Machine Learning II
Week #1

Jan Nagler

Deep Dynamics Group
Centre for Human and Machine Intelligence (HMI)
Frankfurt School of Finance & Management

Frankfurt, October 2019

Connections

Nonlinear Correlation

Causation

Clustering

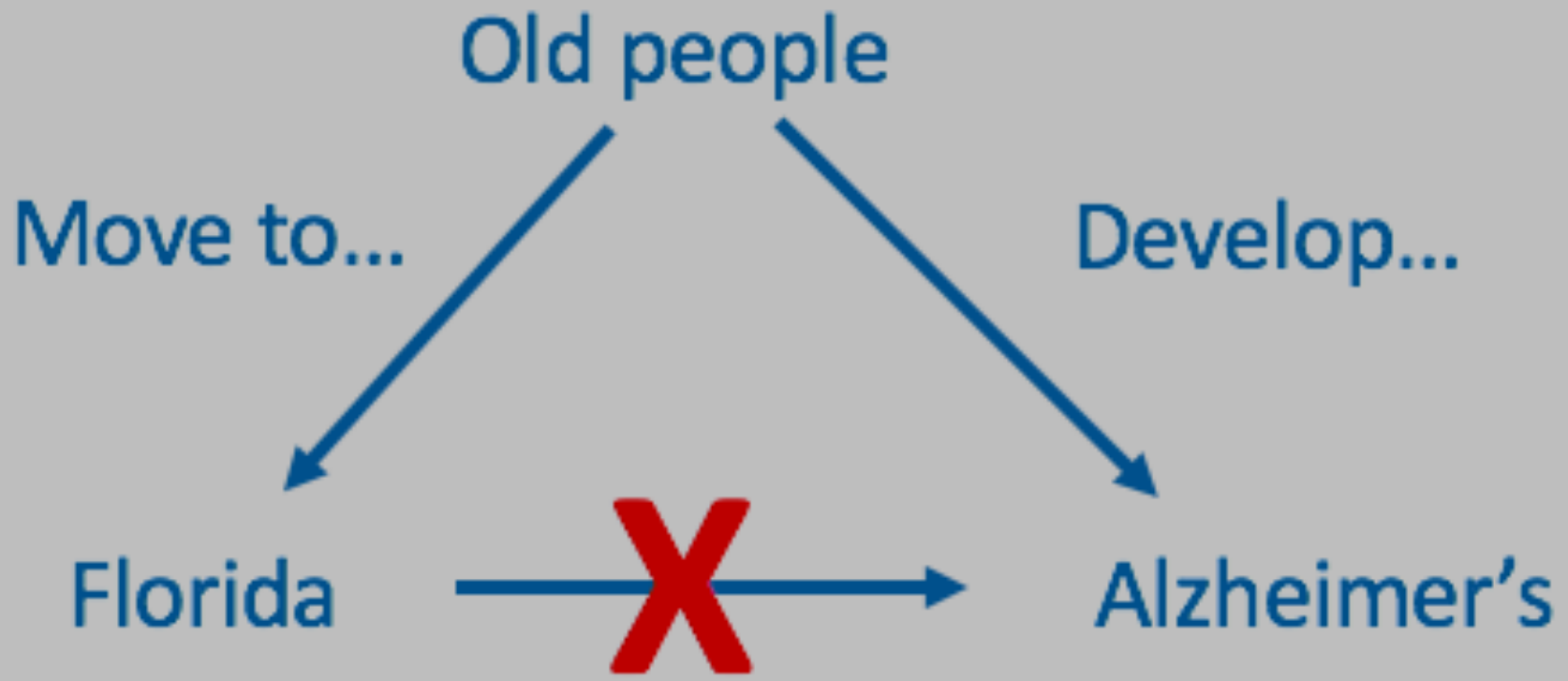
Linear Prediction

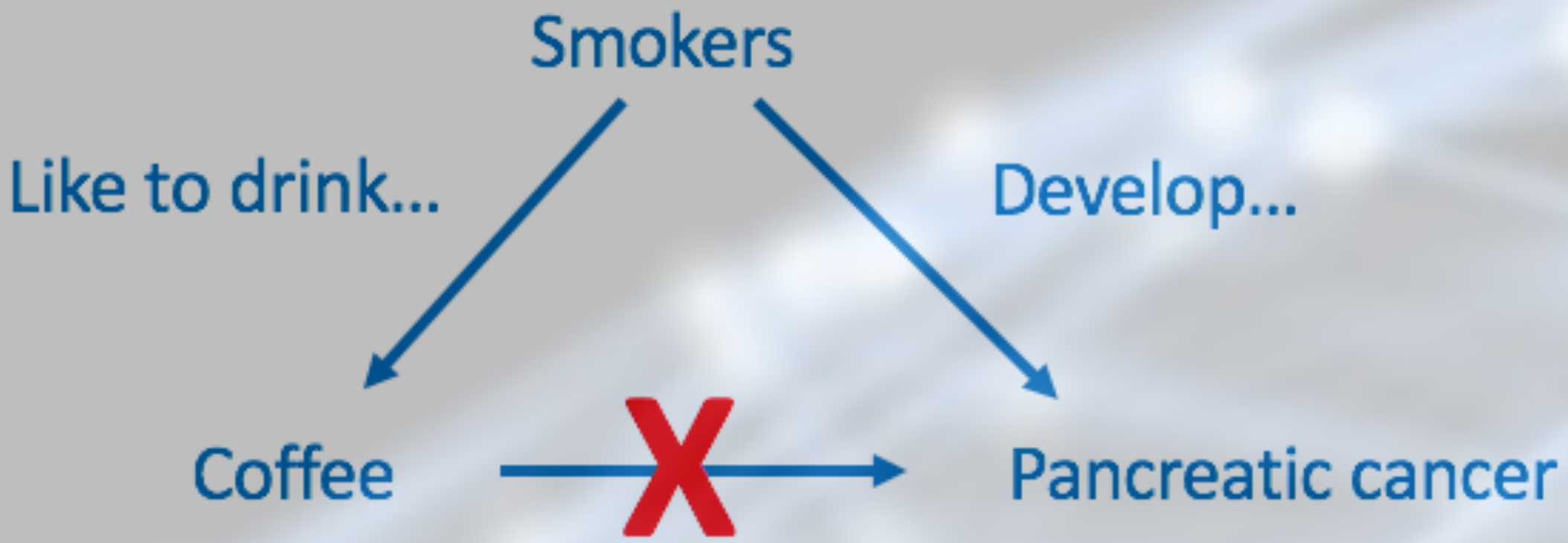
Sample size realities

Methods
(Covariance, K-Means, PCA/OVA)

Data
(Surrogates creation, Wine-178)

Causality and Correlations





Anticorrelations (=neg. corr.) around year 2000

Pearson correlations = Covariance matrix (normalized) of Cryptos

Correlation between Top 15 Market Cap Currencies

@CryptoKit

Large-cap asset 3-month daily return correlation matrix (USD) 01/01/2017 - 10/04/2018

	BTC	ETH	XRP	BCH	LTC	EOS	ADA	XLM	NEO	MIOTA	XMR	DASH	TRX	XEM	ETC
BTC															
ETH	**0.45**														
XRP	0.20	0.19													
BCH	0.21	0.25	0.14												
LTC	**0.46**	**0.42**	0.27	0.23											
EOS	**0.31**	**0.31**	0.17	0.23	0.27										
ADA	0.22	0.18	0.27	0.10	0.17	0.17									
XLM	0.28	0.27	**0.50**	0.12	**0.31**	0.20	**0.31**								
NEO	0.30	**0.33**	0.13	0.15	**0.31**	0.19	0.14	0.21							
MIOTA	**0.44**	**0.39**	0.18	0.23	**0.35**	**0.32**	**0.32**	**0.34**	0.26						
XMR	**0.51**	**0.52**	0.23	0.28	**0.43**	0.28	0.26	**0.40**	0.24	**0.45**					
DASH	**0.42**	**0.45**	0.10	**0.31**	**0.37**	0.23	0.16	0.20	0.29	**0.34**	**0.56**				
TRX	0.28	0.21	0.17	0.11	0.18	0.24	0.28	0.12	0.09	0.15	0.18	0.19			
XEM	0.28	**0.35**	0.22	0.21	**0.34**	0.23	0.24	**0.32**	0.22	**0.37**	**0.32**	0.29	0.13		
ETC	**0.44**	**0.62**	0.17	**0.32**	**0.51**	0.30	**0.32**	0.29	**0.43**	**0.43**	**0.45**	**0.38**	0.20	**0.35**	

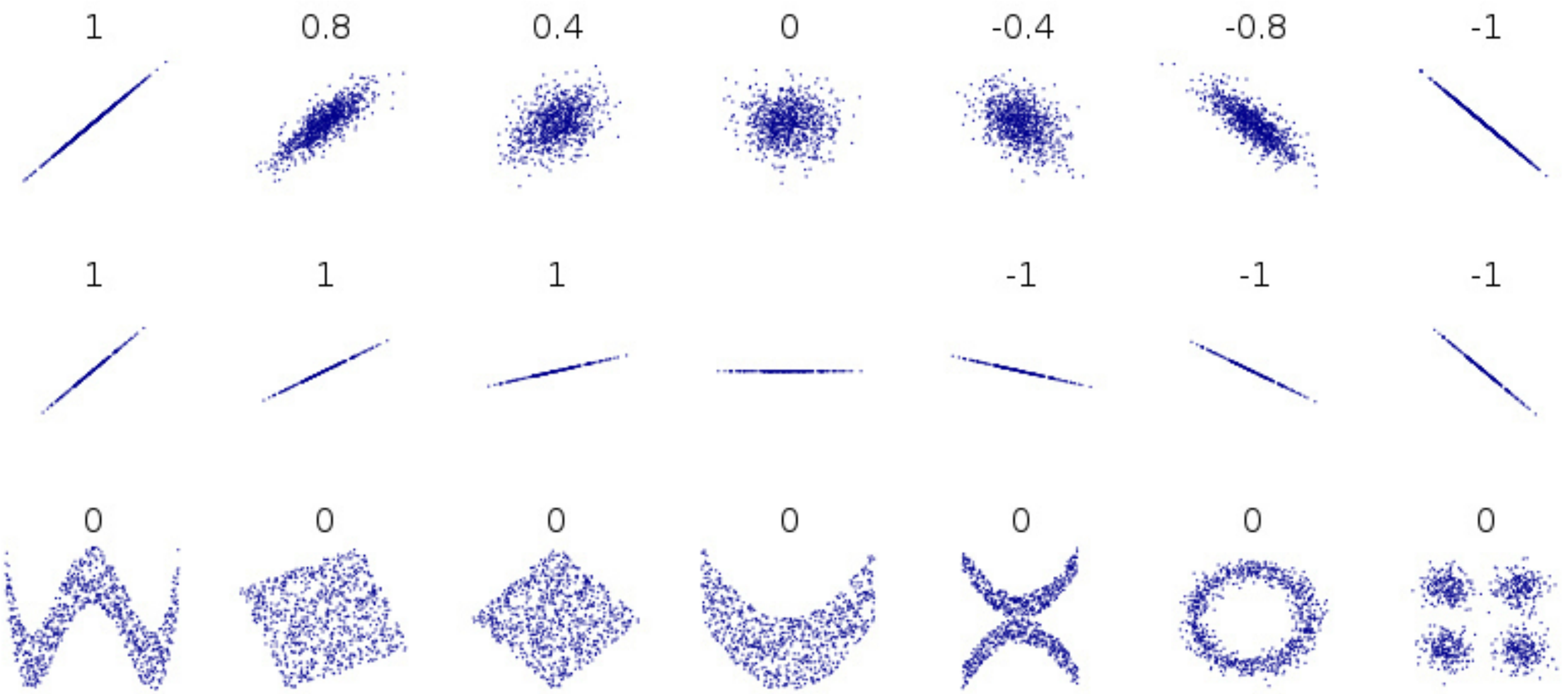
1.00	Strong Positive Relationship
0.50	Moderate Positive Relationship
0.30	Weak Positive Relationship
0.00	No Correlation
-0.30	Weak Negative
-0.50	Moderate Negative Relationship
-1.00	Strong Negative Relationship

Pearson correlations = Covariance matrix (normalized) of Cryptos

3-month daily return correlation matrix (USD) Dec 1 2018 - Mar 1 2019

	BTC	ADA	ETH	XLM	XMR	ZEC	NEO	LSK	XRP	ZRX	EOS	OMG	VET	DASH	QTUM
BTC	1														
ADA	0.908	1													
ETH	0.889	0.900	1												
XLM	0.922	0.893	0.866	1											
XMR	0.929	0.886	0.881	0.878	1										
ZEC	0.909	0.843	0.865	0.861	0.915	1									
NEO	0.873	0.875	0.885	0.864	0.858	0.874	1								
LSK	0.901	0.830	0.837	0.836	0.838	0.847	0.818	1							
XRP	0.875	0.865	0.858	0.887	0.840	0.828	0.847	0.819	1						
ZRX	0.879	0.827	0.834	0.857	0.829	0.828	0.803	0.867	0.812	1					
EOS	0.884	0.889	0.851	0.860	0.824	0.813	0.836	0.770	0.818	0.769	1				
OMG	0.843	0.822	0.846	0.802	0.829	0.839	0.755	0.837	0.740	0.847	0.751	1			
VET	0.855	0.862	0.794	0.839	0.801	0.794	0.793	0.848	0.820	0.820	0.810	0.792	1		
DASH	0.885	0.800	0.799	0.819	0.875	0.892	0.769	0.802	0.752	0.807	0.770	0.840	0.745	1	
QTUM	0.794	0.831	0.826	0.801	0.727	0.746	0.805	0.784	0.793	0.773	0.820	0.754	0.800	0.697	
LTC	0.857	0.851	0.825	0.801	0.805	0.766	0.793	0.753	0.769	0.750	0.831	0.758	0.767	0.706	0.75
ETC	0.783	0.778	0.794	0.760	0.760	0.769	0.773	0.769	0.736	0.806	0.734	0.741	0.724	0.712	0.75
DCR	0.815	0.785	0.795	0.740	0.795	0.785	0.738	0.819	0.760	0.740	0.714	0.722	0.764	0.702	0.69
NEM	0.777	0.766	0.755	0.804	0.755	0.759	0.768	0.777	0.754	0.780	0.652	0.759	0.759	0.699	0.70
IOTA	0.768	0.756	0.768	0.759	0.760	0.752	0.675	0.694	0.738	0.756	0.729	0.777	0.697	0.752	0.78
BCH	0.790	0.752	0.738	0.738	0.778	0.752	0.622	0.745	0.679	0.727	0.675	0.784	0.664	0.815	0.62
ONT	0.697	0.760	0.678	0.743	0.710	0.676	0.747	0.624	0.701	0.618	0.751	0.657	0.709	0.599	0.67
BAT	0.719	0.703	0.728	0.704	0.707	0.702	0.703	0.711	0.674	0.668	0.626	0.688	0.650	0.656	0.62
XTZ	0.696	0.714	0.720	0.711	0.680	0.709	0.724	0.682	0.669	0.666	0.696	0.635	0.605	0.656	0.62
MKR	0.643	0.662	0.698	0.646	0.650	0.642	0.718	0.651	0.618	0.554	0.642	0.566	0.575	0.541	0.64
BNB	0.697	0.652	0.635	0.681	0.667	0.644	0.664	0.643	0.614	0.548	0.643	0.560	0.598	0.569	0.58
TRX	0.615	0.722	0.633	0.628	0.614	0.599	0.646	0.617	0.589	0.583	0.567	0.624	0.693	0.505	0.57

Linear and nonlinear correlations
[number = linear (Pearson) correlation]



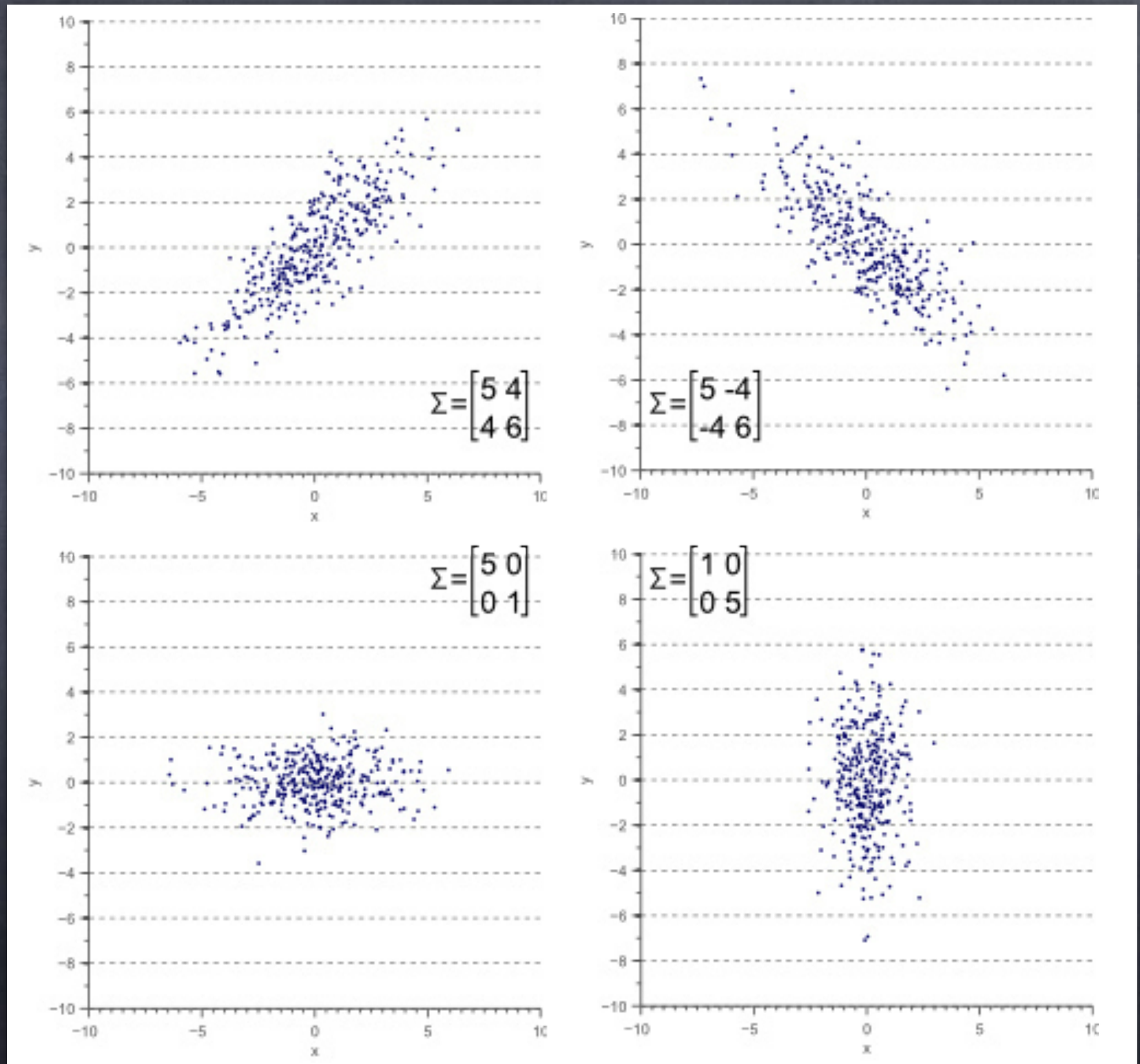
Recall

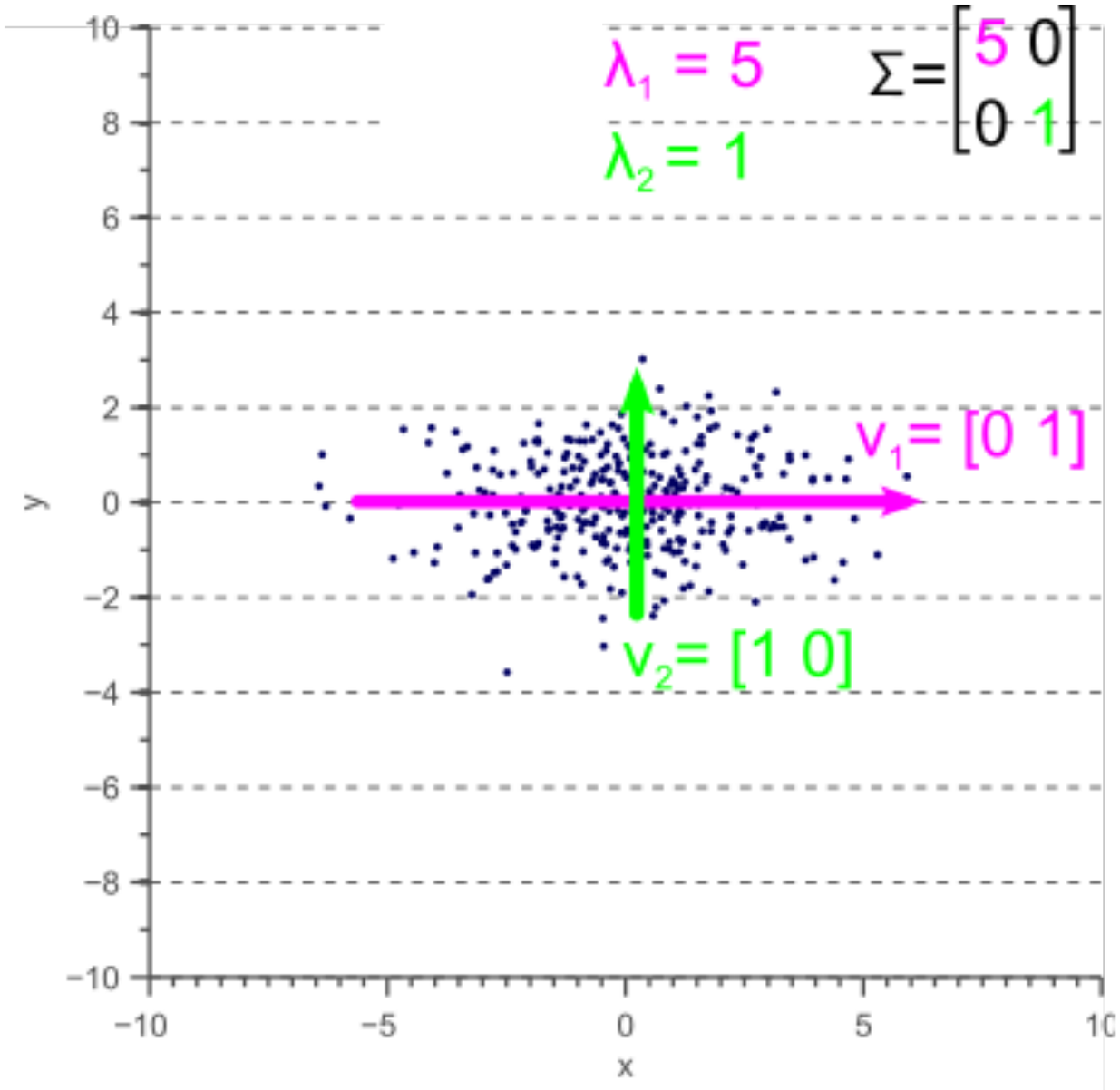
Covariance matrix

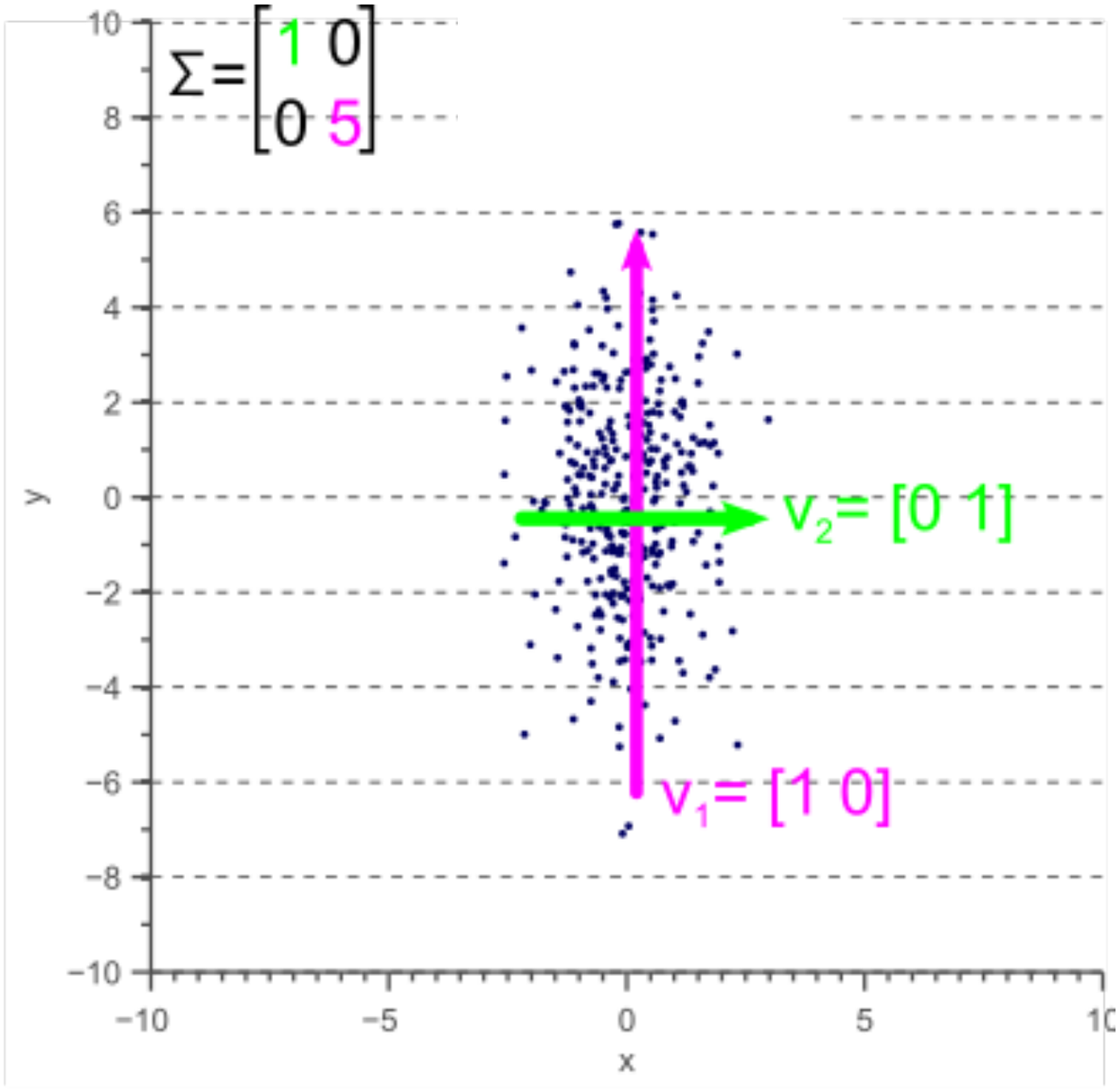
$$\Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix}$$

