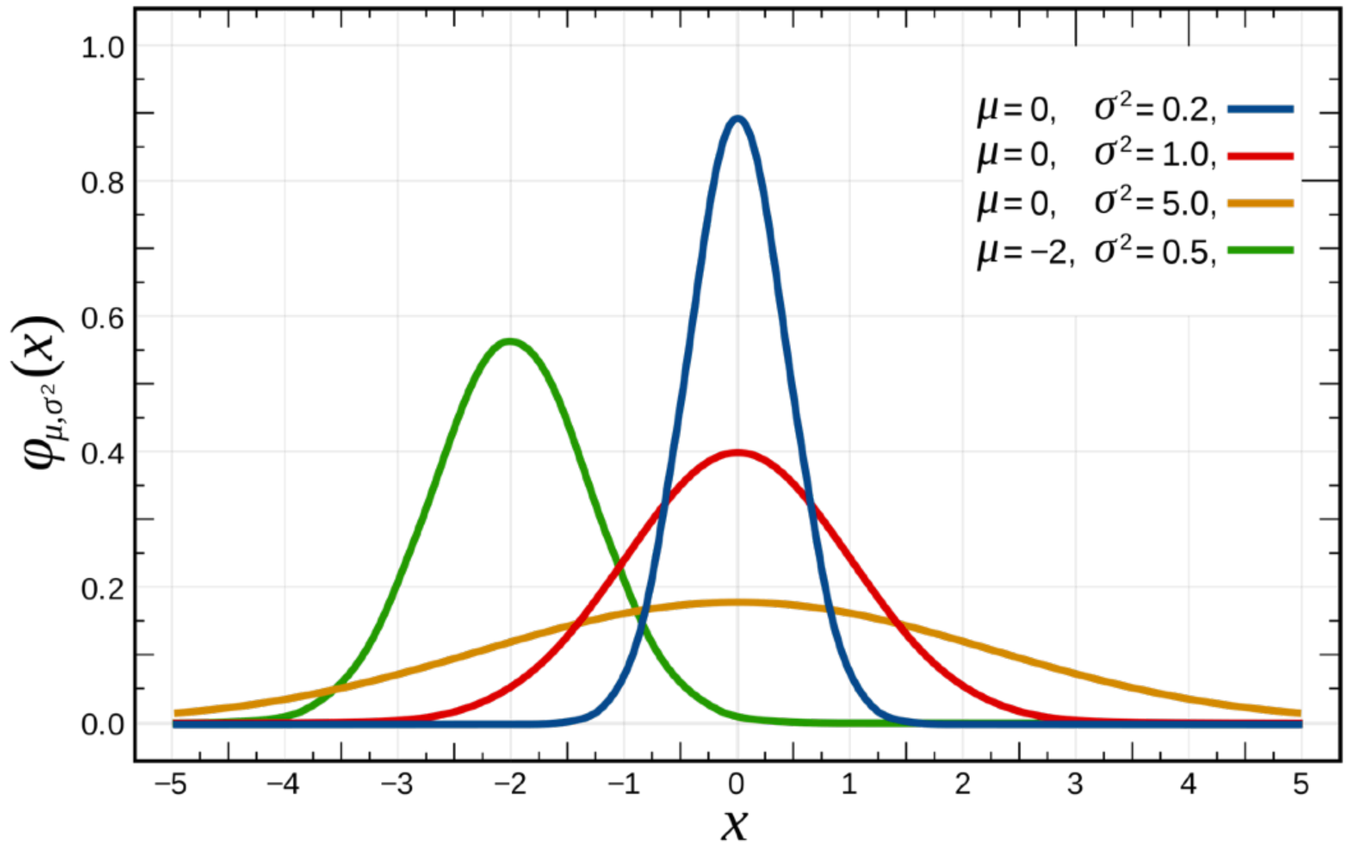
$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{N}\mathrm{RSS}(\boldsymbol{w})$$

mean squared error

Linear Regression



$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + \boldsymbol{w}^\mathsf{T}\boldsymbol{x}, \sigma^2)$$