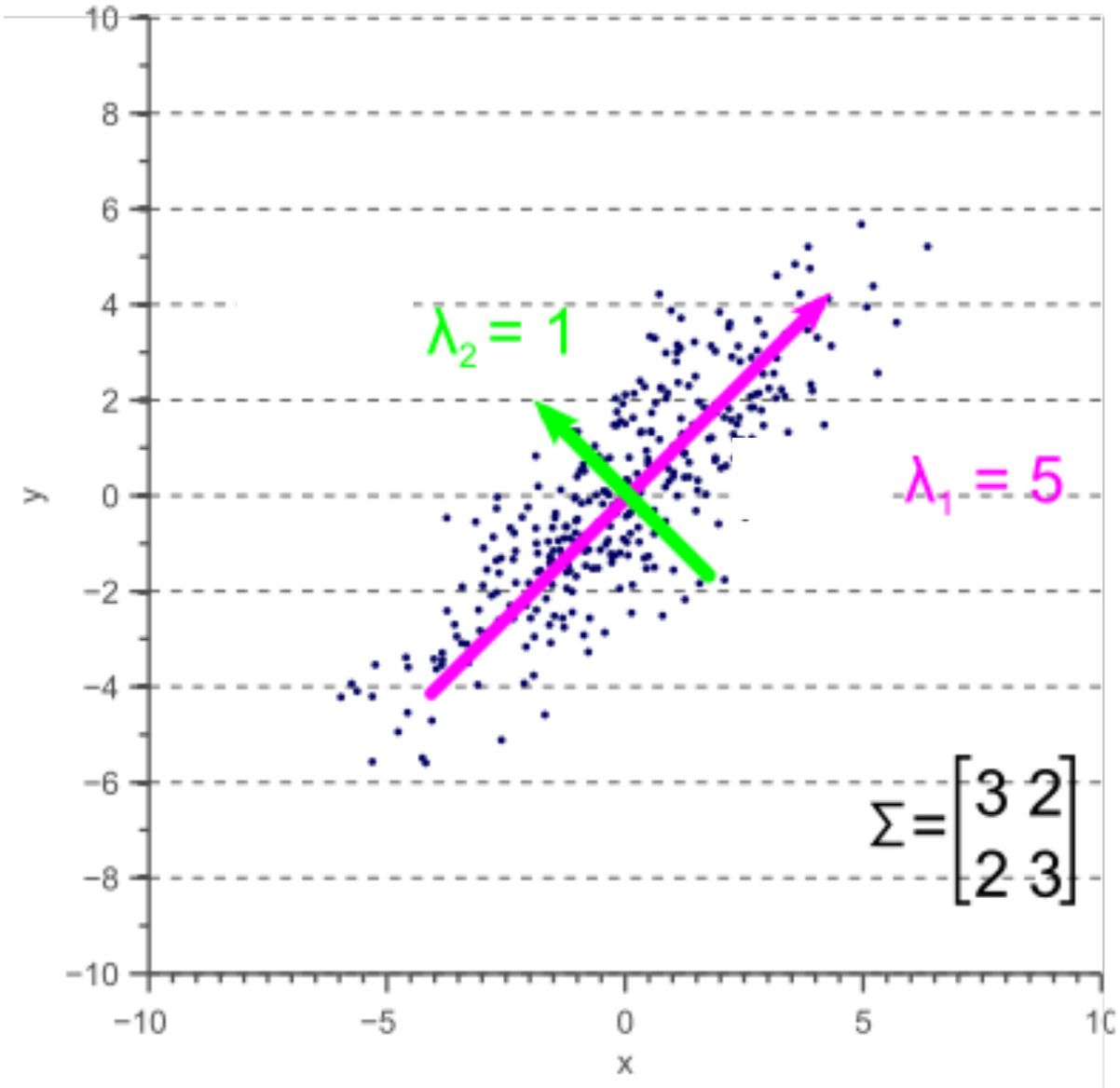
Examples: Linear correlation <-> Nonlinear correlations <-> Causation

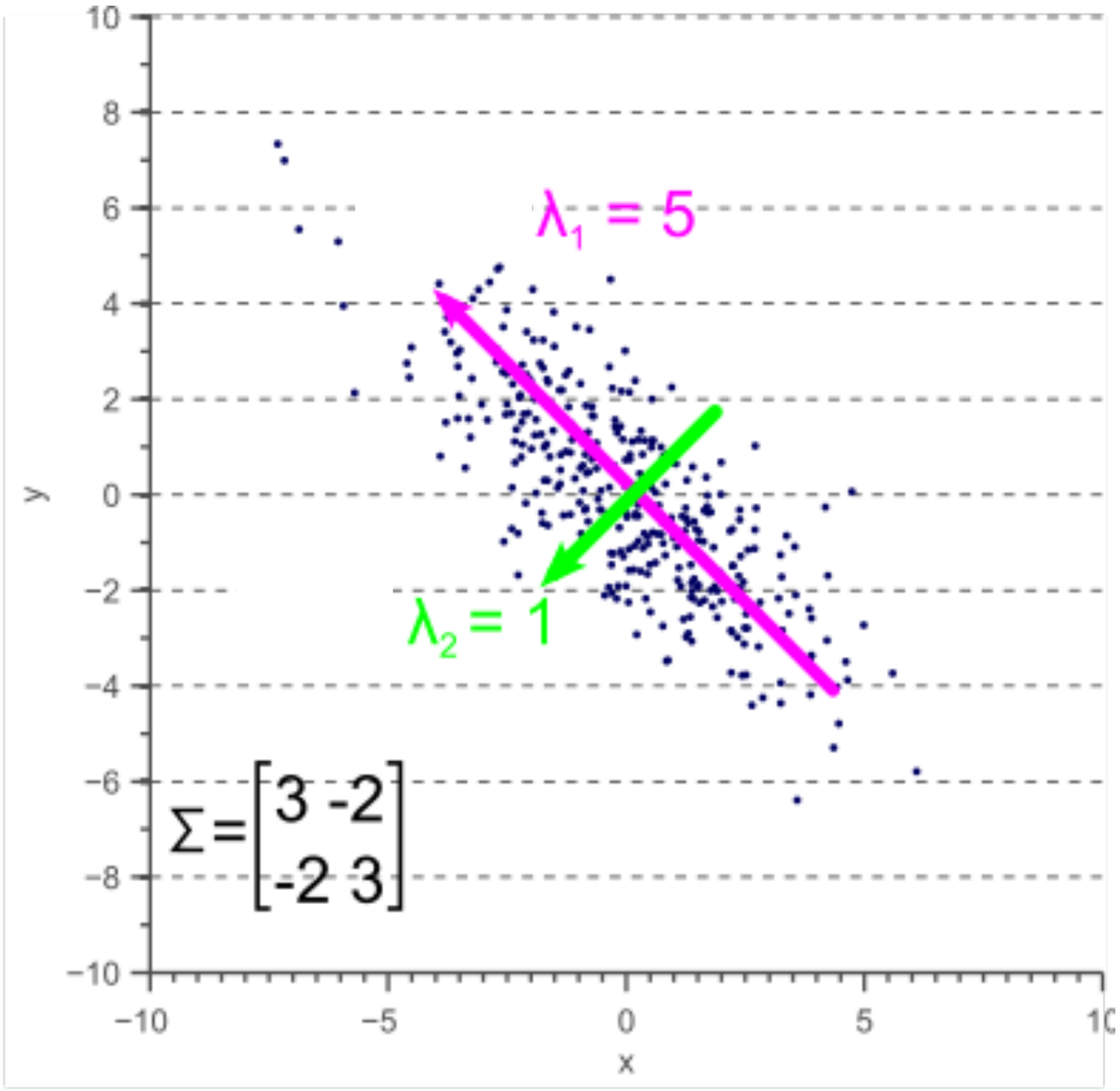
Example: Understanding covariance



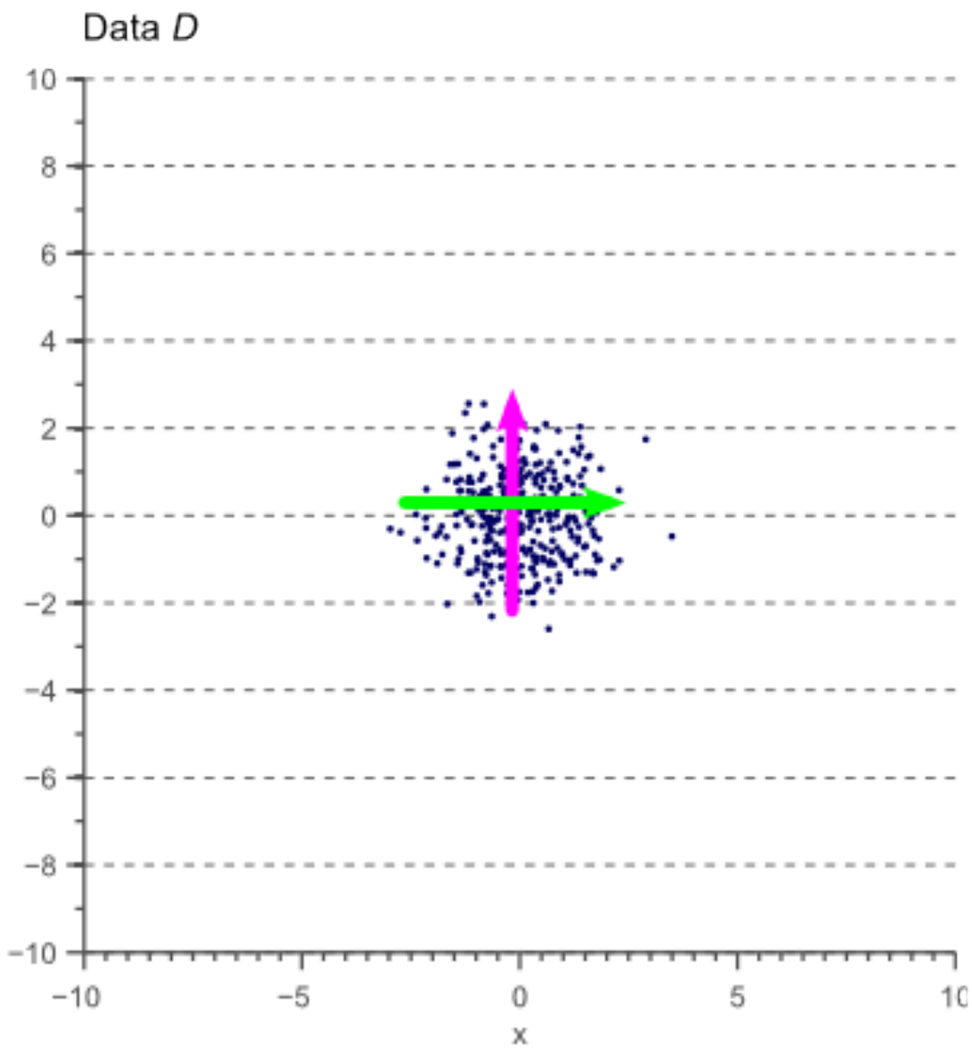

$\lambda_1 = 5$
$\lambda_2 = 1$
$\Sigma = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}$
$v_1 = [0\ 1]$
$v_2 = [1\ 0]$
x
y


$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$
$v_2 = [0\ 1]$
$v_1 = [1\ 0]$
x
y

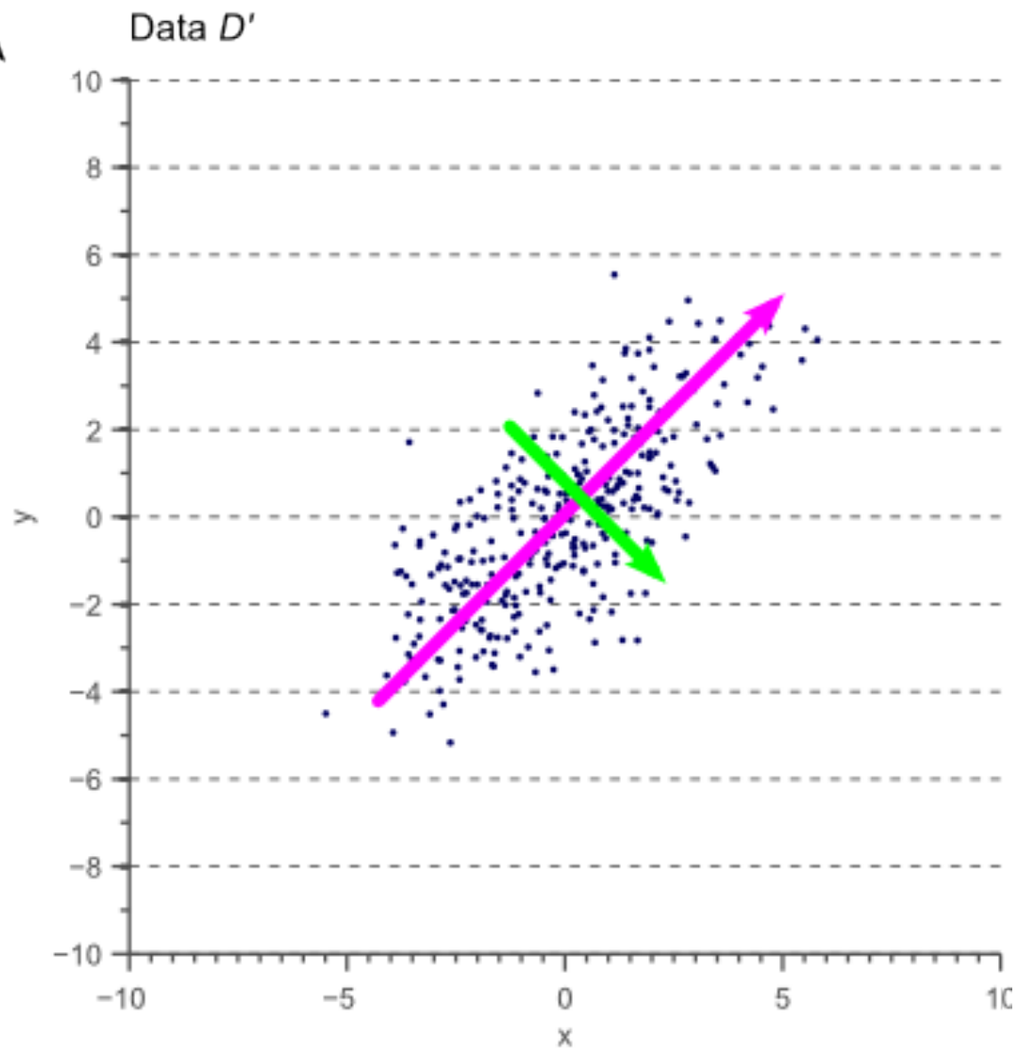

λ2 = 1
λ1 = 5
Σ=[3 2; 2 3]
y
x


λ1 = 5
λ2 = 1
Σ=[3 -2; -2 3]
y
x

Understanding covariance



$$D' = T\,D$$
$$= R\,S\,D$$



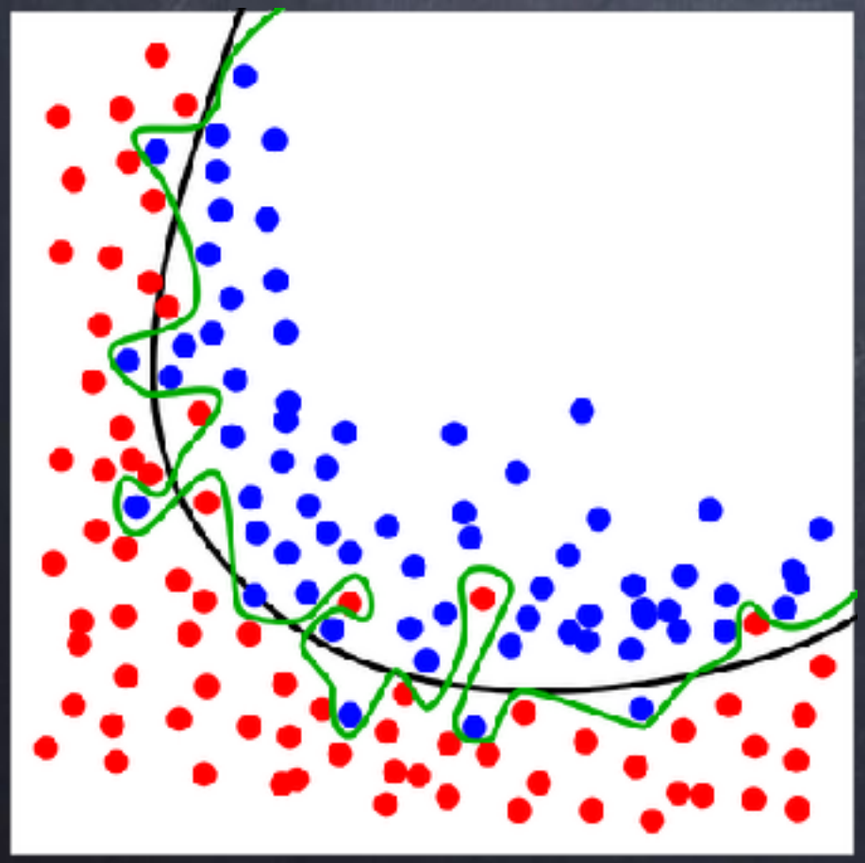
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\Sigma' = \begin{bmatrix} 4.25 & 3.10 \\ 3.10 & 4.29 \end{bmatrix} = R\,S\,S\,R^{-1}$$

PCA

Dimensionality reduction
(Curse of dimensionality)

Avoidance of overfitting
Avoidance of variance
Feature extraction

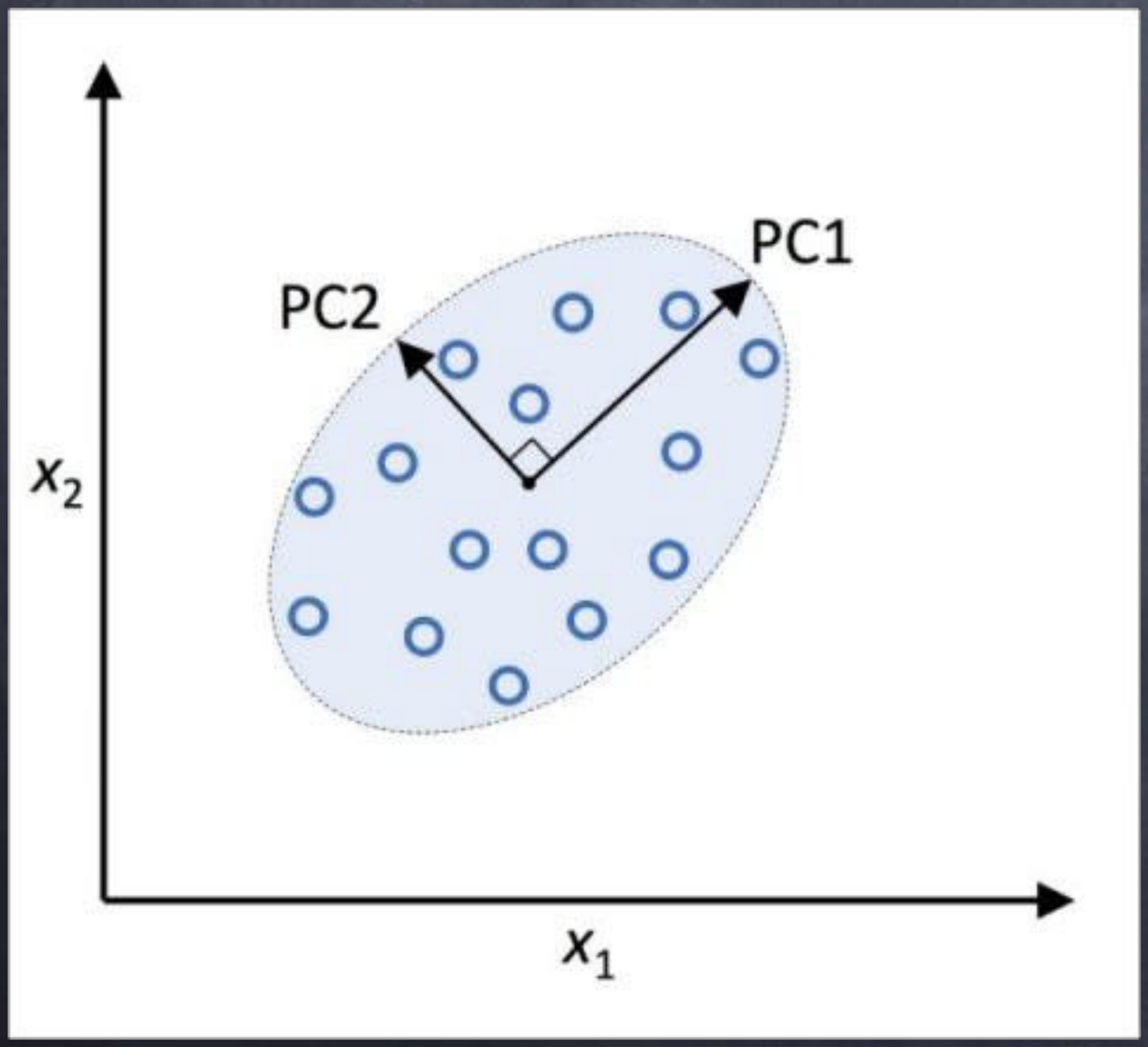


Methods:
PCA
(Principal Component Analysis)

Classifier with linear decision boundary, LDA
(Linear Discriminant Analysis)

P(rincipal) C(omponent) A(nalysis)

Main idea: Obtain principal components
from variance maximization,
given orthogonal feature axes



Main methodology:
Linear algebra

P(rincipal) C(omponent) A(nalysis)

Find linear transformation $W(d \times k)$

$$\mathbf{x} = (x_1, \ldots, x_d) \Rightarrow \mathbf{z} = (z_1, \ldots, z_k)$$

Objective

First principal component selected from largest possible variance,
and all consequent principal components will have the largest variance
given the constraint that these components are uncorrelated (orthogonal)
to the other principal components

Even if the input features are correlated,
resulting principal components will be orthogonal (uncorrelated).

P(rincipal) C(omponent) A(nalysis)

Standardize dataset of dim(d)

Compute covariance matrix

Decompose covariance matrix
using its eigenvectors and eigenvalues

Sort the eigenvalues by decreasing order
to rank the corresponding eigenvectors

Select k eigenvectors which correspond to the k largest eigenvalues,
where k is the dimensionality of the new feature subspace ($k \leq d$)

Construct a projection matrix W from the selected k eigenvectors.

Transform the d-dimensional input dataset X
using the projection matrix W
to obtain the new k-dimensional feature subspace

P(rincipal) C(omponent) A(nalysis)

Test: What is wrong?

$$\sigma_{jk} = \frac{1}{n} \sum_i (x_j^{(i)} - \mu_j) \cdot (y_k^{(i)} - \mu_k)$$

The eigenvalue problem

$$\mathbf{\Sigma v} = \lambda \mathbf{v}$$

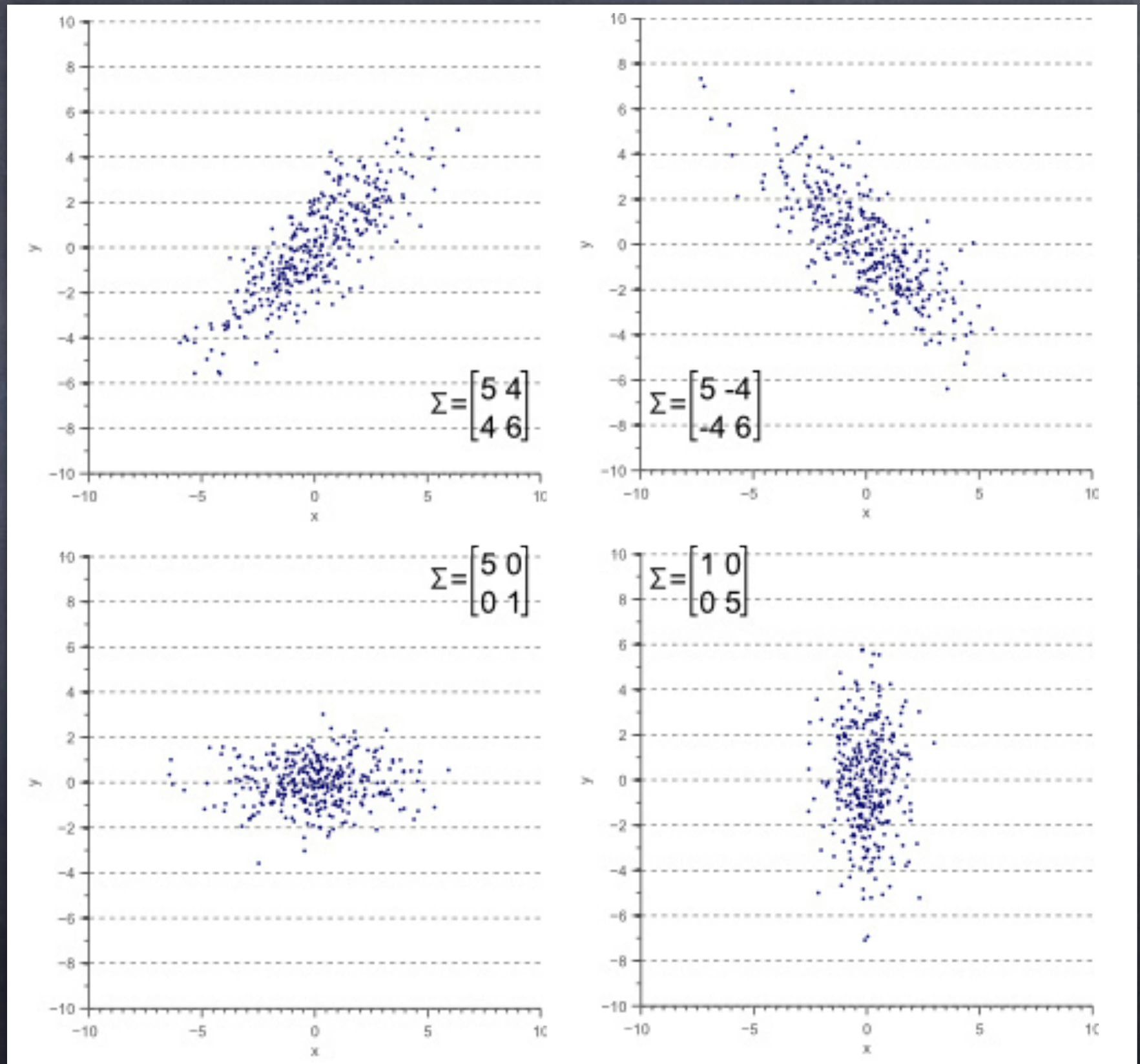
The eigenvectors of the covariance matrix represent
the directions of maximum variance,
hence they are the principal components

2-dim case:

$$\Sigma = \begin{bmatrix} \sigma(x,x) & \sigma(x,y) \\ \sigma(y,x) & \sigma(y,y) \end{bmatrix}$$

P(rincipal) C(omponent) A(nalysis)

Understanding covariance (and just be right but not wrong)



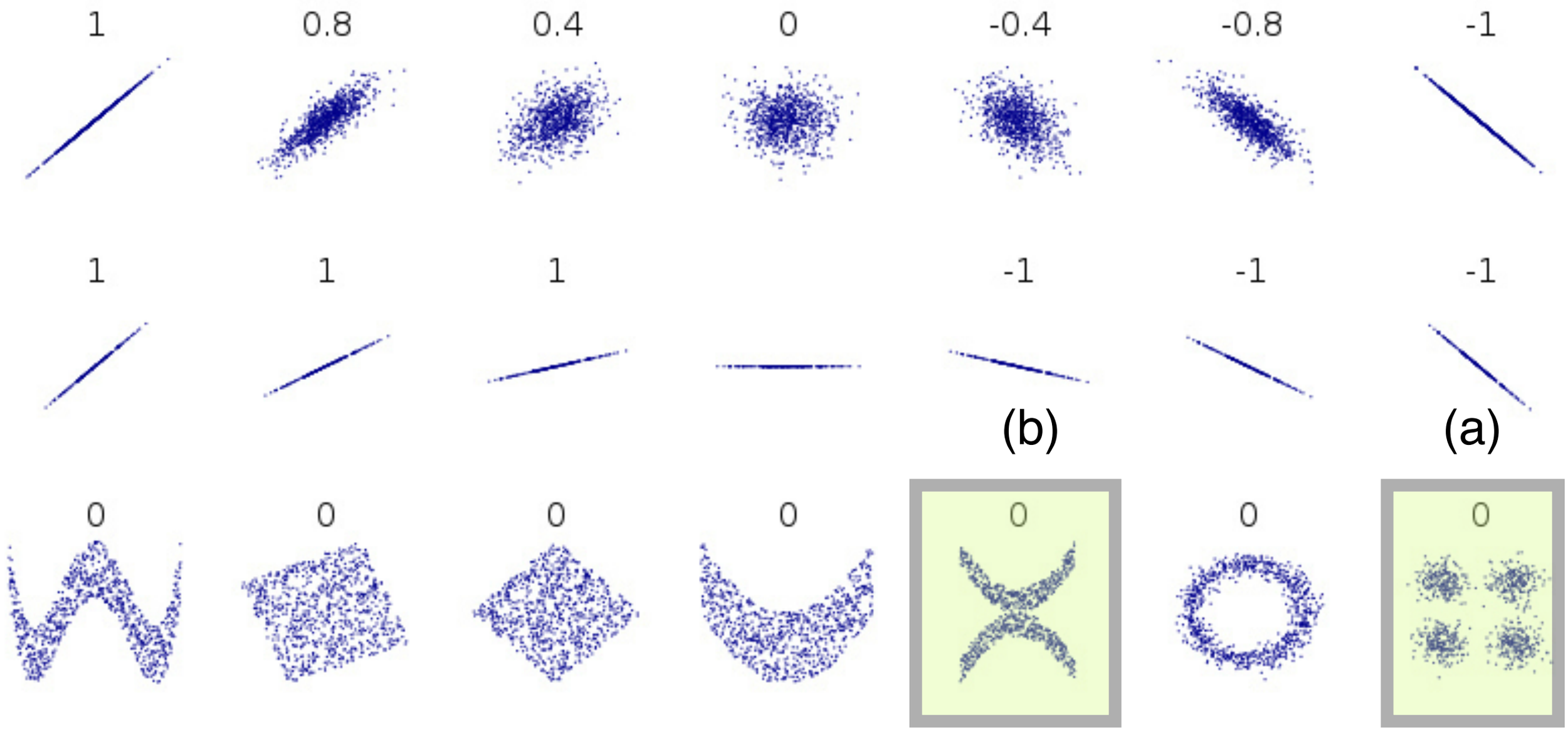
P(rincipal) C(omponent) A(nalysis)

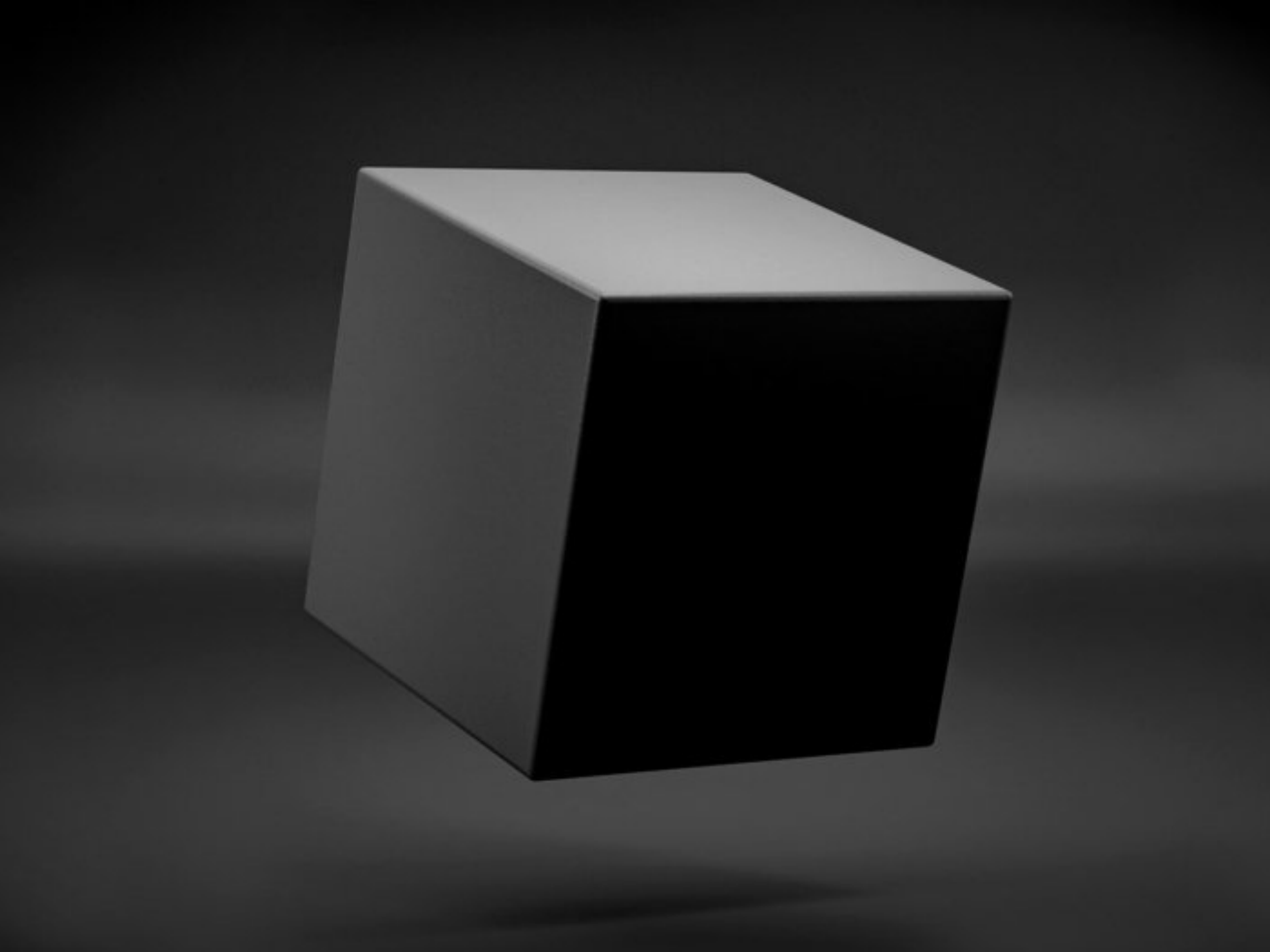
Variance-explained-ratios

$$\frac{|\lambda_j|}{\sum_{i \leq d} |\lambda_i|}$$

Create surrogate data in more than 2 dimension!

0th: labeled data, 1st: PCA, 2nd: classification, for (a) & (b)





K-Means Clustering

centroid (*average*)

Number of clusters, *k*, a priori

K-Means Clustering: Preprocessing

Standardization =
Z-score normalization =
(zero mean, unit variance)

or

Min-Max Scaling

K-Means Clustering

Number of clusters, *k*, a priori
Somehow (e.g., randomly) pick *k* centroids from the sample points as initial cluster centers

Repeat
(1) Assign each point to the nearest centroid

(2) Move the centroids to the center of the samples that were assigned to it

Until
cluster assignments do not change or
within tolerance or
maximum number of iterations

K-Means Clustering

max_iter:
Maximum number of iterations

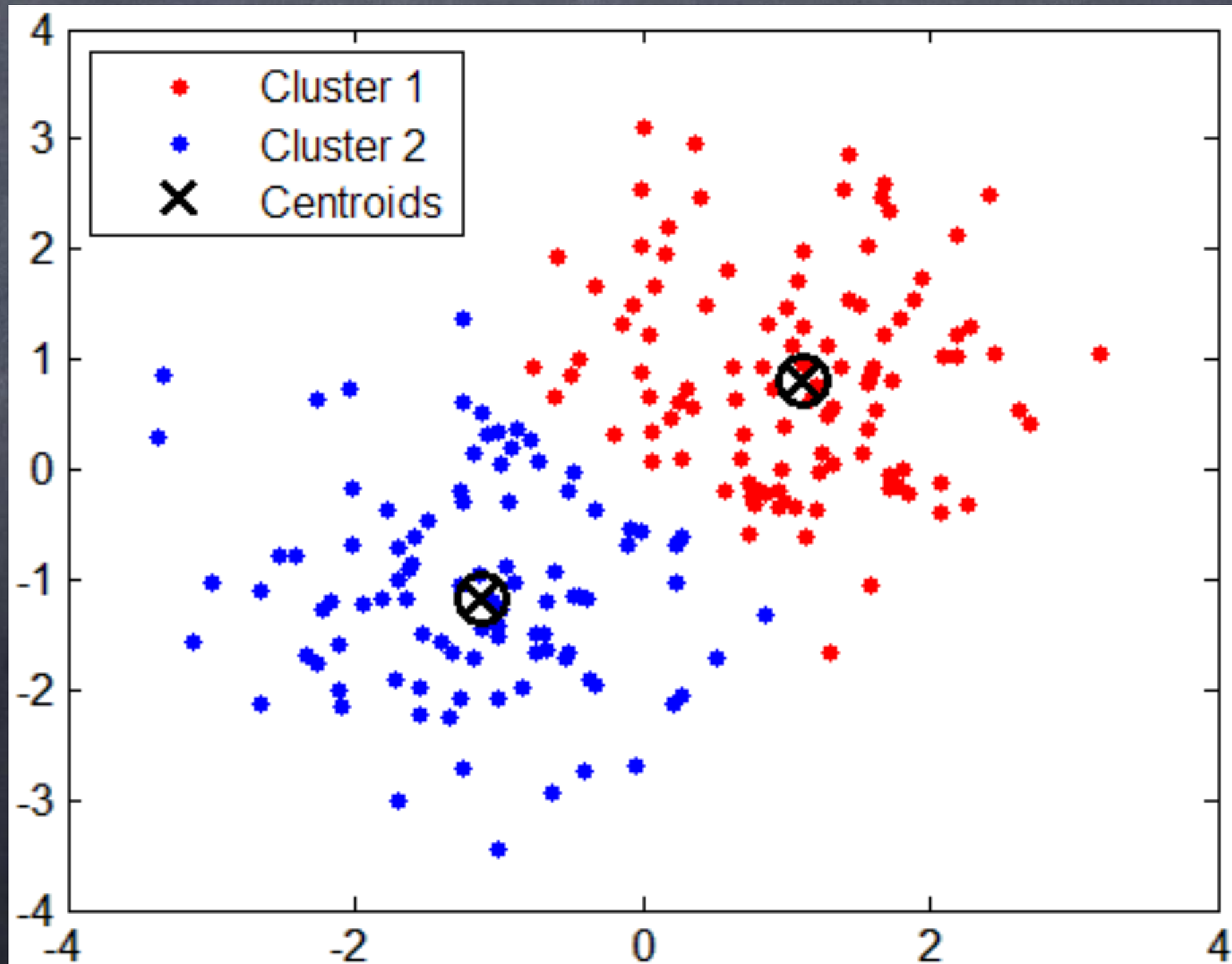
tol(erance):
Relative tolerance with regards to inertia/SSE
to declare convergence,
SSE(iter+1)/SEE(iter) < tol

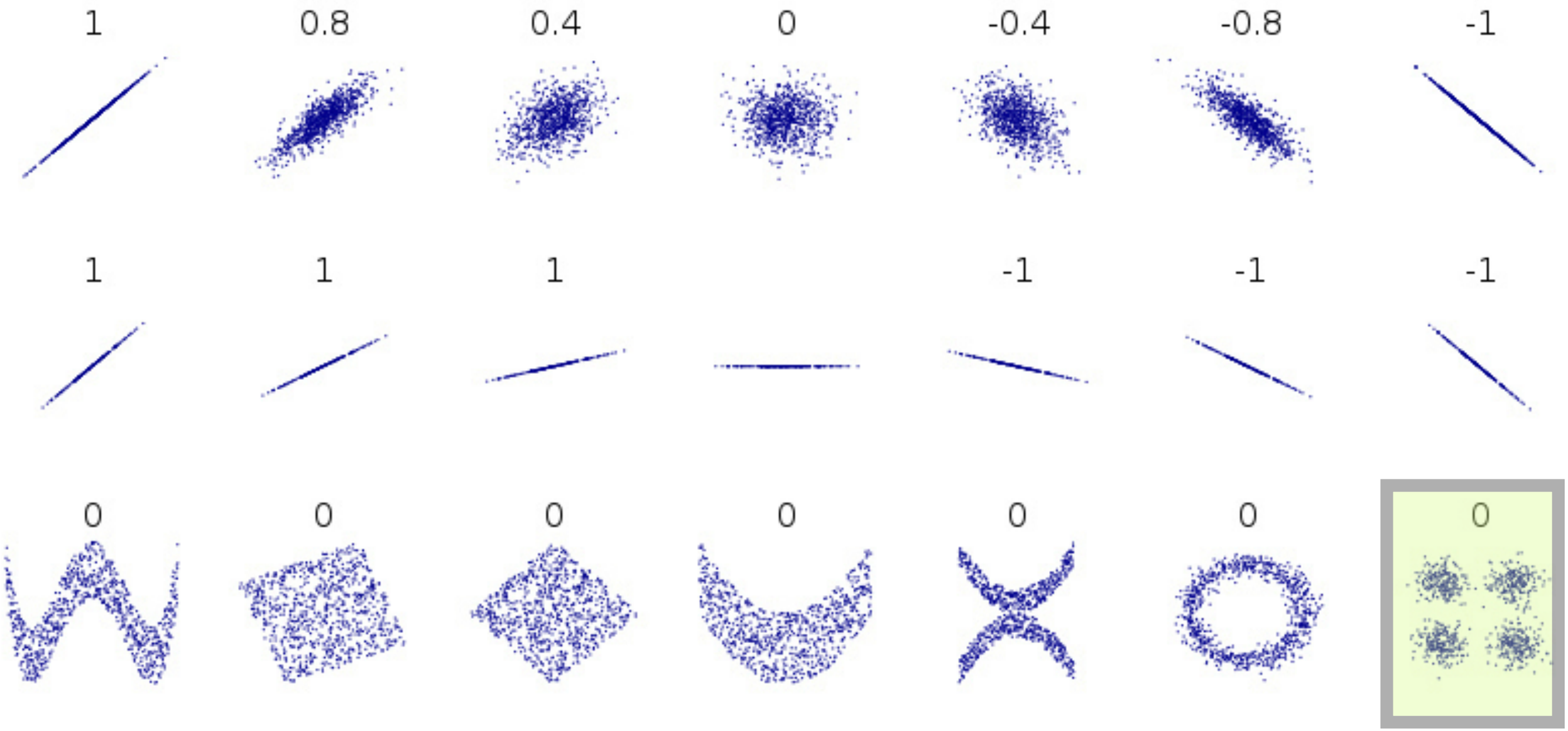
Sum of squared errors (SSE)

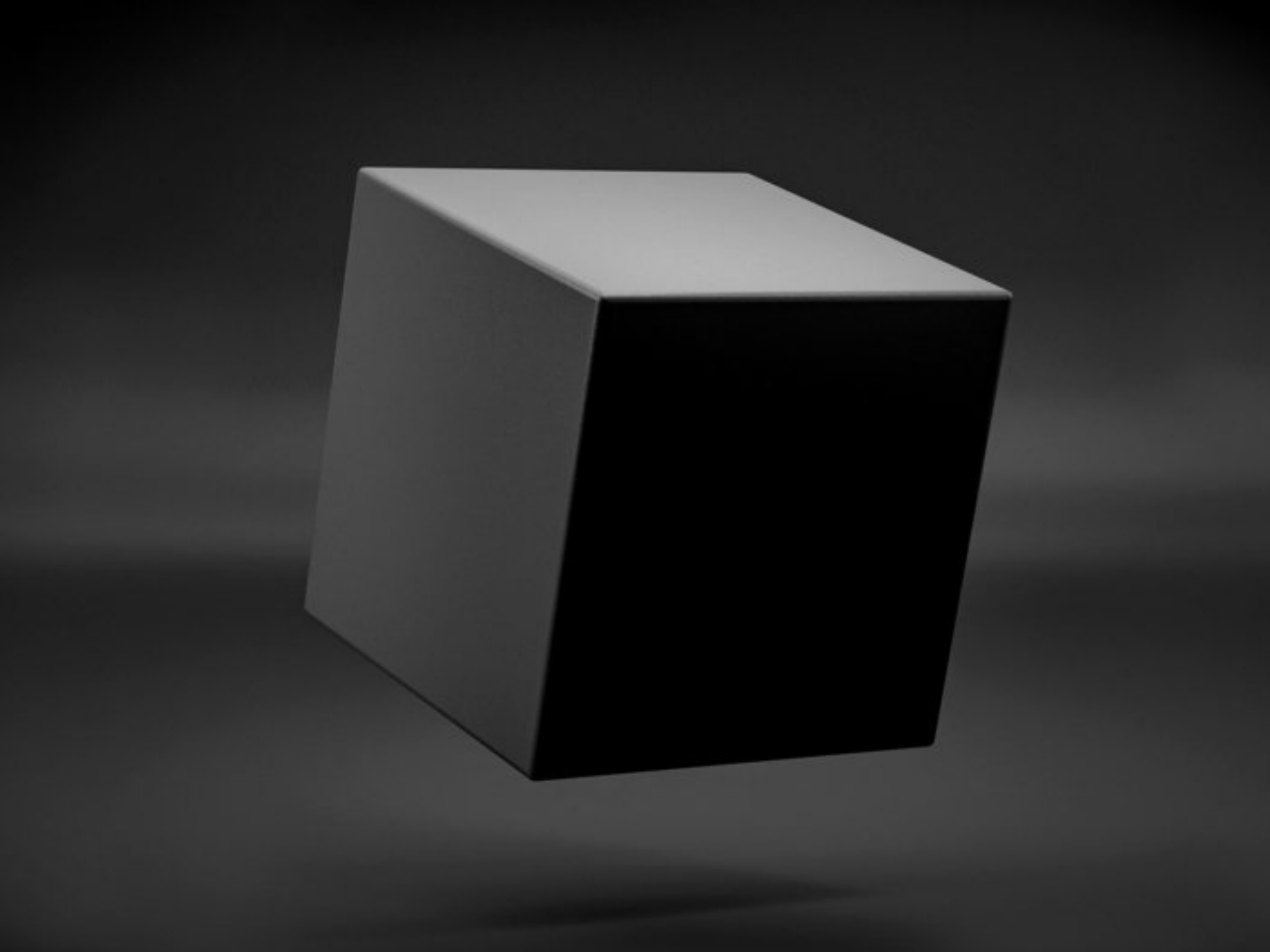
$$\mathrm{SSE} = \sum_{i}^{n} \sum_{j}^{k} w^{(i,j)} ||\mathbf{x}^{(i)} - \mathbf{y}^{(j)}||_2^2$$