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$\begin{aligned}\text{NLL}(\boldsymbol{w}, \sigma^2) &= -\sum_{n=1}^{N} \log\left[\left(\frac{1}{2\pi\sigma^2}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2}(y_n - \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_n)^2\right)\right] \\ &= \frac{1}{2\sigma^2}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 + \frac{N}{2}\log(2\pi\sigma^2)\end{aligned}$$

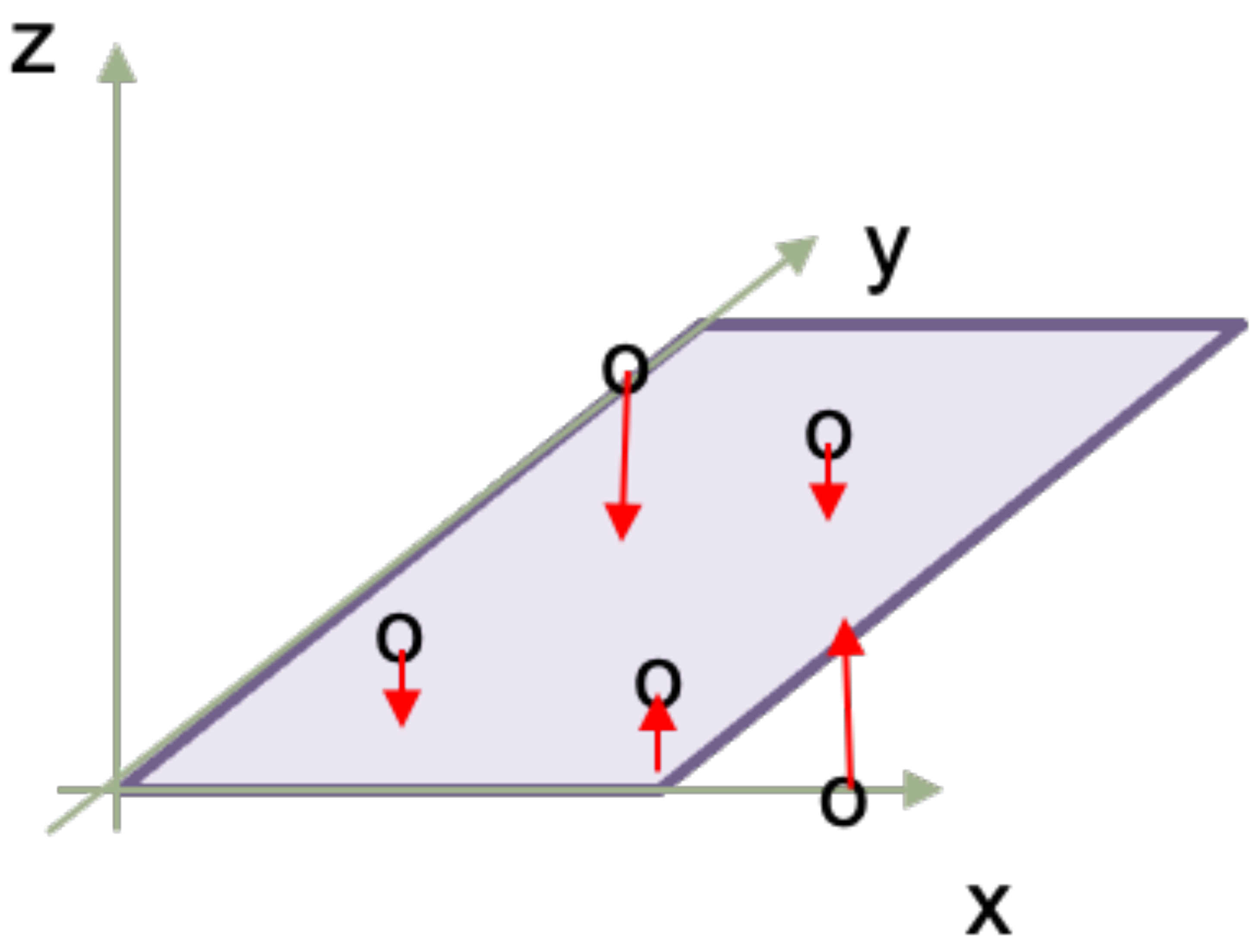
$$\hat{\boldsymbol{w}}_{\text{MLE}} = \arg\min_{\boldsymbol{w}} \text{RSS}(\boldsymbol{w})$$