$w^{(i,j)}$ is 1 if x(i) belongs to cluster j [sitting at y(j)], otherwise 0

$$\text{SSE} = \sum_{i}^{n} \sum_{j}^{k} w^{(i,j)} ||\mathbf{x}^{(i)} - \mathbf{y}^{(j)}||_2^2$$




1
0.8
0.4
0
-0.4
-0.8
-1
1
1
1
-1
-1
-1
0
0
0
0
0
0
0

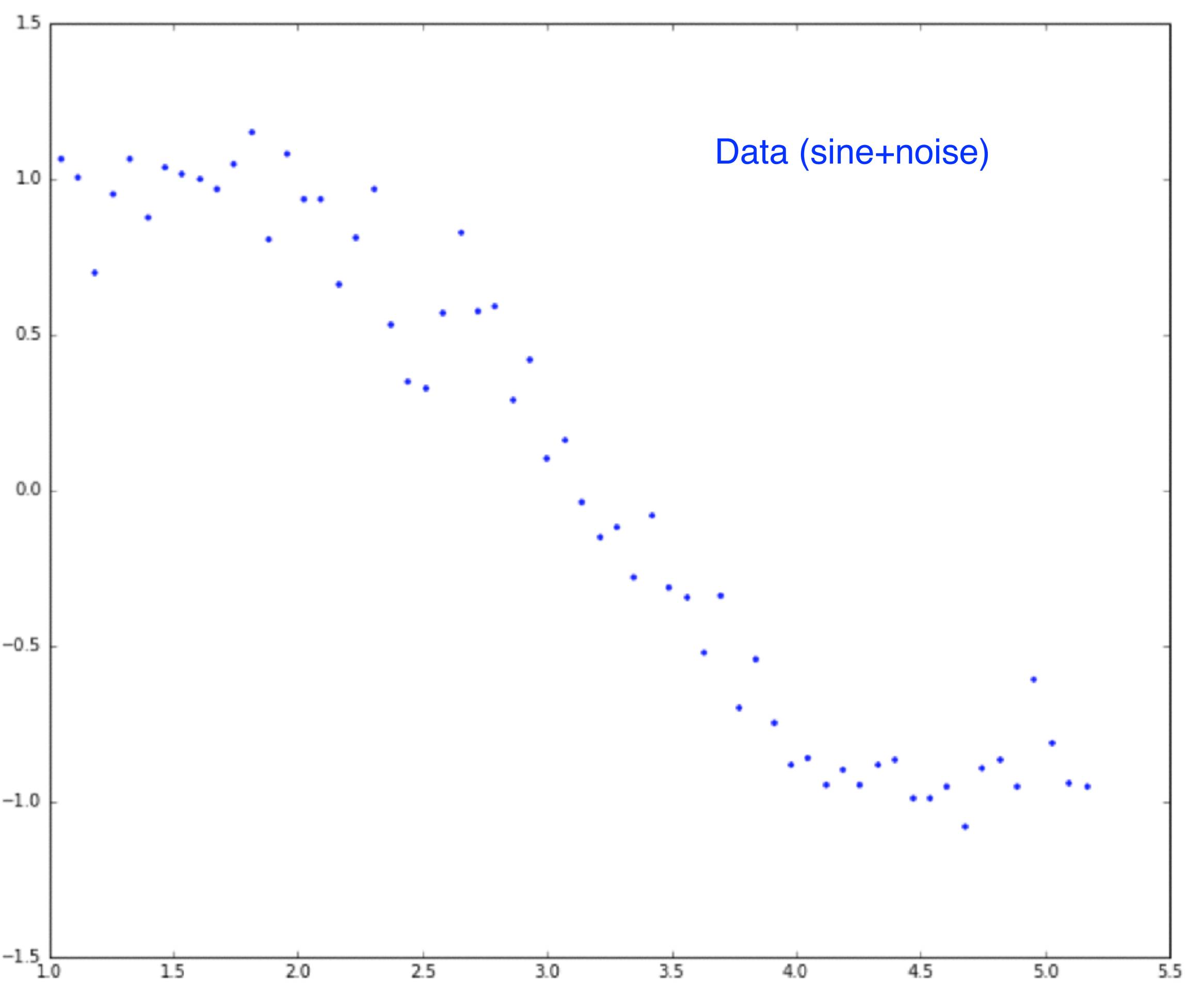


Regularization

Ridge regression

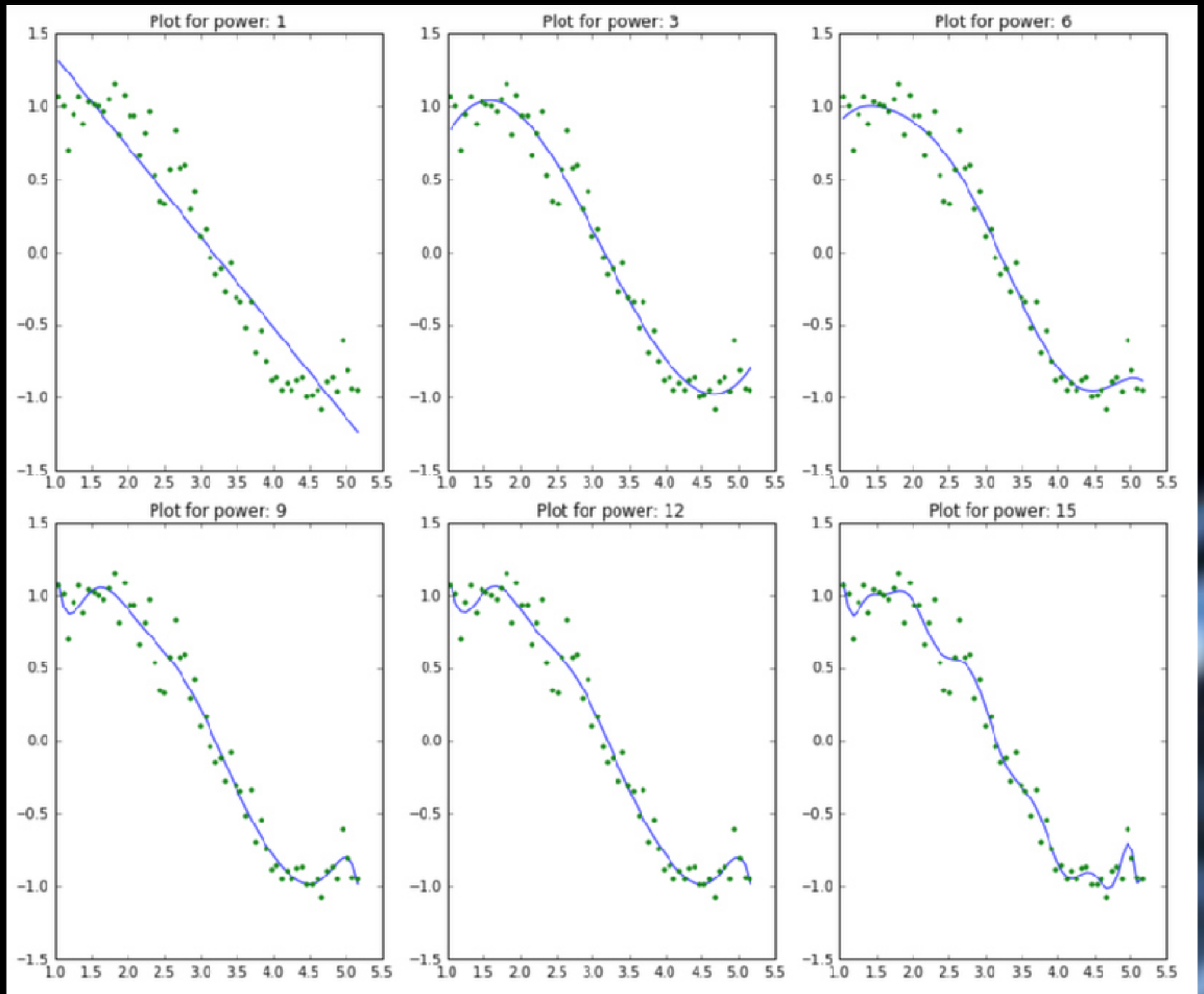
LASSO regression

Elastic Net regression



Courtesy: K. Jain

Fit: OLS Minimization for polynomials of order 1,3,6,9,12,15



Courtesy: K. Jain

Fit coefficients & Residuals
(OLS Minimization for polynomials of order 1-15)

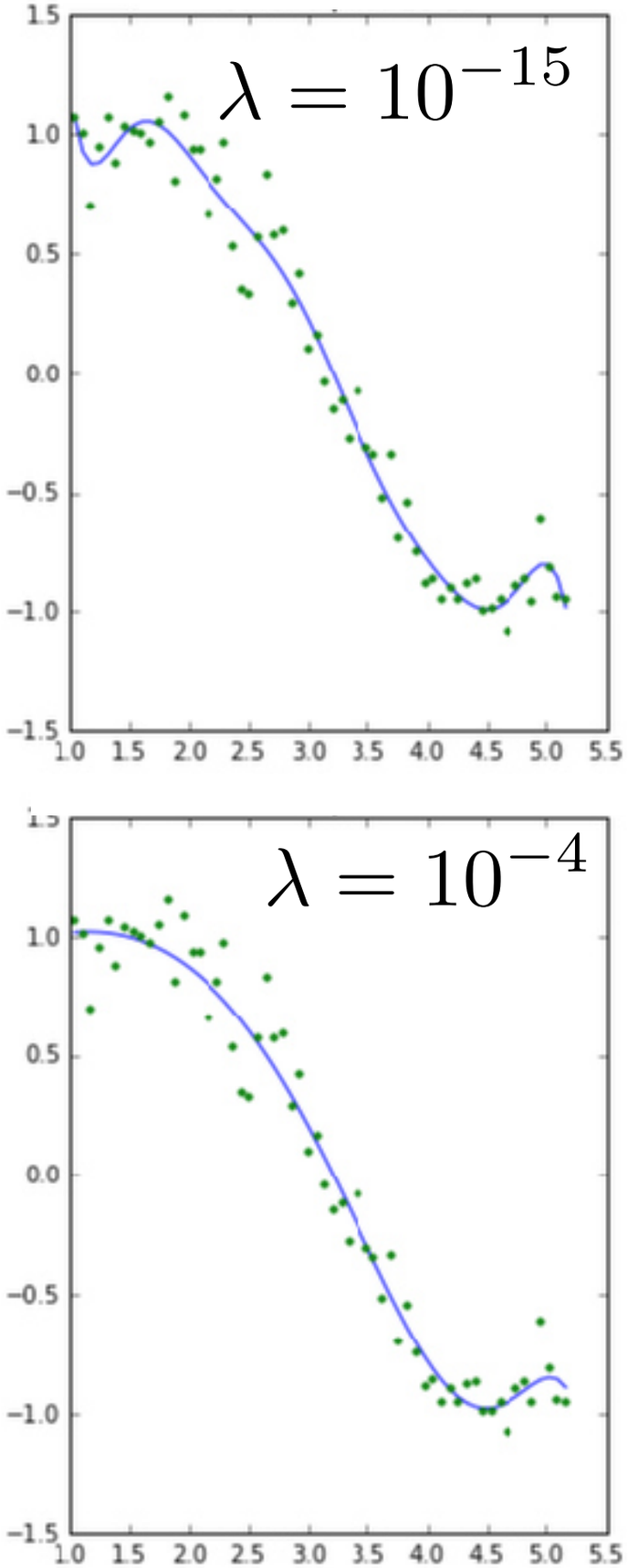
	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_4	1.1	-0.27	1.7	-0.53	-0.036	0.014	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_5	1	3	-5.1	4.7	-1.9	0.33	-0.021	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_6	0.99	-2.8	9.5	-9.7	5.2	-1.6	0.23	-0.014	NaN	NaN	NaN	NaN	NaN
model_pow_7	0.93	19	-56	69	-45	17	-3.5	0.4	-0.019	NaN	NaN	NaN	NaN
model_pow_8	0.92	43	-1.4e+02	1.8e+02	-1.3e+02	58	-15	2.4	-0.21	0.0077	NaN	NaN	NaN
model_pow_9	0.87	1.7e+02	-6.1e+02	9.6e+02	-8.5e+02	4.6e+02	-1.6e+02	37	-5.2	0.42	-0.015	NaN	NaN
model_pow_10	0.87	1.4e+02	-4.9e+02	7.3e+02	-6e+02	2.9e+02	-87	15	-0.81	-0.14	0.026	-0.0013	NaN
model_pow_11	0.87	-75	5.1e+02	-1.3e+03	1.9e+03	-1.6e+03	9.1e+02	-3.5e+02	91	-16	1.8	-0.12	0.0034
model_pow_12	0.87	-3.4e+02	1.9e+03	-4.4e+03	6e+03	-5.2e+03	3.1e+03	-1.3e+03	3.8e+02	-80	12	-1.1	0.062
model_pow_13	0.86	3.2e+03	-1.8e+04	4.5e+04	-6.7e+04	6.6e+04	-4.6e+04	2.3e+04	-8.5e+03	2.3e+03	-4.5e+02	62	-5.7
model_pow_14	0.79	2.4e+04	-1.4e+05	3.8e+05	-6.1e+05	6.6e+05	-5e+05	2.8e+05	-1.2e+05	3.7e+04	-8.5e+03	1.5e+03	-1.8e+02
model_pow_15	0.7	-3.6e+04	2.4e+05	-7.5e+05	1.4e+06	-1.7e+06	1.5e+06	-1e+06	5e+05	-1.9e+05	5.4e+04	-1.2e+04	1.9e+03

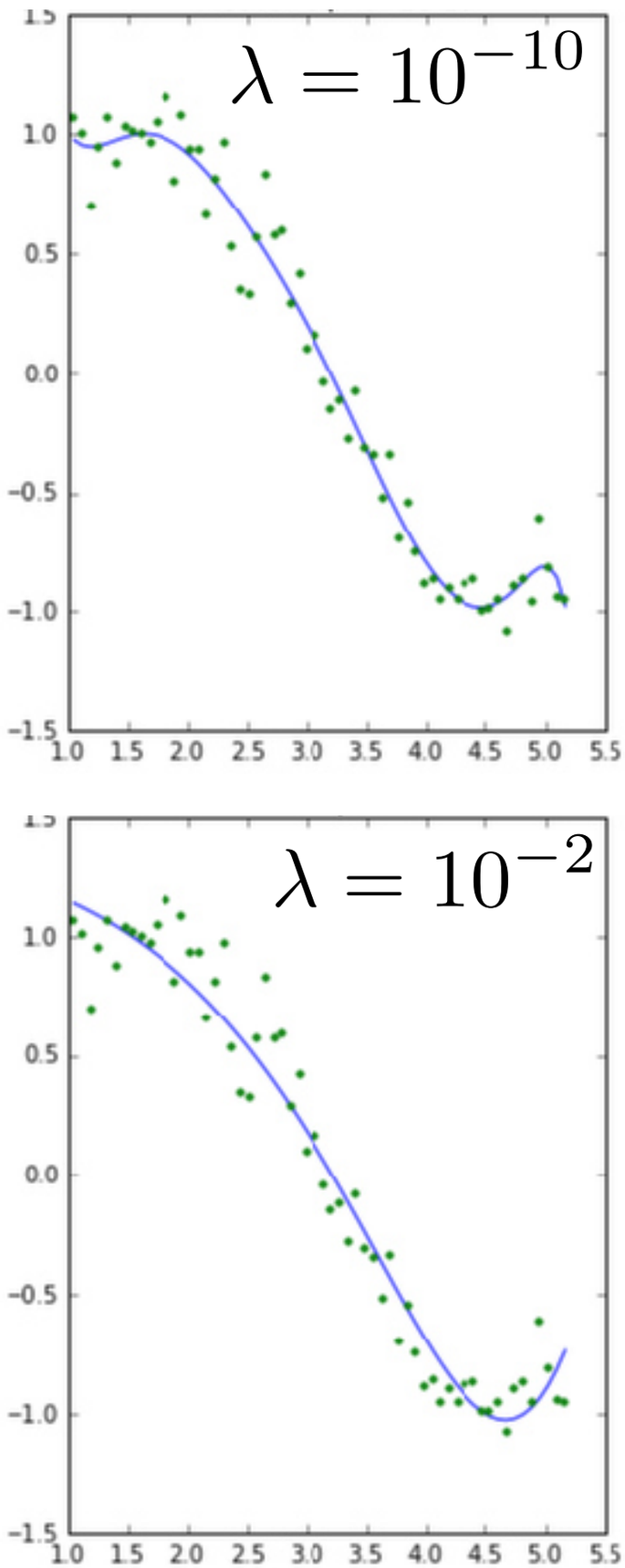
Coefficients increase exponentially with model complexity!

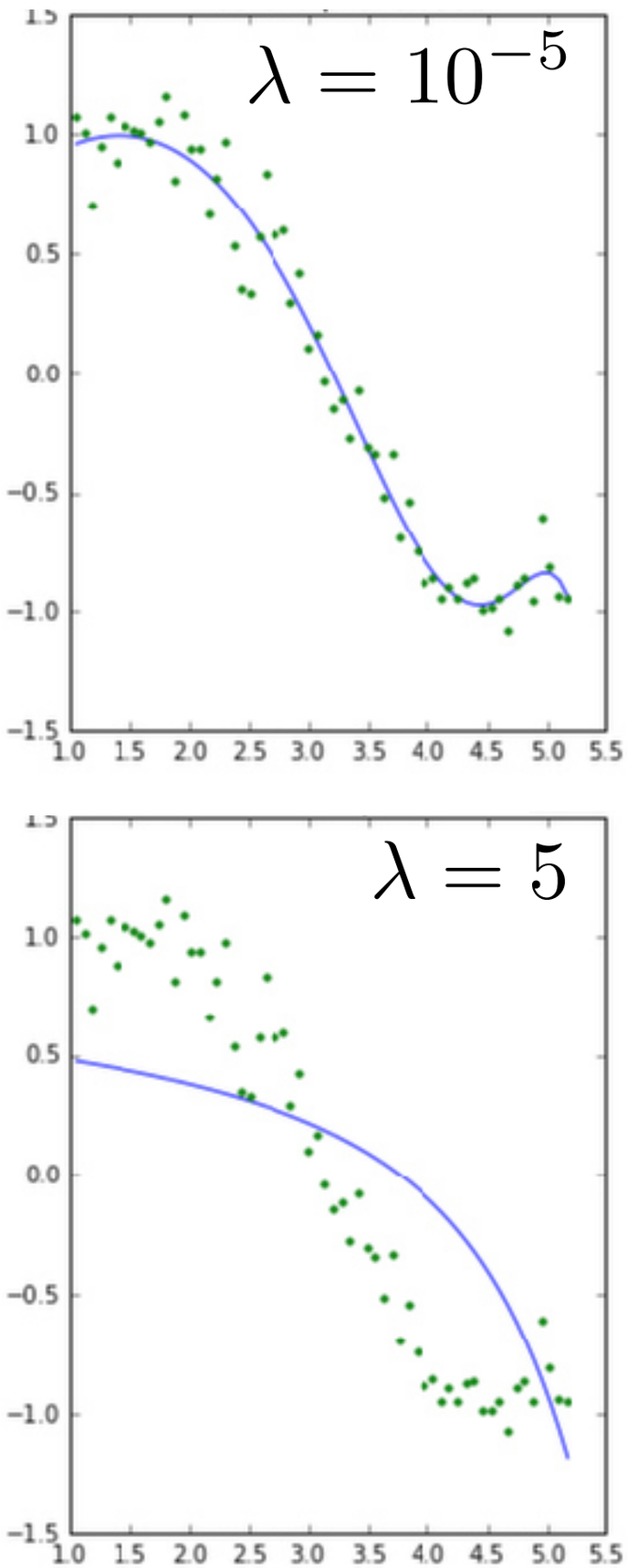
Regularization needed!

Courtesy: K. Jain

Ridge regression (regularization)







Courtesy: K. Jain

Fit coefficients & Residuals

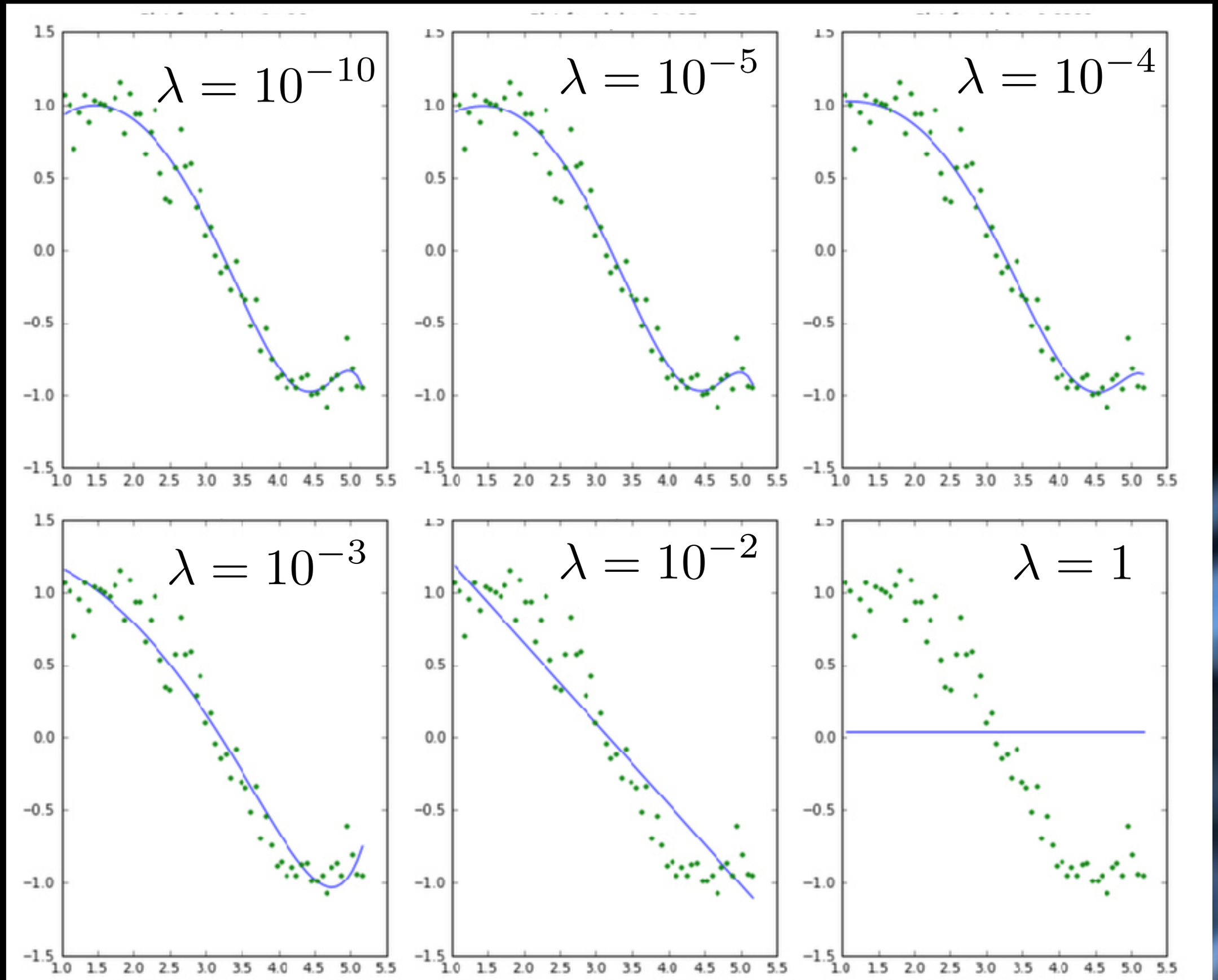
(OLS Minimization with L2 penalty = Ridge regularization)

λ

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	c
alpha_1e-15	0.87	95	-3e+02	3.8e+02	-2.4e+02	66	0.96	-4.8	0.64	0.15	-0.026	-0.0054	0.00086	0.
alpha_1e-10	0.92	11	-29	31	-15	2.9	0.17	-0.091	-0.011	0.002	0.00064	2.4e-05	-2e-05	-4
alpha_1e-08	0.95	1.3	-1.5	1.7	-0.68	0.039	0.016	0.00016	-0.00036	-5.4e-05	-2.9e-07	1.1e-06	1.9e-07	2e
alpha_0.0001	0.96	0.56	0.55	-0.13	-0.026	-0.0028	-0.00011	4.1e-05	1.5e-05	3.7e-06	7.4e-07	1.3e-07	1.9e-08	1.
alpha_0.001	1	0.82	0.31	-0.087	-0.02	-0.0028	-0.00022	1.8e-05	1.2e-05	3.4e-06	7.3e-07	1.3e-07	1.9e-08	1.
alpha_0.01	1.4	1.3	-0.088	-0.052	-0.01	-0.0014	-0.00013	7.2e-07	4.1e-06	1.3e-06	3e-07	5.6e-08	9e-09	1.
alpha_1	5.6	0.97	-0.14	-0.019	-0.003	-0.00047	-7e-05	-9.9e-06	-1.3e-06	-1.4e-07	-9.3e-09	1.3e-09	7.8e-10	2.
alpha_5	14	0.55	-0.059	-0.0085	-0.0014	-0.00024	-4.1e-05	-6.9e-06	-1.1e-06	-1.9e-07	-3.1e-08	-5.1e-09	-8.2e-10	-1
alpha_10	18	0.4	-0.037	-0.0055	-0.00095	-0.00017	-3e-05	-5.2e-06	-9.2e-07	-1.6e-07	-2.9e-08	-5.1e-09	-9.1e-10	-1
alpha_20	23	0.28	-0.022	-0.0034	-0.0006	-0.00011	-2e-05	-3.6e-06	-6.6e-07	-1.2e-07	-2.2e-08	-4e-09	-7.5e-10	-1

Courtesy: K. Jain

LASSO regression (regularization)



Courtesy: K. Jain

Fit coefficients & Residuals

(OLS Minimization with L1 penalty = LASSO regularization)

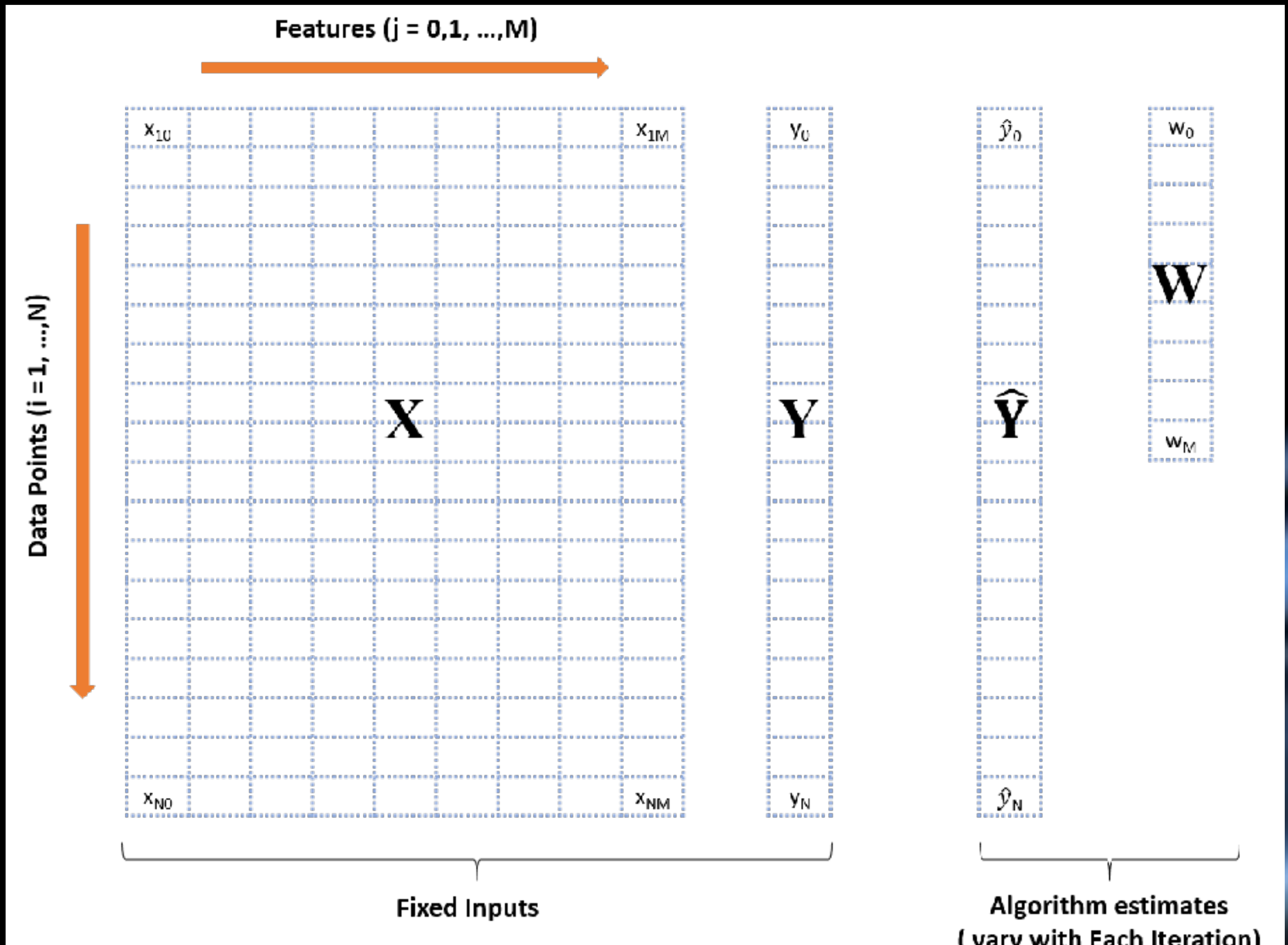
λ

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	coe
alpha_1e-15	0.96	0.22	1.1	-0.37	0.00089	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-10	0.96	0.22	1.1	-0.37	0.00088	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-08	0.96	0.22	1.1	-0.37	0.00077	0.0016	-0.00011	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.3
alpha_1e-05	0.96	0.5	0.6	-0.13	-0.038	-0	0	0	0	7.7e-06	1e-06	7.7e-08	0	0
alpha_0.0001	1	0.9	0.17	-0	-0.048	-0	-0	0	0	9.5e-06	5.1e-07	0	0	0
alpha_0.001	1.7	1.3	-0	-0.13	-0	-0	-0	0	0	0	0	0	1.5e-08	7.5
alpha_0.01	3.6	1.8	-0.55	-0.00056	-0	-0	-0	-0	-0	-0	-0	0	0	0
alpha_1	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_5	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_10	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0

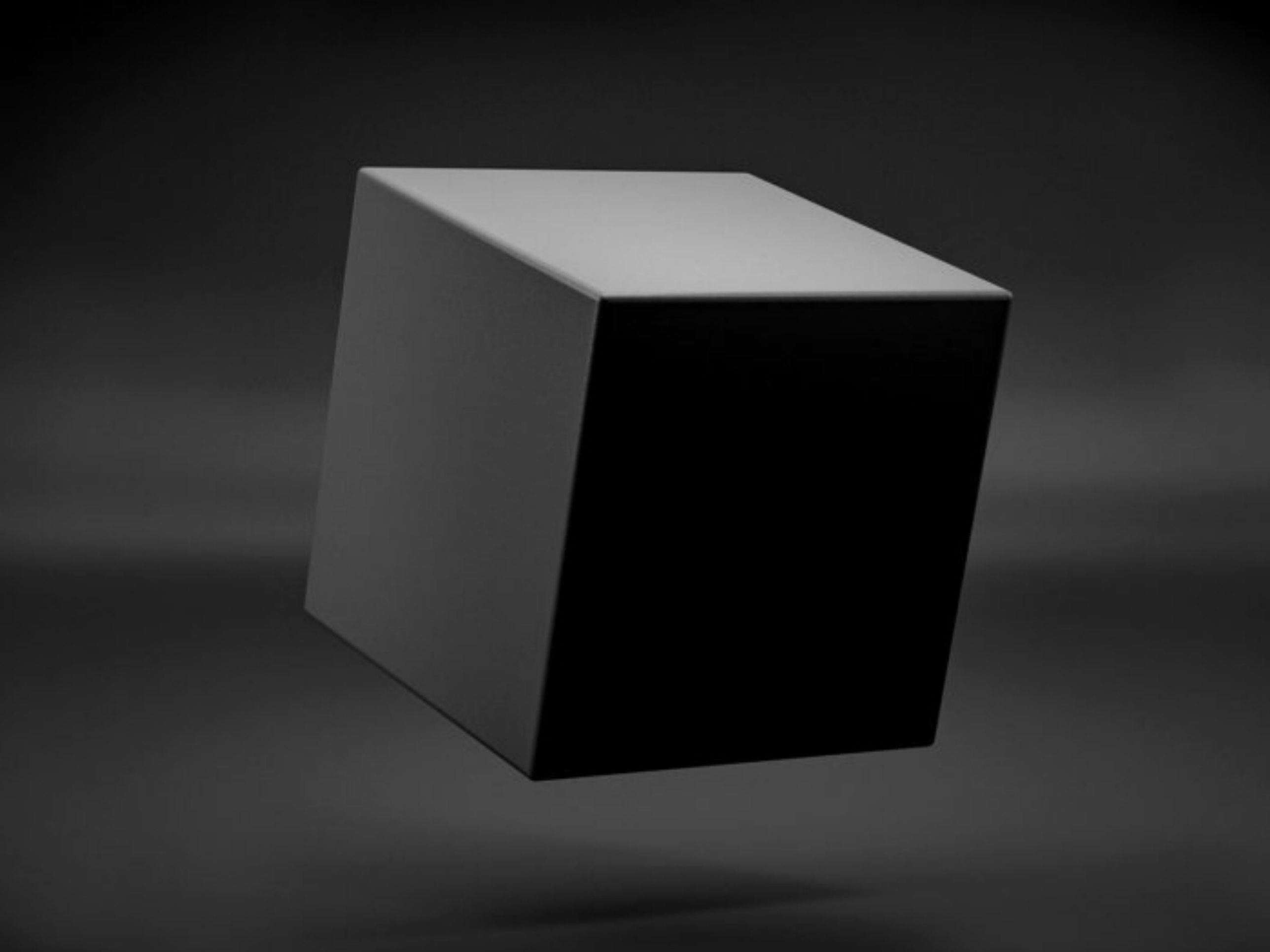
HIGH SPARSITY

Courtesy: K. Jain

Regression: (more) general case



Courtesy: K. Jain



Machine Learning II
Week #2

Jan Nagler

Deep Dynamics Group
Centre for Human and Machine Intelligence (HMI)
Frankfurt School of Finance & Management

(Details in lecture!)

Frankfurt, October 2019

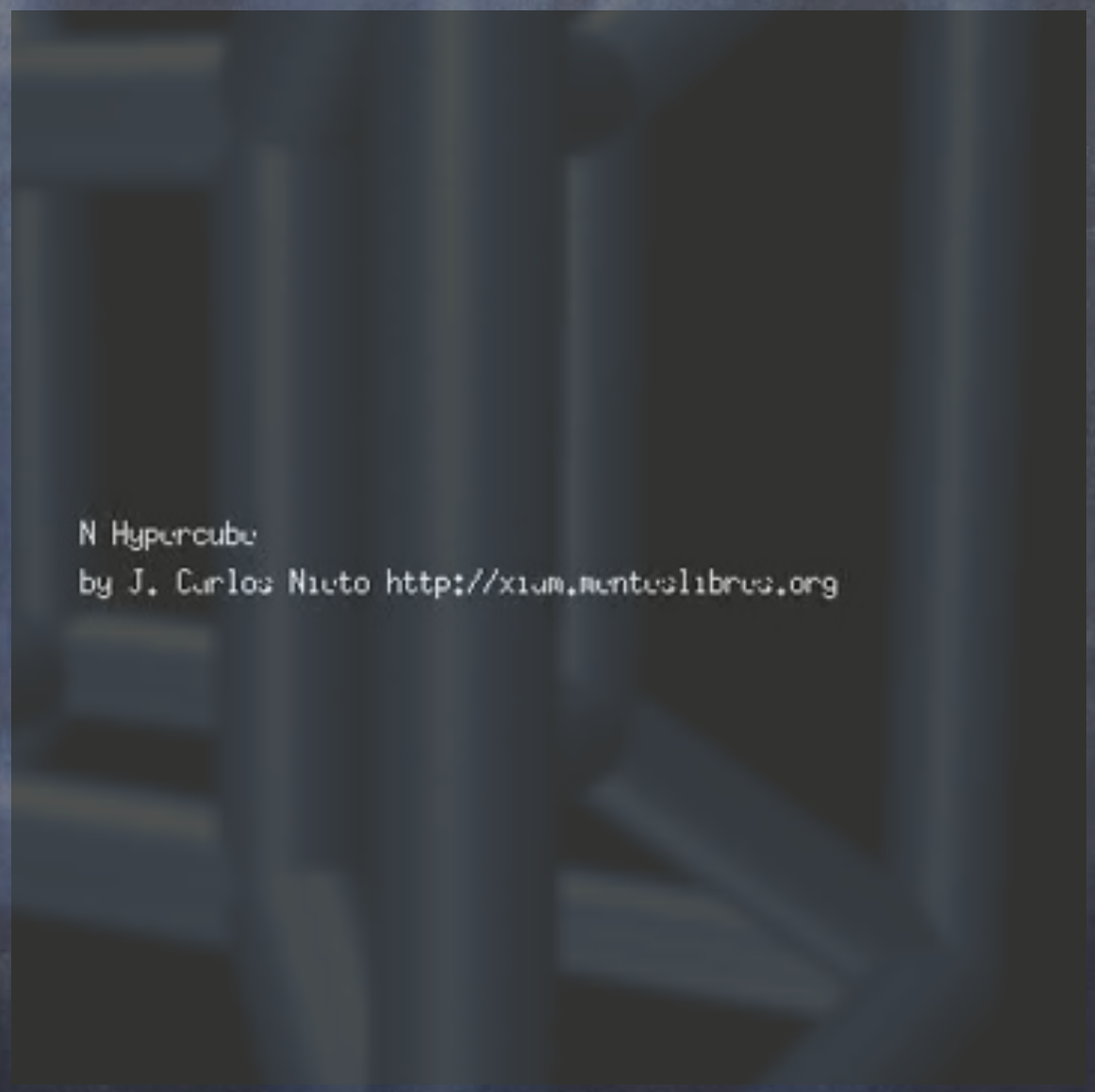
Assignments#1—Comments

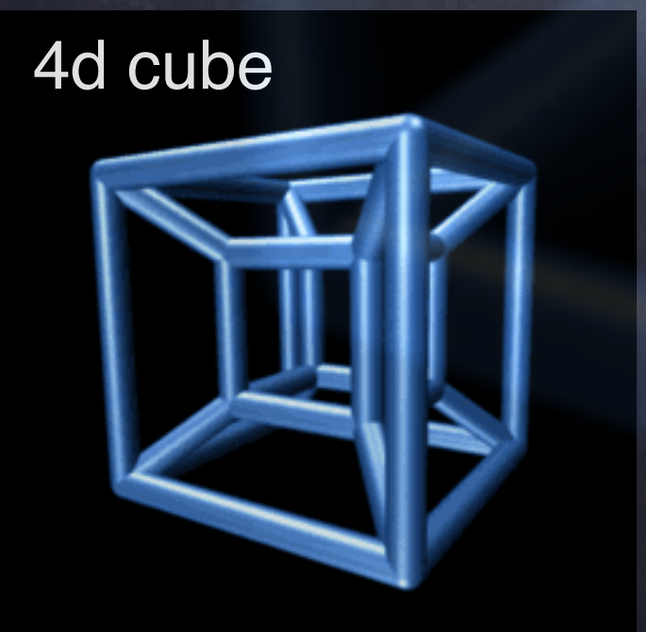
Use Email as requested
Most comments in class

Presentation:
Read assignment,
Avoid useless graphs,
avoid pdf graphs over page limit,
avoid useless copy and paste,
answer the questions and do not leave irrelevant stuff in,
make the code your code,
only submit pdf and code (no html)
add your full name to filenames

Lecture:
Solution presentation: Week #1, based on submitted code,
66% for A1

Curse of Dimensionality revisited





Curse of Dimensionality revisited

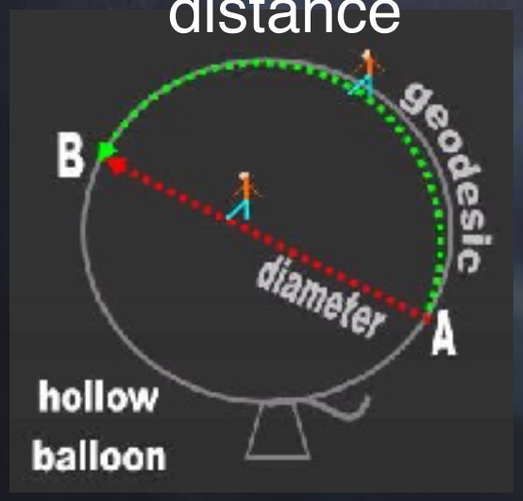
Combinatorial explosion

Vanishing volume ratio
(Hypersphere versus hypercube)

Vanishing center domain
(Distance random to corner)

Physical embedding / Meaning of orthogonality
Physical space, chemical concentrations, asset prices,
eye color, age, IQ … often relate to objects in space

good & bad & ugly
distance



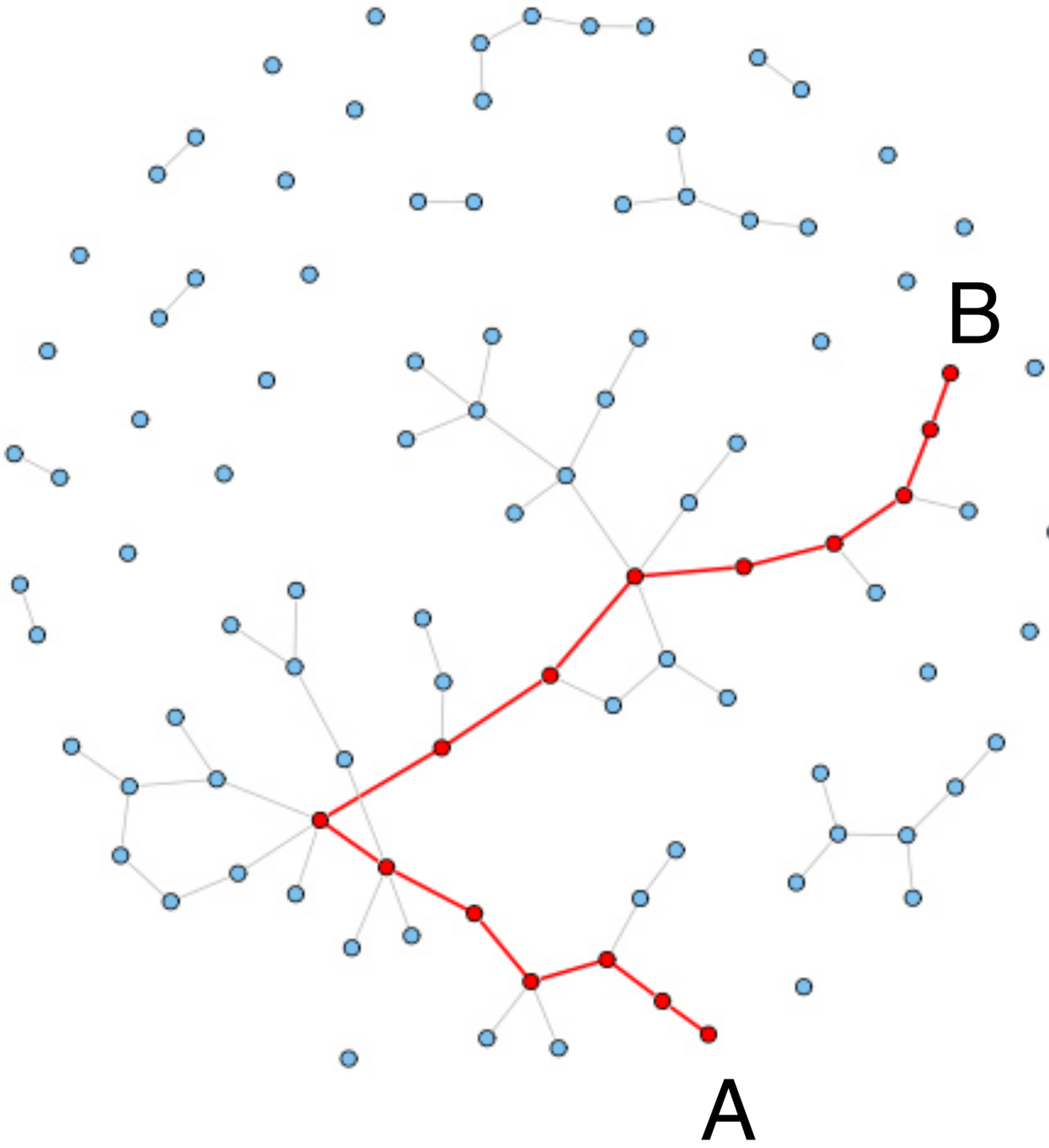
— types: nominal, ordinal, ratio, text,
image, video, audio, …

—distance: can be undefined, meaningless,
totally unknown or ambiguous but needs to be
well chosen

Curse of metric distance

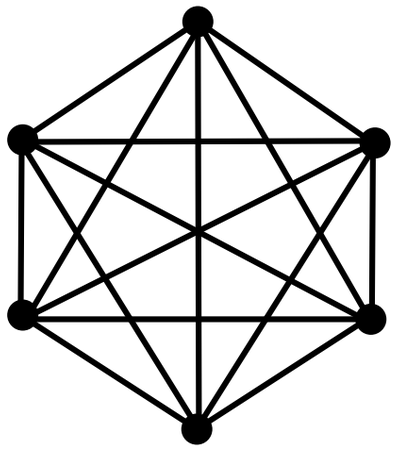
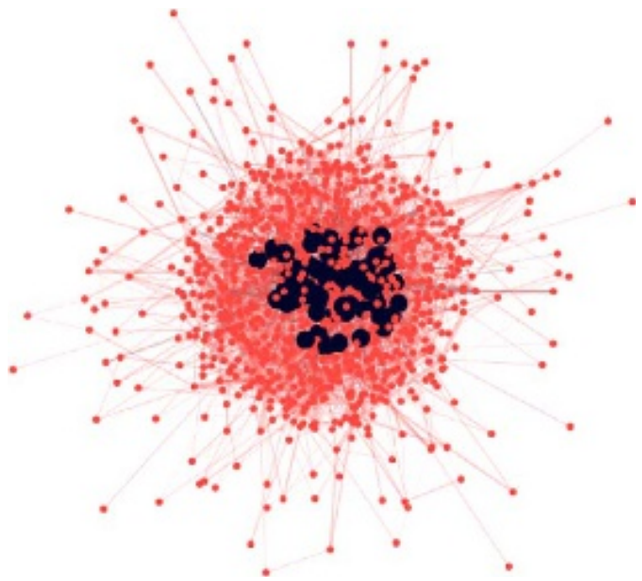
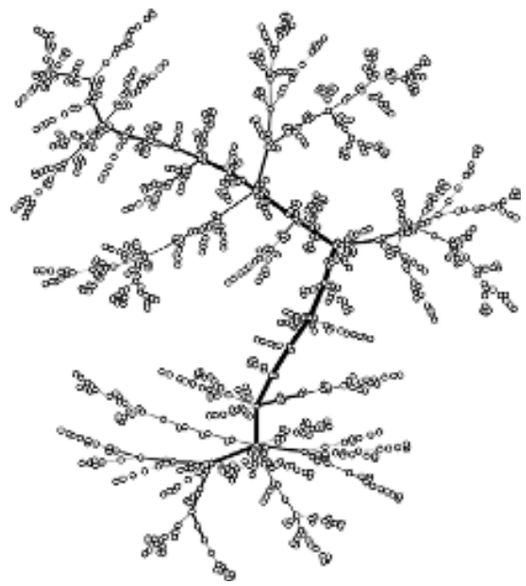
Curse of data types