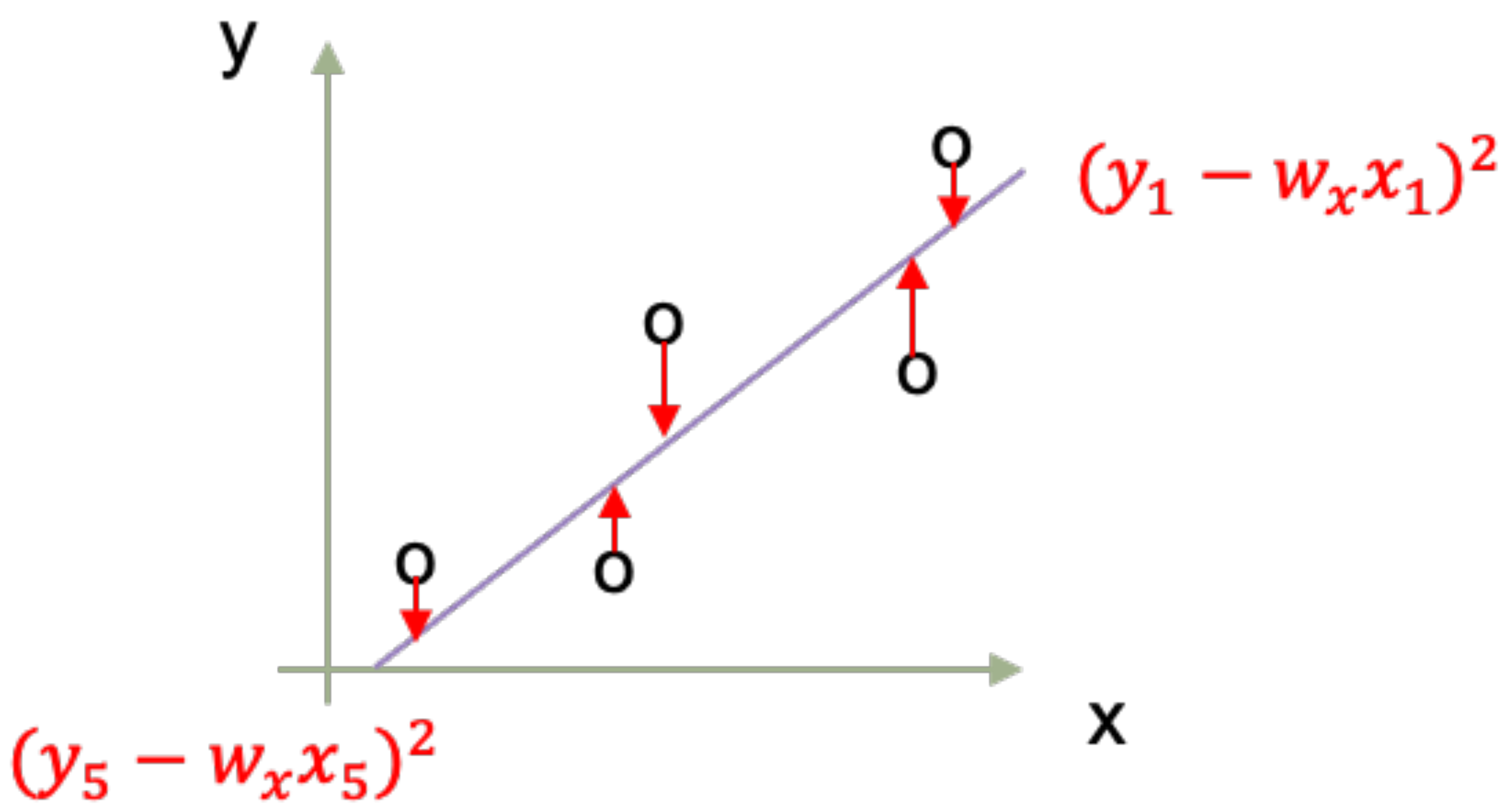
Optimization

$$\mathrm{RSS}(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 = \frac{1}{2}||\mathbf{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 = \frac{1}{2}(\mathbf{X}\boldsymbol{w} - \boldsymbol{y})^\top(\mathbf{X}\boldsymbol{w} - \boldsymbol{y})$$

$$\nabla_{\boldsymbol{w}}\mathrm{RSS}(\boldsymbol{w}) = \mathbf{X}^\top\mathbf{X}\boldsymbol{w} - \mathbf{X}^\top\boldsymbol{y}$$