Network distance metrics



Network metrics

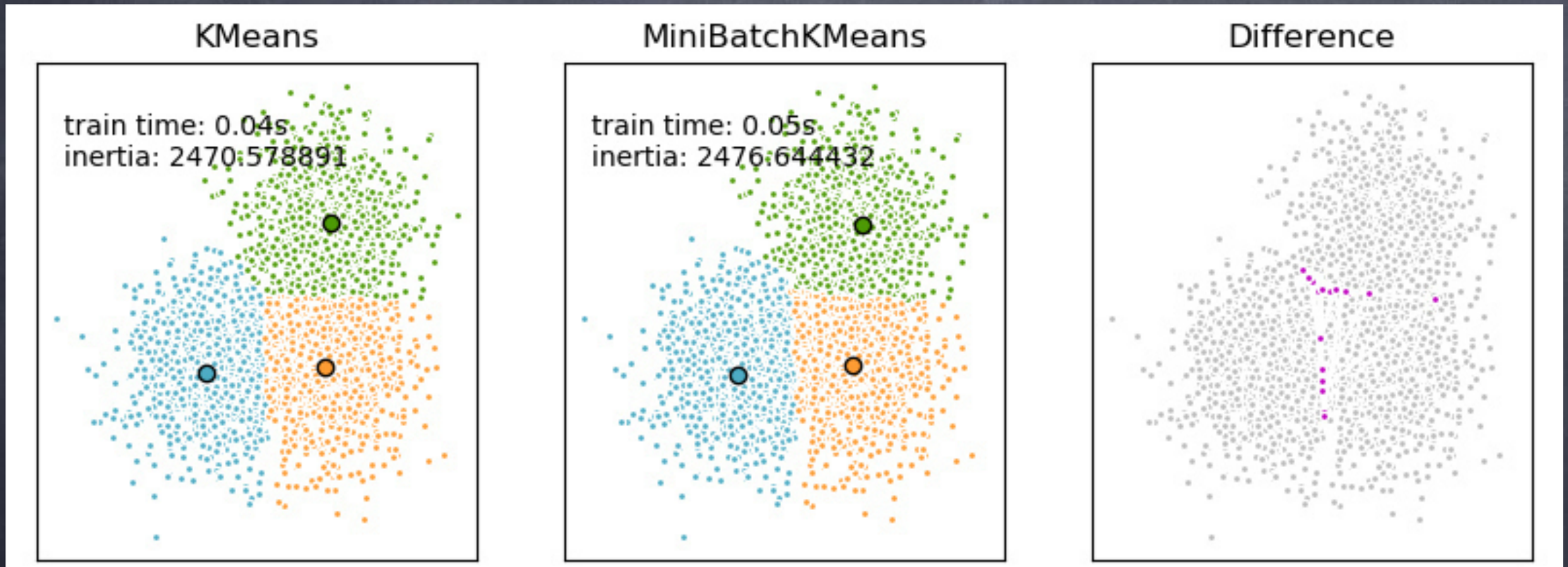
fully connected | real networks | tree

	Connectivity driven : A	Critical window : B	Utility driven : C
Pruning model			

Clustering monster

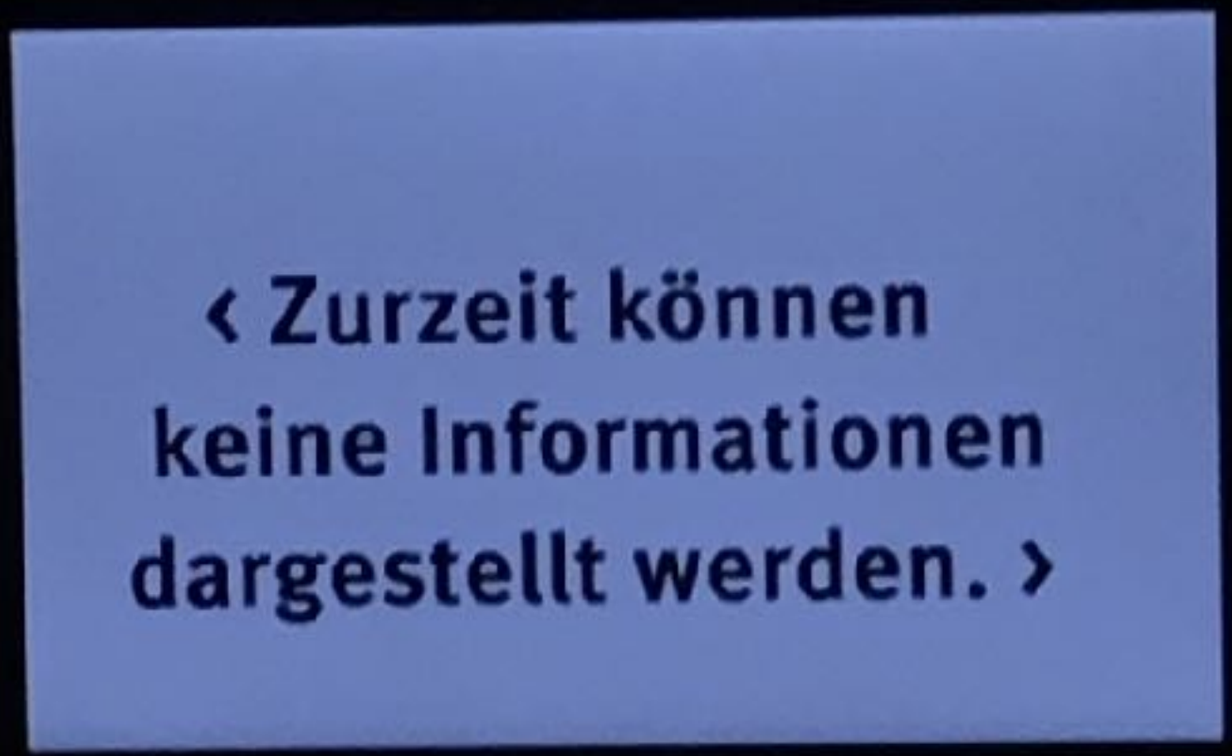
Python in class

Example: Difference (output) K-Means vs Mini Batch K-Means



Expectation Maximization (EM)

Expectation Maximization



Conflicting bus display:
“Currently no information can be displayed”

EM based on
information theory

Expectation Maximization

Sort of soft clustering approach

Example #1:
Gaussian Mixure Model

More general theoretical framework

Example #2:
Coin toss

FS ML2 Class Clustering

(creative joint paper and team work project)

Group red: Coding

Group green: Methods

Group blue: Distances

Group blue

You are the distance group: Self-learn clustering methods and present the main ideas to the class

Group blue

The right (metric) measure:
What distance??

Euklidean? What p(ower)-norms?

Based on correlation & similarity?
Cosine, Mahalanobis distance,
Jaccard, Edit distance

Neighrest k neighbors-manifold based:
Isomap

Add cr(azy)(eative) ideas:
Randomized component pick?
How to determine the importance of components? Given data, what metric?

Study but focus:
Main ideas behind the distances

Resources:
Use web (re)sources
(skikit-learn, medium,
kdnuggets, dzone, wiki,
google)

Self-organize:
Split up the tasks

Final presentation

Be incomplete, choose what you like!

Group green

You are the methods group:
Self-learn methods and
present the main ideas to the class

Group green

Presentation of clustering methods

Monster(*) methods, included

Keyword guide to the cluster galaxy:

Inference/Expectation Maximization

Neigherst Neighbors

Hard / Soft Clustering

Closet point

Intercluster distance

Cohesion

Maximal distance from “Clustroid”

Cluster diameter

Density

Stopping criteria

Noise points / Retained sets

Study but focus:
Main ideas behind the methods

Resources:
Use web (re)sources
(skikit-learn, medium,
kdnuggets, dzone, wiki,
google)

Self-organize:
Split up the tasks

Final presentation:
Be brief & clear
pro / cons of methods

Be incomplete, choose what you like!

(*tbd)

Group red

You are the c0d1ng group: implementation, embedding in monster, optimization, modifications, exploration of other ideas

Gro

Explosive cluster merging (ECM)

Create ordered list of distances $\{d_1, d_2, d_3, \ldots, d_n\}$

$$d_{(.)} = d_{ij} = d(x_i, x_j)$$

Epsilon loop in delta steps $\varepsilon = \delta, 2\delta, 3\delta, \ldots$ $\delta \sim \langle d_{ij} \rangle$

Among $d(x_i, x_j) < \varepsilon$

Choose k distances at random

Choose pair (avoid large clusters) $i^\star,\ j^\star = argmin_{ij}^{k \text{ pairs}}[C_i * C_j]$

$$C_i = \text{ size of class/cluster}$$

Merge clusters/classes

$$C_i + C_j \Rightarrow C_i;\ n \to n-1$$

Monitor #classes as a function of steps (Elbow plot)

Optimization

$$\mathbb{O}(n^2)$$

Explosive Clustering

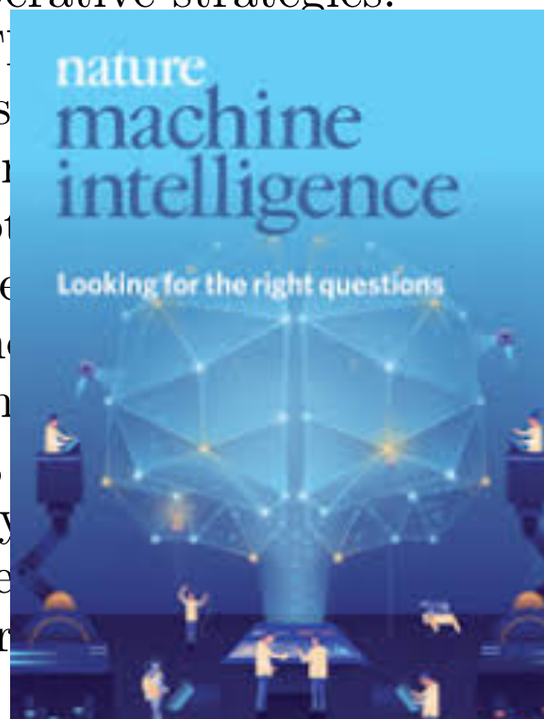
Michelle Liu, Kulkarni Nupur, Loa Marx, Theresa Pareno, Juliet Black, Felix Schmidt, Victor Johannes Holl, Huiting Xu, Wen-Hui Ting, Si-Tong Ye, Dominik Bette, Ananya Neogi, Erik Lehman, Maximillian Vogt, Samar Habibi, Zakariya Yousef Abu-Grin, Jan Nagler

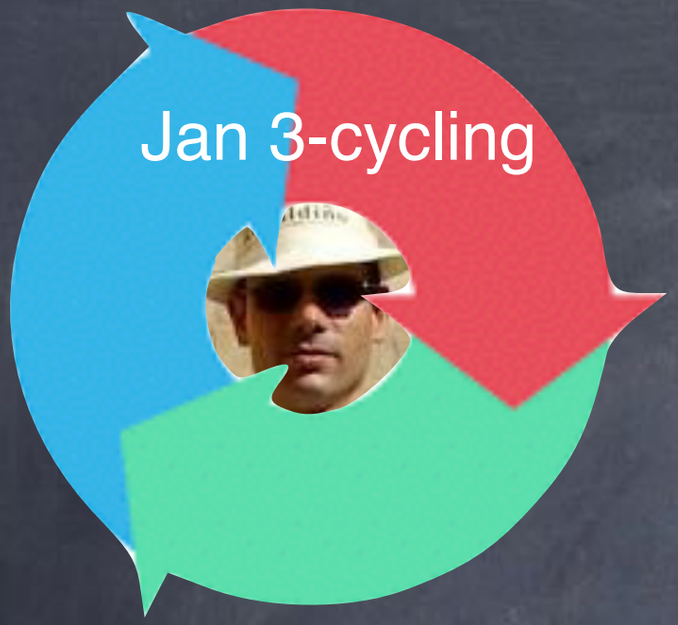
Deep Dynamics Group & Centre for Human and Machine Intelligence,
Frankfurt School of Finance & Management, Frankfurt, Germany

We propose a new clustering algorithm based on *the power of choice*, where neighboring clusters are being merged, yet mergers of large clusters are delayed as much as possible. The idea is based on the DBSCAN algorithm but the proposed algorithm is not based on sample density. Yet, it is ten quadrillions times faster on any Aldi quantum computer, in case Tensorflow 3++ runs smoothly. This links the theory of explosive phase transitions on networks with efficient clustering.

This text here needs to be written. I am very sorry about that. It is a recent submission in review and resubmit mode. Basta! Iterated games are models for sentient, evolutionary behaviors and the emergence of cooperation. In iterated two-player games two players interact repeatedly aiming to use the best conditional strategies against each other [1]. However, until the landmark paper by Press and Dyson [2], it was generally assumed that there exists no conditional cooperator that, independently of the opponent's actions, ensures an equal or higher expected payoff in the long run. They proposed a new two-player class, so-called zero-determinant (ZD) strategies, that enforces a linear relationship between one's own strategy and that of the other player. A subset of those strategies, the so-called extortion (extZD) strategies, can unilaterally enforce an expected long-term payoff that is larger than or equal to those of *any* other strategy, extZD included, for action space noise int
ties ranging from zero to a value where all action
come equivalent (meaning totally random). In add
we show that degenerated mirror strategies may be
erous when mirroring cooperative strategies.

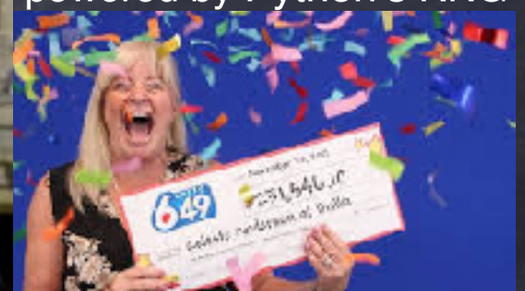
Prisoner's Dilemma T
two-player game that aims
cooperation in a selfish wor
tional individuals might not
that it is in their best inte
each player chooses wheth
(d). Throughout the man
tive payoff notation in [2],
the perspective of each play
(cc), they each receive a re
ates while Y defects (cd fr
from the Y's perspective),





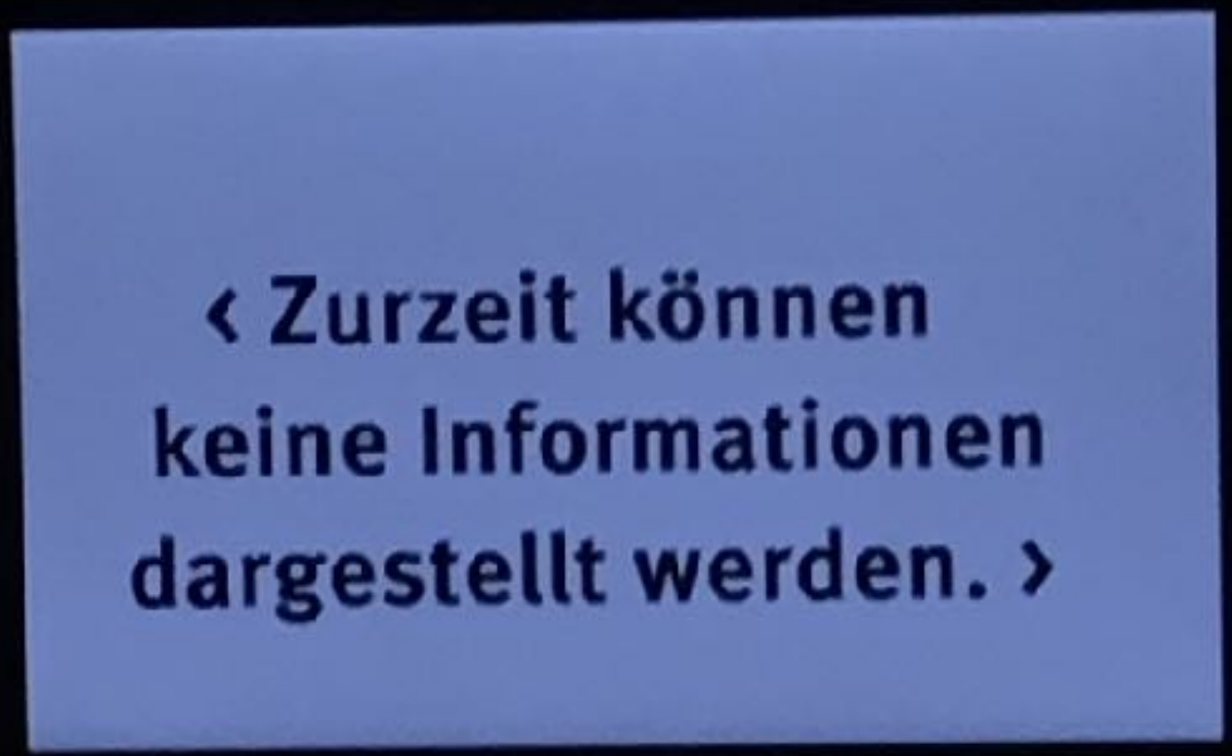
2 Polls & Lotteries!

powered by Python’s RNG



Expectation Maximization (EM)

Expectation Maximization



Conflicting bus display:
“Currently no information can be displayed”

EM based on
information theory

Machine Learning II
Week #3

Jan Nagler

Deep Dynamics Group
Centre for Human and Machine Intelligence (HMI)
Frankfurt School of Finance & Management

(Details in lecture!)

Frankfurt, October 2019

Presentation & Discussion

group red

in lecture

Mahalanobis distance add on:

to point cloud: $$d(\mathbf{x}) = \left[(\mathbf{x}-\mu)^T S^{-1}(\mathbf{x}-\mu)\right]^{1/2}$$

between x and y: $$d(\mathbf{x},\mathbf{y}) = \left[(\mathbf{x}-\mathbf{y})^T S^{-1}(\mathbf{x}-\mathbf{y})\right]^{1/2}$$

Consequence, if cov matrix S diagonal?

Expectation Maximization (EM)

Comments to Solutions #2:
Take always warnings in python into consideration for updating your code, or, if not possible, disable them:

```
import warnings
warnings.filterwarnings("ignore")
```

Avoid messy output with highly insignificant digits:

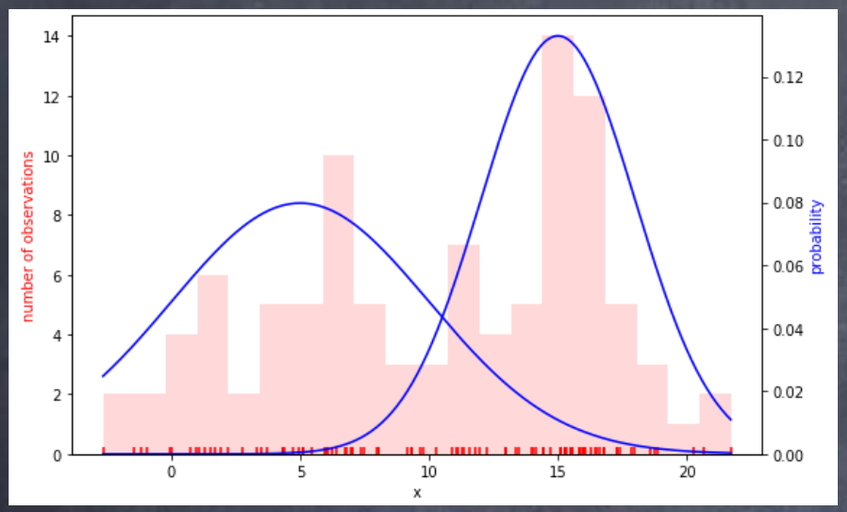
```
ations: 40  Mean 1: 0.014638056849572876  Standard Deviation: 2.
266237  Mean2: 5.00101576331132  Standard Deviation2: 4.94574044
ations: 41  Mean 1: 0.014638486508665 86  Standard Deviation: 2.0
0957  Mean2: 5.001015294334565  Standard Deviation2: 4.945740860
ations: 42  Mean 1: 0.014638761386177682  Standard Deviation: 2.
```

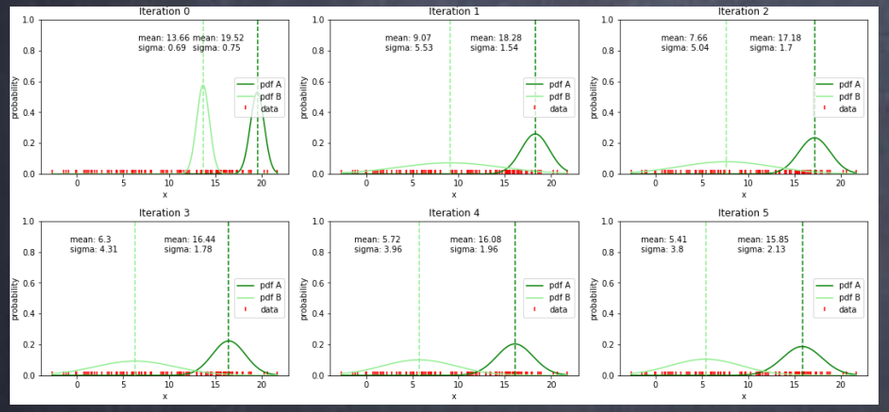
filenames w/ your name!

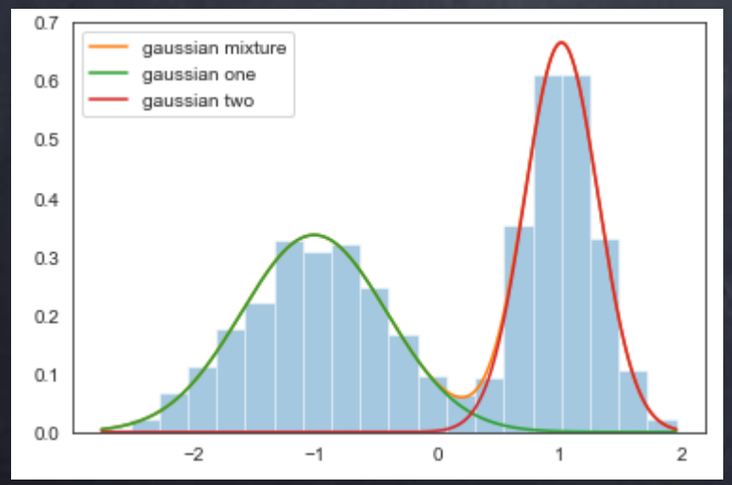
Clean `imports`

Expectation Maximization (EM)

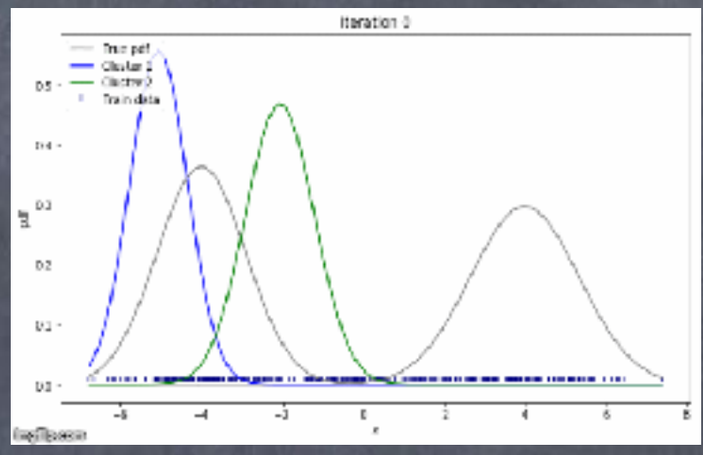
Gaussian Mixure Model: Solution presentation in lecture



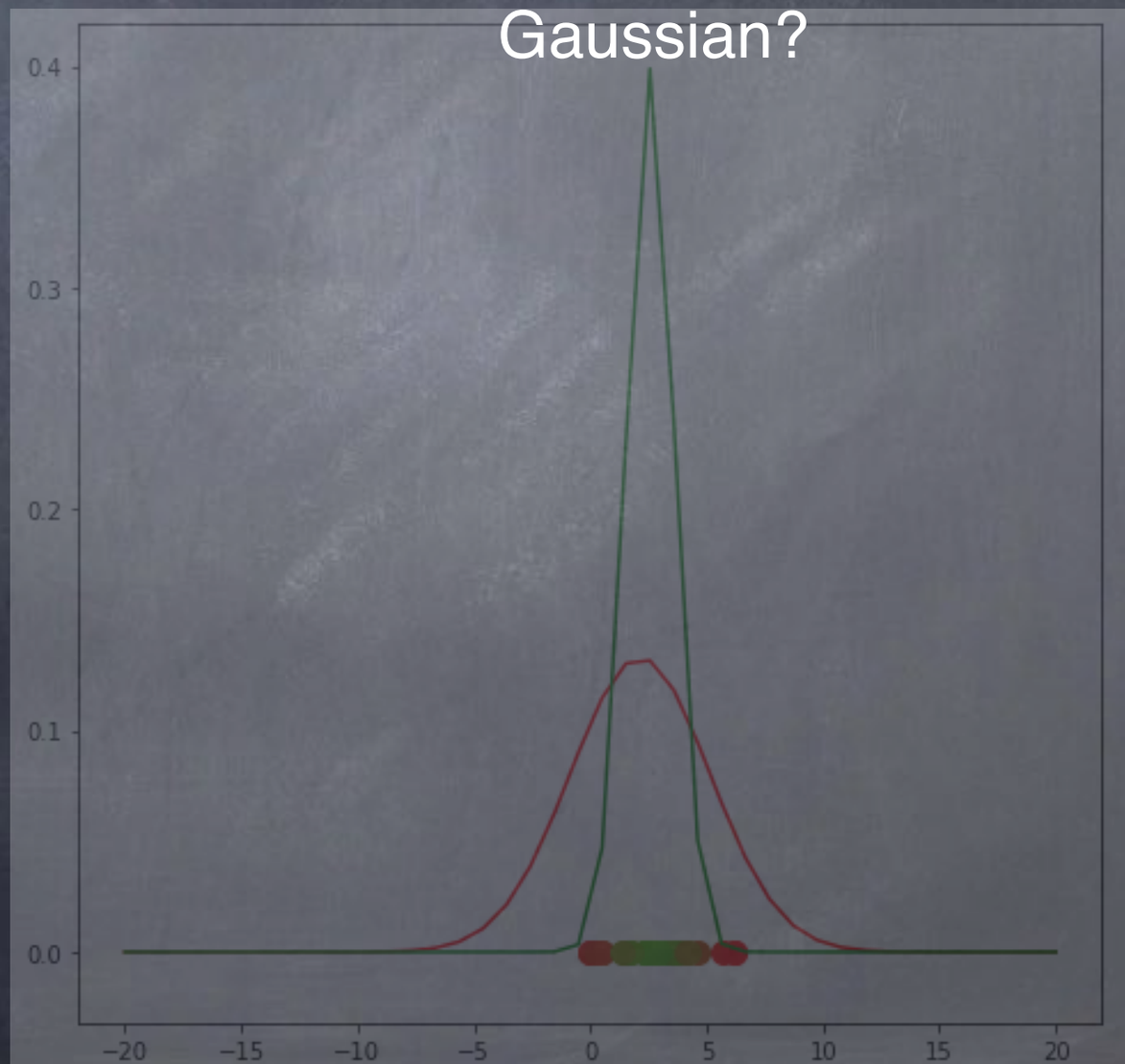




Comments to Solutions #2



Gaussian?



Expectation Maximization (EM)

Example #1:
Gaussian Mixure Model

More general theoretical framework:

Example #2:
Coin toss

Expectation Maximization

Sort of soft clustering approach

Mixure models
Generative “model”, means pdf
typically each cluster is Gaussian or Multinomial (e.g., coin)

Hidden / latent variables:
Parameters of pdf

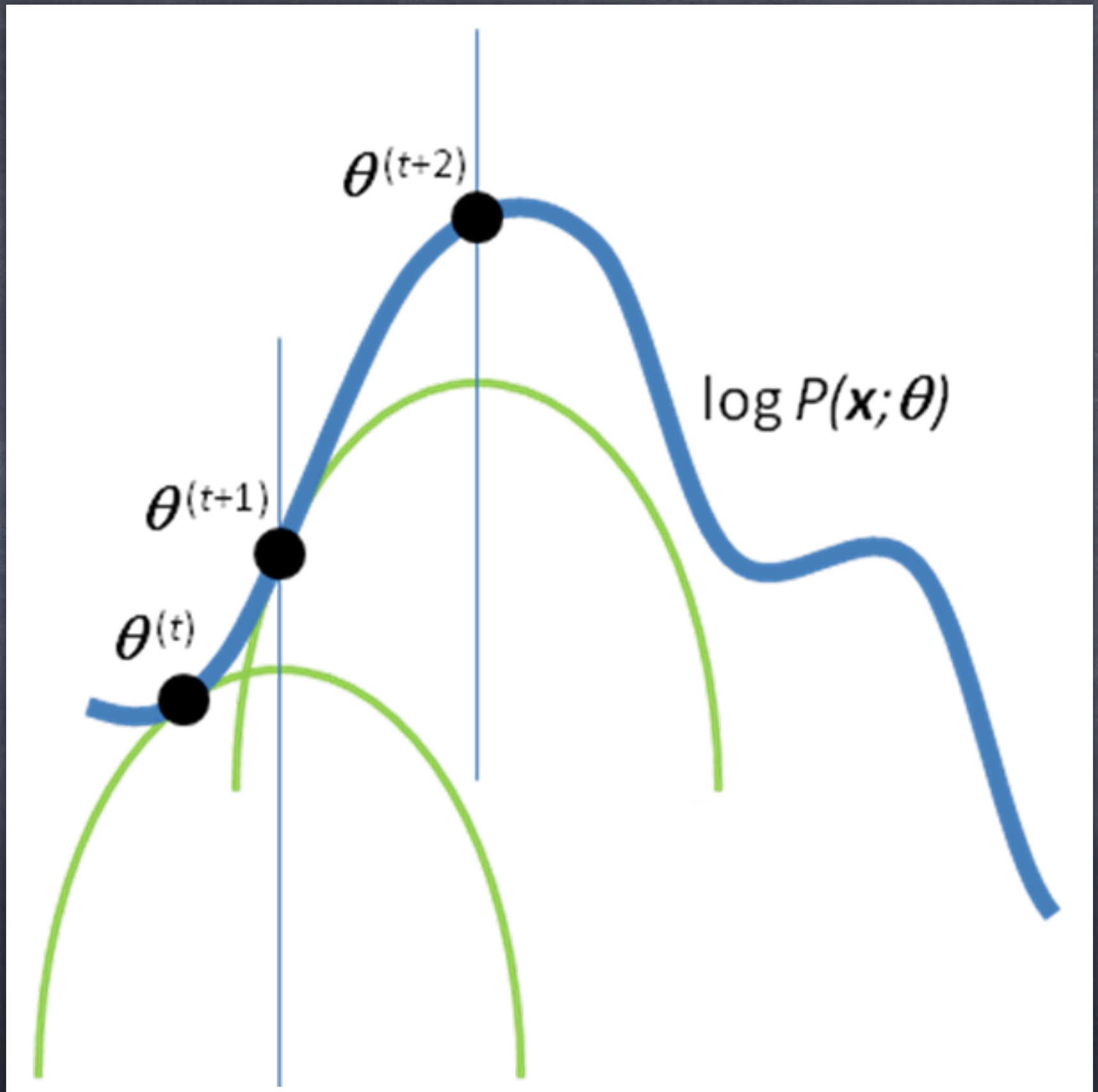
Gaussians: mean & co(variance)
Multinomial: probabilities of events

EM estimates iteratively the parameters,
by the lower bound trick that
maximizes the log(pdf[data given parameters])

$$\log p(x|\theta)$$

Theory and example in lecture

Expectation Maximization



Expectation Maximization (EM) — Coin flips

$$Q(\theta|\theta') = E\left[\log \Pi_{i\leq n}[p_A\ p^{x_i}(1-p)^{1-x_i}]^{z_i}[p_B\ q^{x_i}(1-q)^{1-x_i}]^{1-z_i}\right]$$

$$= \sum_{i\leq n} E[z_i|x_i,\theta^{(t)}][\log(p_A) + x_i \log p + (1-x_i)\log(1-p)]$$

$$+(1-E[z_i|x_i,\theta^{(t)}])[\log p_B + x_i \log q + (1-x_i)\log(1-q)]$$

From this, we seek: $\frac{\partial Q}{\partial p_A} = 0$ $\frac{\partial Q}{\partial p} = 0$ $\frac{\partial Q}{\partial q} = 0$

Consequences developed in lecture

Regularization

LASSO regression

Ridge regression

Elastic Net regression

Regularization

Ridge, Lasso and Elastic Net regression are seeking to alleviate the consequences of correlations / dependencies / multicollinearity of fit parameters

Aim is a parsimonious model (low variance, low bias)

Regularization imposes an upper threshold on the values taken by the coefficients, thereby producing a more parsimonious solution, and a set of coefficients with smaller variance, together with negative coefficient correlations.

Examples & Theory developed in class

Regularization

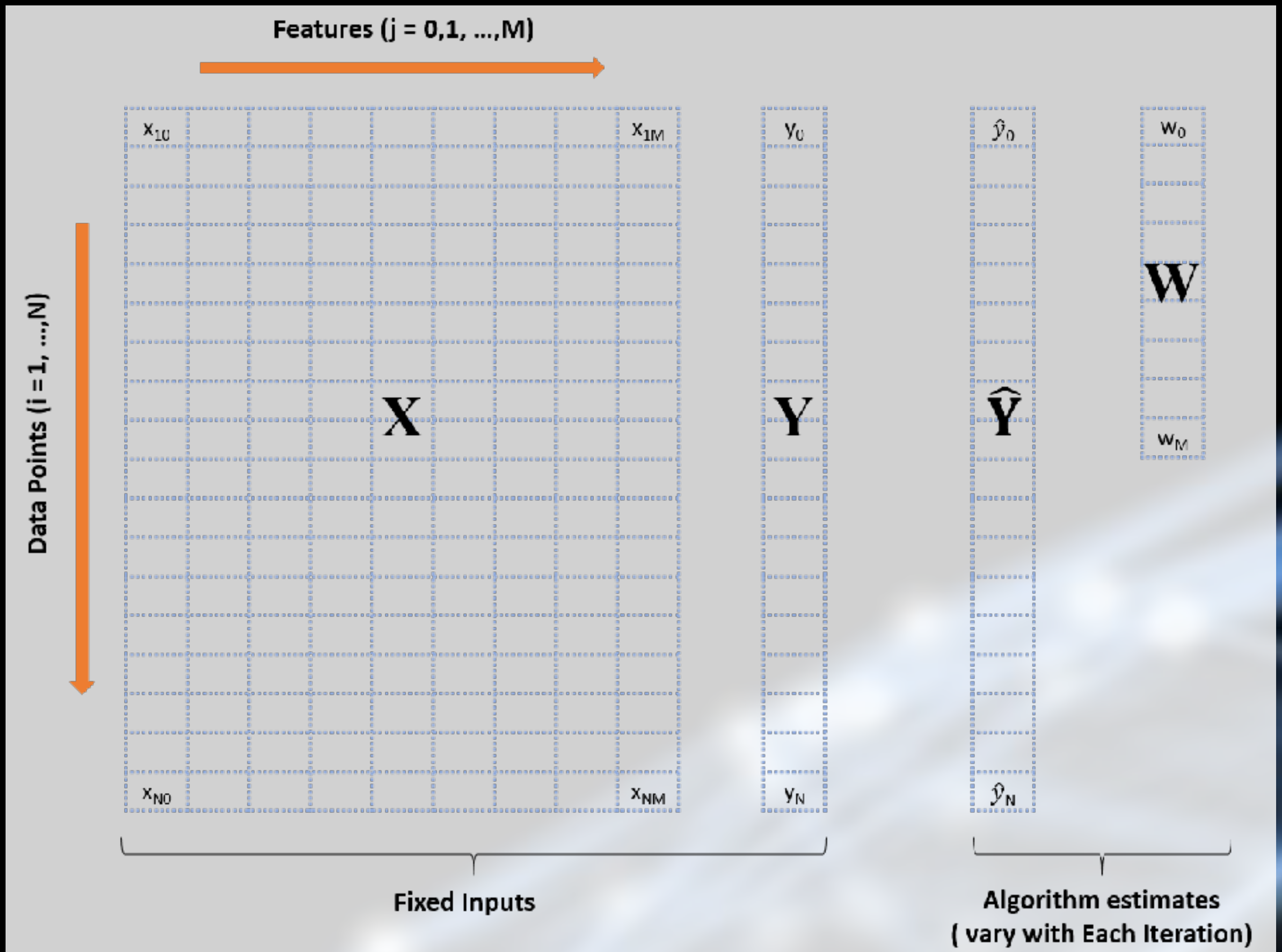
Ridge Regression:
Performs L2 regularization,
i.e. adds penalty equivalent to square of the magnitude of coefficients,
with
minimization objective =
sum of squared residuals + factor * (sum of square of coefficients)

Lasso Regression:
Performs L1 regularization,
i.e. adds penalty equivalent to absolute value of the magnitude of coefficients
with
minimization objective =
ssr + factor * (sum of absolute value of coefficients)

Elastic Net:
Combination

sum of squared residuals = OLS

Regression: General case, general notation (incl. deep nets)



Courtesy: K. Jain

Ridge regression

Minimize $$\sum_{i=1}^{N}\left[y_i - \sum_{j=0}^{M} w_j x_{ij}\right]^2 + \lambda \sum_{j=0}^{M} w_j^2$$

penalty

is equivalent to (proof in lecture)

Minimize $$\sum_{i=1}^{N}\left[y_i - \sum_{j=0}^{M} w_j x_{ij}\right]^2$$ under constraint $$\sum_{j=0}^{M} w_j^2 < const$$

shrinkage property

$$\lambda = 0 \rightarrow const = \infty$$

$$\lambda > 0 \rightarrow const < \infty$$

Ridge is an example of convex optimization
(strictly convex in omegas)

Ridge has a closed-form solution, as derived in the lecture

Least Absolute Shrinkage and Selection Operator (LASSO) regression

penalty

Minimize $$\sum_{i=1}^{N}\left[y_i - \sum_{j=0}^{M} w_j x_{ij}\right]^2 + \lambda \sum_{j=0}^{M} |w_j|$$

is equivalent to (proof in lecture)

Minimize $$\sum_{i=1}^{N}\left[y_i - \sum_{j=0}^{M} w_j x_{ij}\right]^2$$ under constraint $$\sum_{j=0}^{M} |w_j| < const$$

$$\lambda = 0 \rightarrow const = \infty$$

$$\lambda > 0 \rightarrow const < \infty$$

LASSO is an example of non-linear optimization (quadratic programming yields efficiently approximation)

LASSO has no closed-form solution (nonlinearity)

LASSO

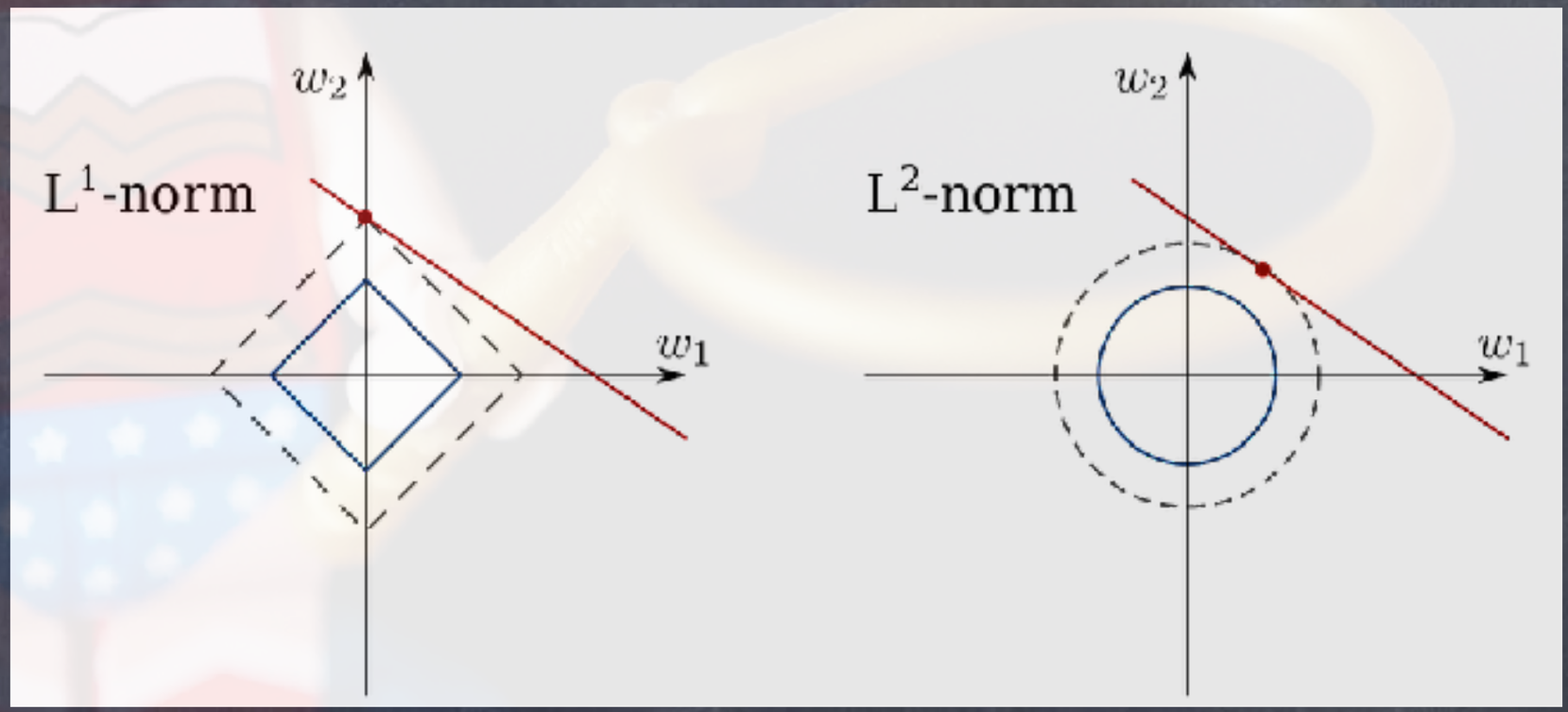
hard thresholding

$|w_1|+|w_2|< const$

Ridge

soft thresholding

$|w_1|^2+|w_2|^2< const$



LASSO can be optimized for
prediction error
or
feature selection
but typically not both!
Some omegas become zero!

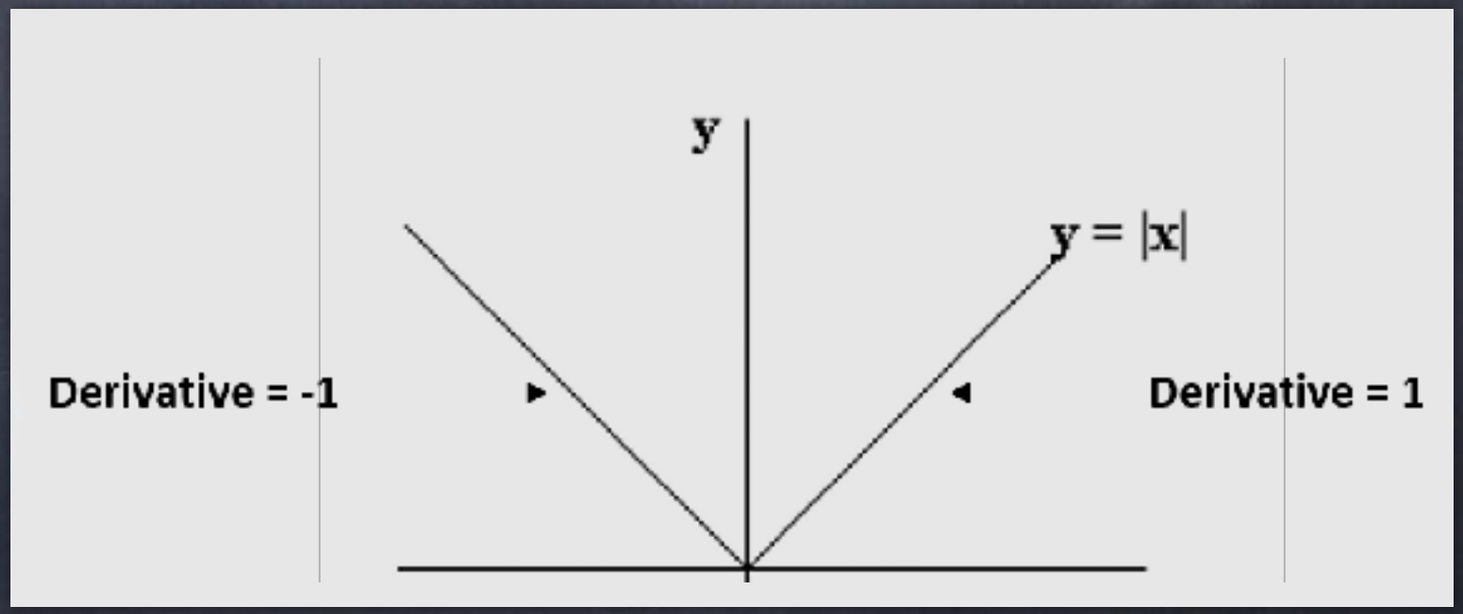
Lambda choice for ridge:
Information criterion or
min. prediction error / CV,
omega never zero

Ridge Gradient Descent

$$w_j^{t+1} = w_j^t - \eta\left[-2\sum_{i=1}^{N} x_{ij}\left\{y_i - \sum_{k=0}^{M} w_k * x_{ik}\right\} + 2\lambda w_j\right]$$

$$w_j^{t+1} = (1 - 2\lambda\eta)w_j^t + 2\eta\sum_{i=1}^{N} x_{ij}\left\{y_i - \sum_{k=0}^{M} w_k * x_{ik}\right\}$$

Lasso Gradient Descent?