Gradient Descent





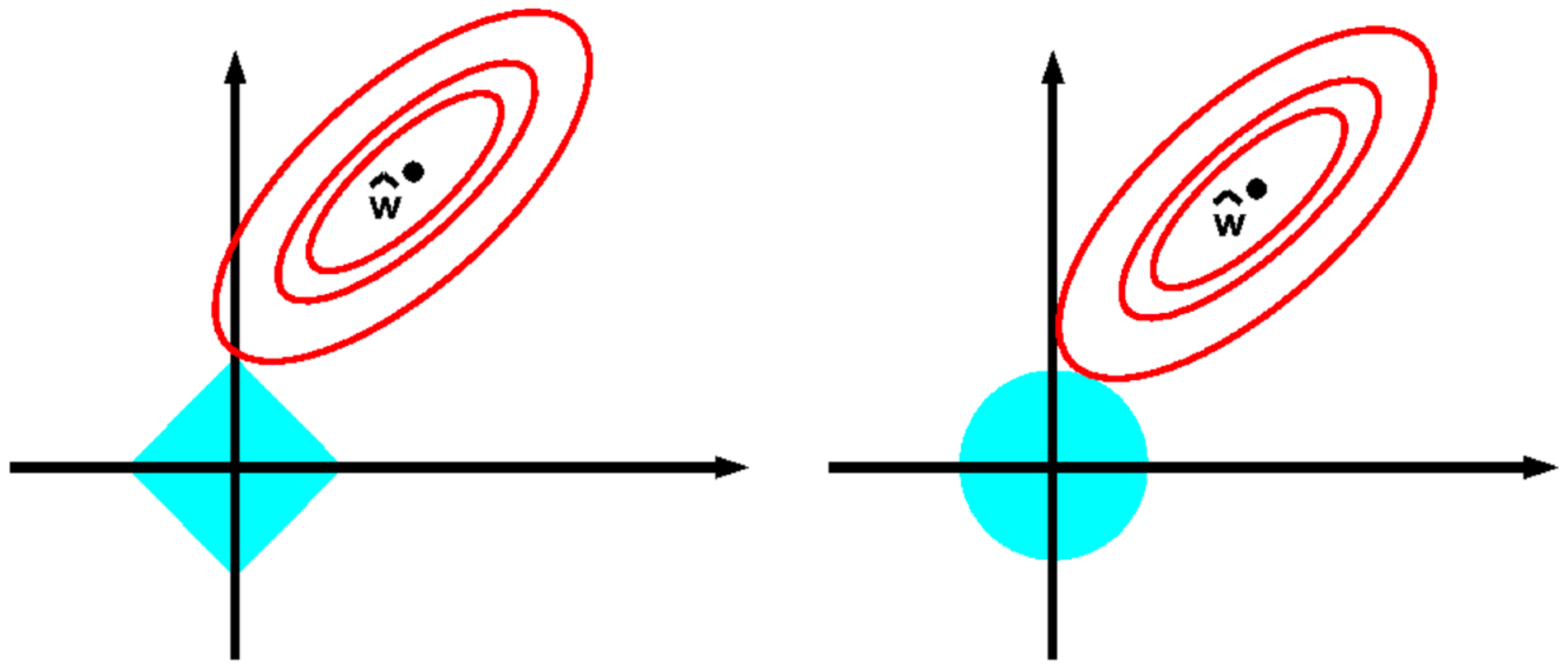
Ridge / Lasso Regression

Ridge Regression (MAP derivation)

$$\hat{\boldsymbol{w}}_{\text{map}} = \operatorname{argmin} \frac{1}{2\sigma^2}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w})^\mathsf{T}(\boldsymbol{y} - \mathbf{X}\boldsymbol{w}) + \frac{1}{2\tau^2}\boldsymbol{w}^\mathsf{T}\boldsymbol{w}$$
$$= \operatorname{argmin} \mathrm{RSS}(\boldsymbol{w}) + \lambda||\boldsymbol{w}||_2^2$$

Lasso Regression (Loss function derivation)

$$\mathrm{PNLL}(\boldsymbol{w}) = -\log p(\mathcal{D}|\boldsymbol{w}) - \log p(\boldsymbol{w}|\lambda) = ||\mathbf{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_1$$



Lasso Regression

$$\text{PNLL}(\boldsymbol{w}) = -\log p(\mathcal{D}|\boldsymbol{w}) - \log p(\boldsymbol{w}|\lambda) = ||\mathbf{X}\boldsymbol{w} - \boldsymbol{y}||_2^2 + \lambda||\boldsymbol{w}||_1$$

Ridge allows parameters to be small but cannot reach zero.

Lasso allow parameters to be exactly zero.

This is useful because it can be used to perform feature selection, where the weight of certain features can be zero.

Q-norm

General Equation

$$||\boldsymbol{w}||_q = (\sum_{d=1}^{D} |w_d|^q)^{1/q}$$

L0 Regularization

$$||\boldsymbol{w}||_0 = \sum_{d=1}^{D} \mathbb{I}(|w_d| > 0)$$

L1 Regularization

$$||\boldsymbol{w}||_1 \triangleq \sum_{d=1}^{D} |w_d|$$

L2 Regularization

$$||\boldsymbol{w}||_2 \triangleq \sqrt{\sum_{d=1}^{D} |w_d|^2} = \sqrt{\boldsymbol{w}^\top \boldsymbol{w}}$$

Elastic Net — Lasso + Ridge

$$\mathcal{L}(\boldsymbol{w}, \lambda_1, \lambda_2) = ||\boldsymbol{y} - \mathbf{X}\boldsymbol{w}||^2 + \lambda_2 ||\boldsymbol{w}||_2^2 + \lambda_1 ||\boldsymbol{w}||_1$$

"elastic net" regularization
(Zou and Hastie, 2005; Friedman et al., 2008)

Example — Cancer Data

id	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa	train
1	-0.579818	2.76946	50	-1.38629	0	-1.38629	6	0	-0.430783	T
2	-0.994252	3.31963	58	-1.38629	0	-1.38629	6	0	-0.162519	T
3	-0.510826	2.69124	74	-1.38629	0	-1.38629	7	20	-0.162519	T
4	-1.20397	3.28279	58	-1.38629	0	-1.38629	6	0	-0.162519	T
5	0.751416	3.43237	62	-1.38629	0	-1.38629	6	0	0.371564	T
6	-1.04982	3.22883	50	-1.38629	0	-1.38629	6	0	0.765468	T
8	0.693147	3.53951	58	1.53687	0	-1.38629	6	0	0.854415	T
11	0.254642	3.60414	65	-1.38629	0	-1.38629	6	0	1.26695	T
12	-1.34707	3.59868	63	1.26695	0	-1.38629	6	0	1.26695	T

Term	Least Squares	Ridge	Lasso	Elastic Net
Intercept	2.452	2.452	2.3520	2.423
lcavol	0.705	0.552	0.5710	0.611
lweight	0.292	0.278	0.2290	0.212
age	-0.142	-0.091	0.0000	0.000
lbph	0.211	0.193	0.1050	0.054
svi	0.307	0.269	0.1710	0.121
lcp	-0.276	-0.102	0.0000	0.000
gleason	-0.012	0.025	0.0000	0.000
pgg45	0.262	0.177	0.0653	0.021
Test MSE	**0.547**	**0.511**	**0.4820**	**0.450**

Least Square – Worst
Ridge – weight is smaller but won't reach zero, better than LS
Lasso – some features' weight are zero; features eliminated
Elastic Net - Best

Learning Linear Models

General training setup:

We observe a collection of examples.

Perform statistical analysis to discover from the data.

Ranges from “count and normalize” to complex optimization routines.

Optimization view:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \underbrace{\frac{1}{N}\sum_{n=1}^{N} L(\mathbf{w}; x_n, y_n)}_{\text{empirical loss}} + \underbrace{\Omega(\mathbf{w})}_{\text{regularizer}}$$