Proof idea : const

$$\min_{\omega} \| \underline{y} - \underline{\underline{X}}\,\underline{\omega} \|$$

$$\text{s.t.} \quad \|\omega\|_2^2 \leq c$$

Solution

$$\underline{\omega} \quad \underline{\mathcal{L}(\omega, \lambda)}$$

Lagrange Method

$$\left[\|\omega\|_2^2 - c \right]$$

$$= \| \underline{y} - \underline{\underline{X}} \left(\underline{\underline{X}}^T \underline{\underline{X}} + \lambda \underline{\underline{II}} \right)^{-1} \underline{\underline{X}}^T \underline{y} \|^2$$

$$\frac{\partial g}{\partial \lambda} \overset{!}{=} 0$$

$$\hookrightarrow \frac{\partial g}{\partial \lambda} = \dots\dots = 0 \quad \Rightarrow$$

$$\Theta\left(\| (\underline{\underline{X}}^T \underline{\underline{X}} + \lambda \underline{\underline{\Pi}})^{-1} \underline{\underline{X}}^T \underline{y} \|_2^2 - c\right)$$

$$\frac{\partial f}{\partial \lambda} = \dots = 0 \Rightarrow \boxed{c = \text{const} = \|f(\lambda)\|_2^2}$$

1:1

$$\boxed{\begin{aligned} c &= \text{const} = f(\lambda) \\ \lambda &= h(\text{const.}) \end{aligned}}$$

Observation 1 $rss(\underline{w})$ is convex in $\underline{w}$

$\Leftrightarrow \underline{\underline{X}}^T \underline{\underline{X}}$ full rank

---

Observation 2

$$\frac{\partial}{\partial x}\left(\underline{x}^T \underline{\underline{A}}\, \underline{x}\right) = \underline{\underline{A}}\,\underline{x} + \underline{\underline{A}}^T \underline{x}$$

$$= \left(\underline{\underline{A}} + \underline{\underline{A}}^T\right) \underline{x}$$

Idea $rss(\underline{w}, \lambda) = (\underline{y} - \underline{\underline{X}}\,\underline{w})^T (\underline{y} - \underline{\underline{X}}\,\underline{w}) + \lambda \underline{w}^T \underline{w}$

↳ $\frac{\partial}{\partial w} rss(\underline{w}, \lambda) = \underline{\underline{X}}^T \underline{\underline{X}}\,\underline{w} + 2\lambda \underline{w} - 2\underline{\underline{X}}^T \underline{y} \stackrel{!}{=} 0$

$\Rightarrow 2(\underline{\underline{X}}^T \underline{\underline{X}})\underline{w} + 2\lambda \underline{w} = 2\underline{\underline{X}}^T \underline{y}$ Linear eqn. in $\underline{w}$

$$\Rightarrow \underline{w}^{\text{Ridge}} = \left(\underline{\underline{X}}^T \underline{\underline{X}} + \lambda \underline{\underline{I}}\right)^{-1} \underline{\underline{X}}^T \underline{y}$$

$$\underline{\underline{A}}\,\underline{x} = \underline{b}$$

$$\underline{x} = \underline{\underline{A}}^{-1} \underline{b}$$

Closed form solution of Ridge Problem

Elastic Net

LASSO — many variables useless

$$|w_1|+|w_2|< const$$

ssr + L1 penalty

Ridge — most variables useful

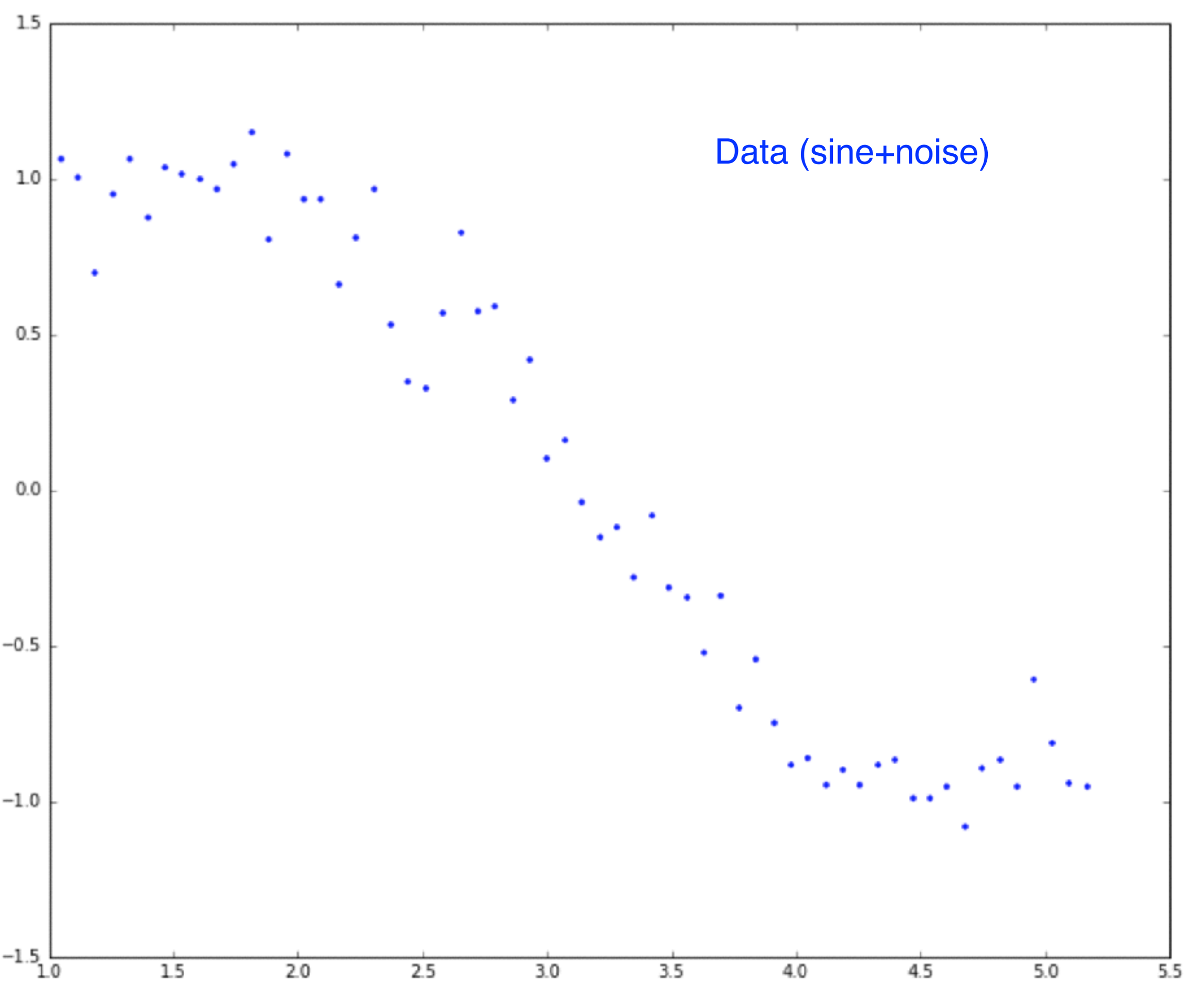
$$|w_1|^2+|w_2|^2< const$$

ssr + L2 penalty

makes Elastic Net

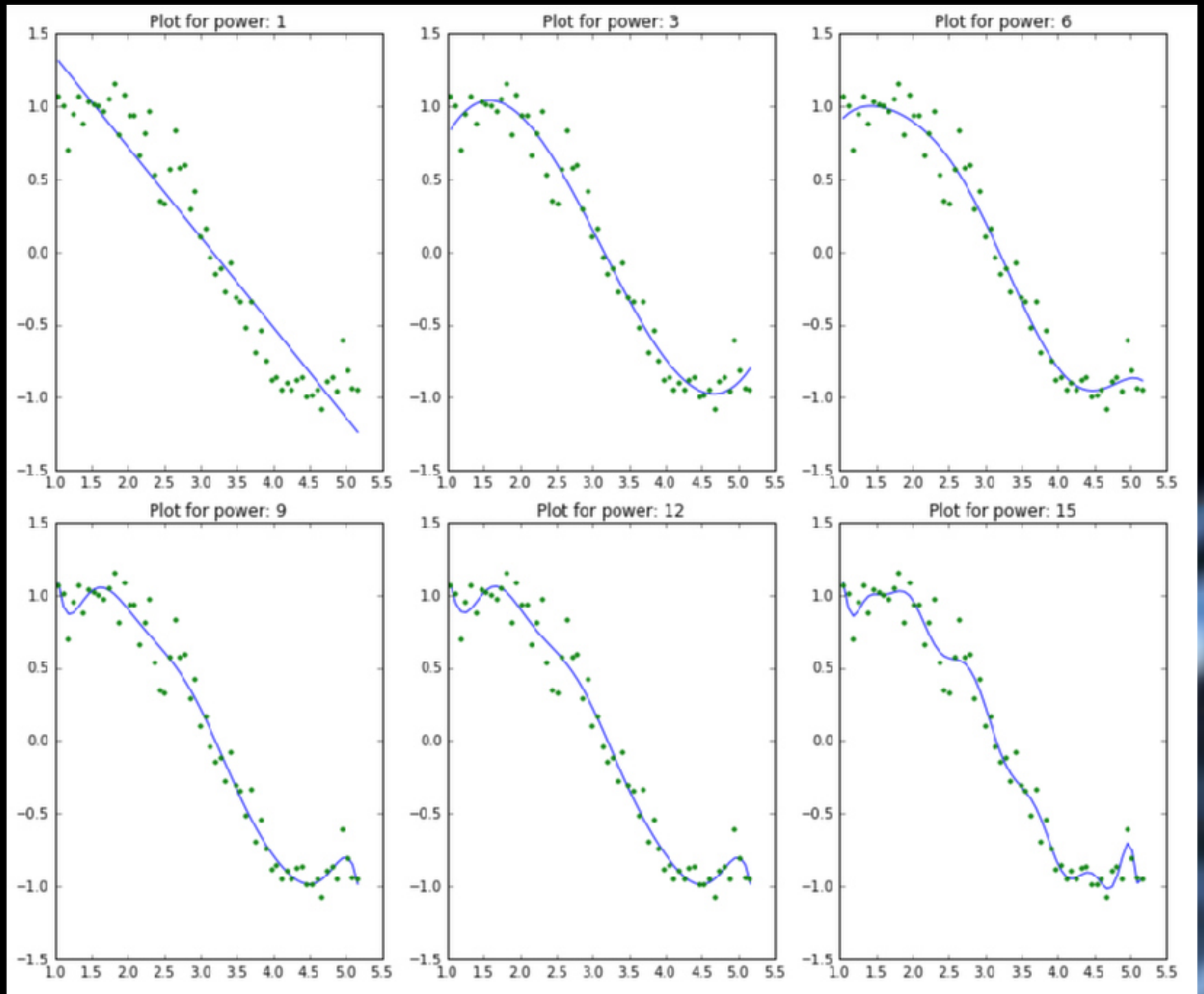
$$\sum_{i=1}^{N}\left[y_i-\sum_{j=0}^{M}w_j x_{ij}\right]^2+\lambda_1\sum_{j=0}^{M}|w_j|+\lambda_2\sum_{j=0}^{M}w_j^2$$

Elastic Net: Good for correlated parameters, combines both strengths, typically (λ_1, λ_2) via cross-validation



Courtesy: K. Jain

Fit: OLS Minimization for polynomials of order 1,3,6,9,12,15



Courtesy: K. Jain

Fit coefficients & Residuals
(OLS Minimization for polynomials of order 1-15)

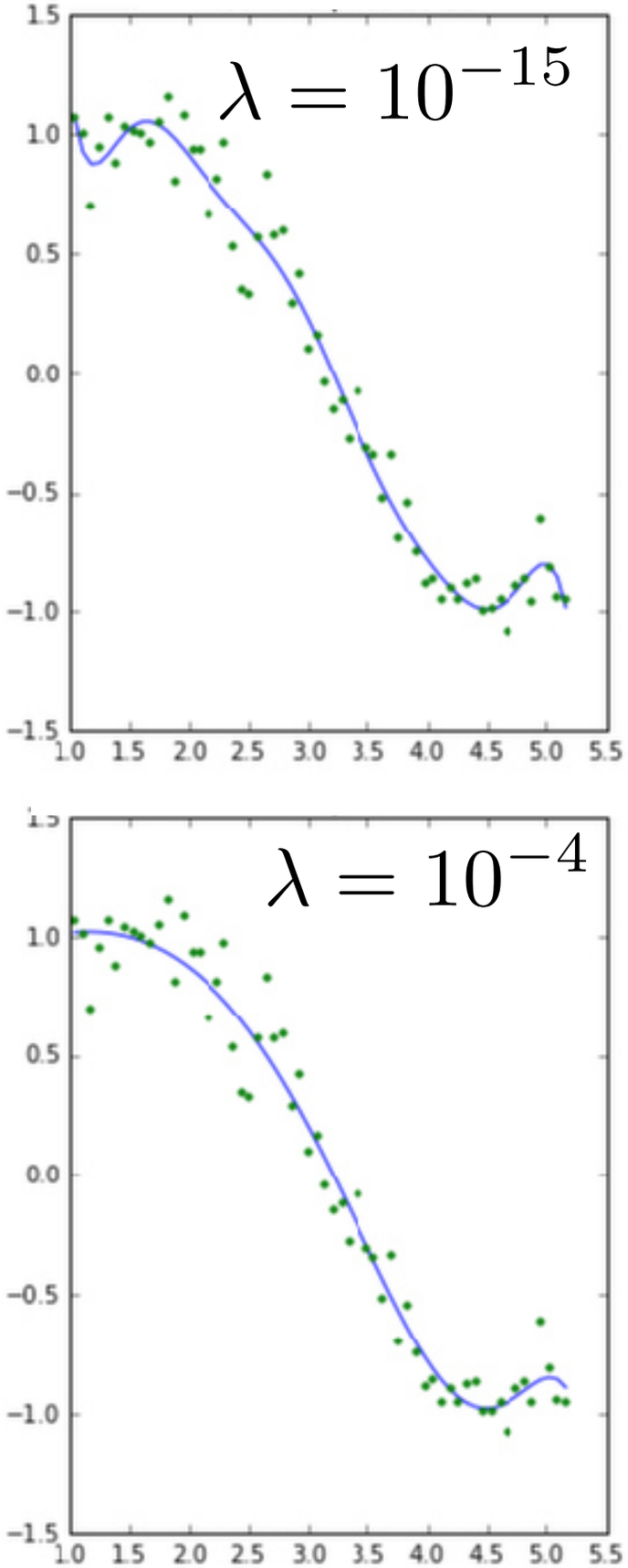
	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_4	1.1	-0.27	1.7	-0.53	-0.036	0.014	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_5	1	3	-5.1	4.7	-1.9	0.33	-0.021	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_6	0.99	-2.8	9.5	-9.7	5.2	-1.6	0.23	-0.014	NaN	NaN	NaN	NaN	NaN
model_pow_7	0.93	19	-56	69	-45	17	-3.5	0.4	-0.019	NaN	NaN	NaN	NaN
model_pow_8	0.92	43	-1.4e+02	1.8e+02	-1.3e+02	58	-15	2.4	-0.21	0.0077	NaN	NaN	NaN
model_pow_9	0.87	1.7e+02	-6.1e+02	9.6e+02	-8.5e+02	4.6e+02	-1.6e+02	37	-5.2	0.42	-0.015	NaN	NaN
model_pow_10	0.87	1.4e+02	-4.9e+02	7.3e+02	-6e+02	2.9e+02	-87	15	-0.81	-0.14	0.026	-0.0013	NaN
model_pow_11	0.87	-75	5.1e+02	-1.3e+03	1.9e+03	-1.6e+03	9.1e+02	-3.5e+02	91	-16	1.8	-0.12	0.0034
model_pow_12	0.87	-3.4e+02	1.9e+03	-4.4e+03	6e+03	-5.2e+03	3.1e+03	-1.3e+03	3.8e+02	-80	12	-1.1	0.062
model_pow_13	0.86	3.2e+03	-1.8e+04	4.5e+04	-6.7e+04	6.6e+04	-4.6e+04	2.3e+04	-8.5e+03	2.3e+03	-4.5e+02	62	-5.7
model_pow_14	0.79	2.4e+04	-1.4e+05	3.8e+05	-6.1e+05	6.6e+05	-5e+05	2.8e+05	-1.2e+05	3.7e+04	-8.5e+03	1.5e+03	-1.8e+02
model_pow_15	0.7	-3.6e+04	2.4e+05	-7.5e+05	1.4e+06	-1.7e+06	1.5e+06	-1e+06	5e+05	-1.9e+05	5.4e+04	-1.2e+04	1.9e+03

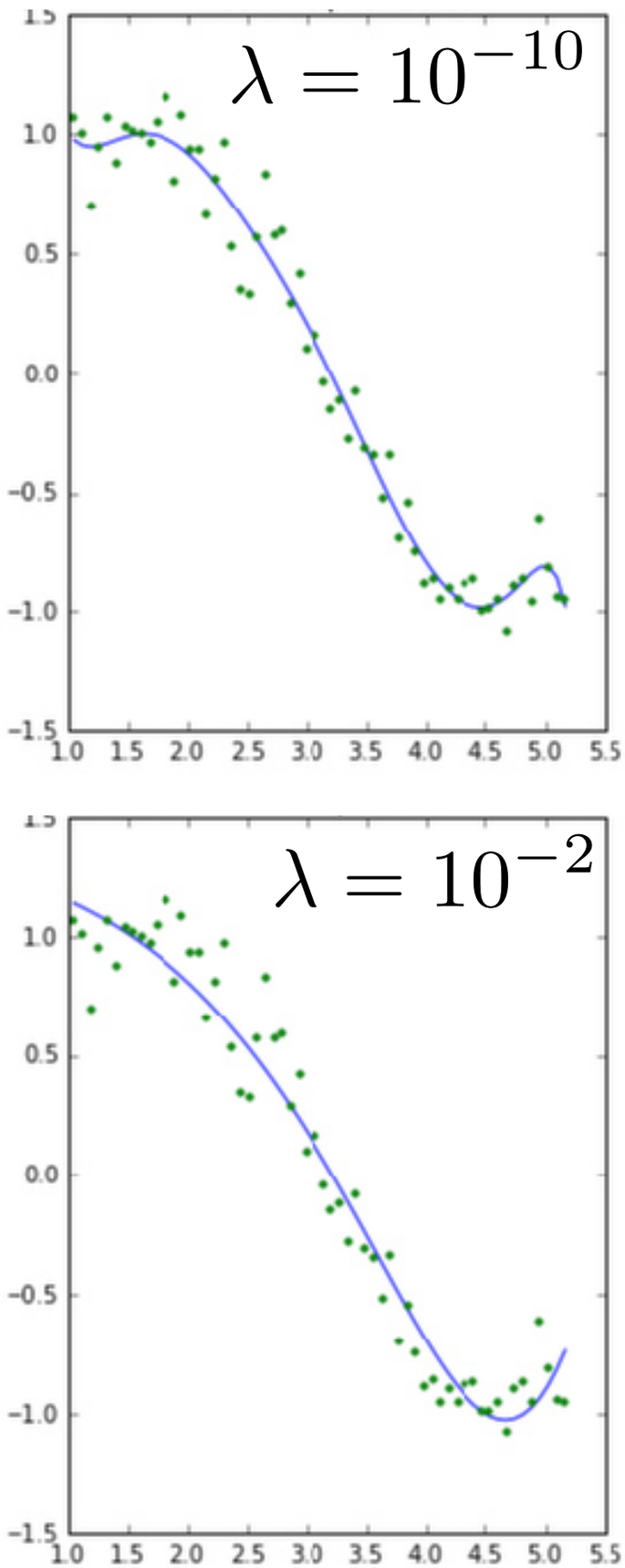
Coefficients increase exponentially with model complexity!

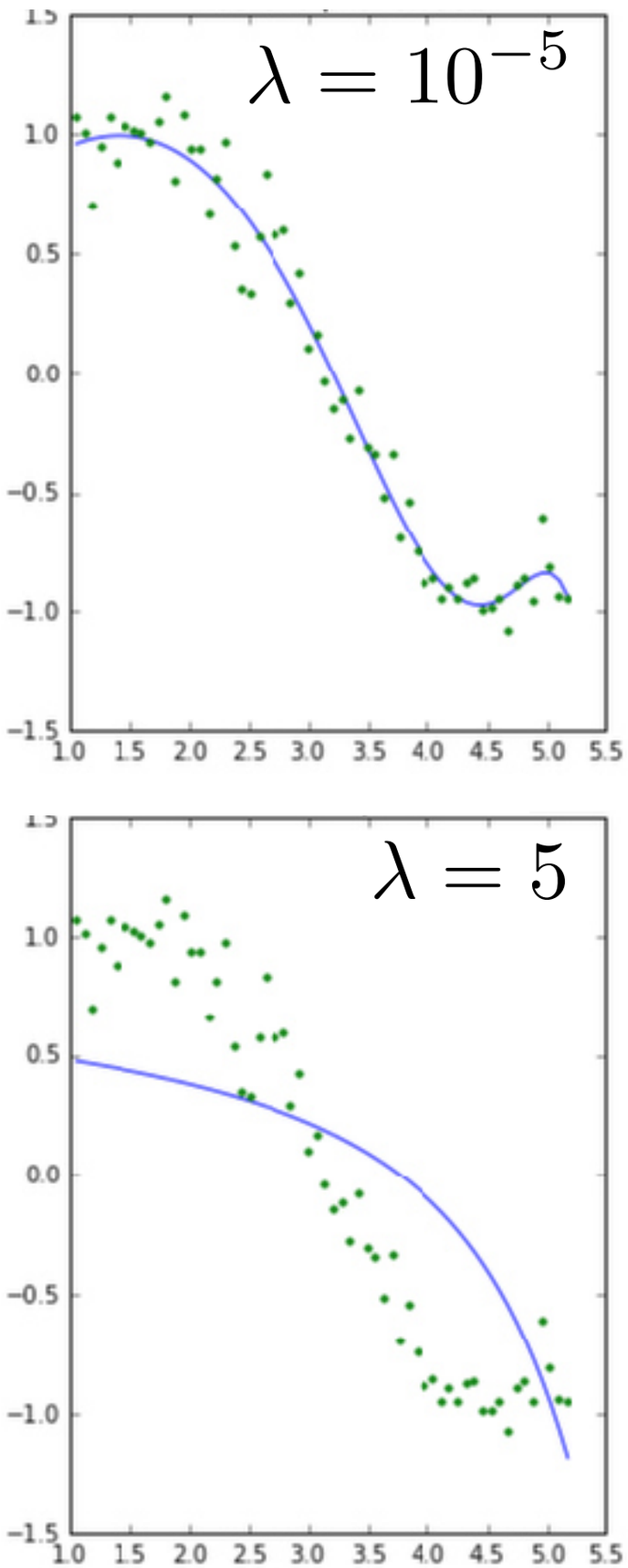
Regularization needed!

Courtesy: K. Jain

Ridge regression (regularization)







Courtesy: K. Jain

Fit coefficients & Residuals

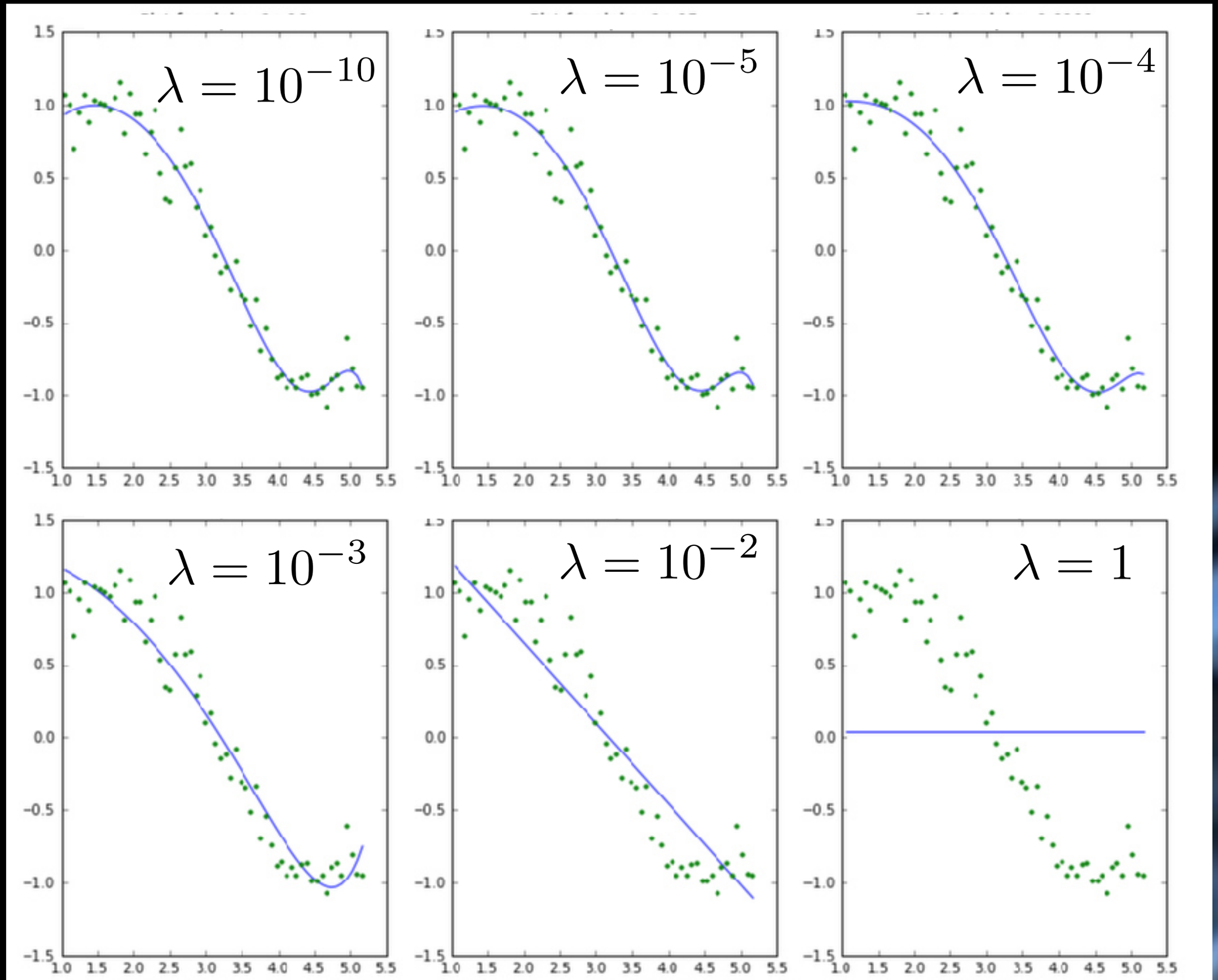
(OLS Minimization with L2 penalty = Ridge regularization)

λ

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	c
alpha_1e-15	0.87	95	-3e+02	3.8e+02	-2.4e+02	66	0.96	-4.8	0.64	0.15	-0.026	-0.0054	0.00086	0.
alpha_1e-10	0.92	11	-29	31	-15	2.9	0.17	-0.091	-0.011	0.002	0.00064	2.4e-05	-2e-05	-4
alpha_1e-08	0.95	1.3	-1.5	1.7	-0.68	0.039	0.016	0.00016	-0.00036	-5.4e-05	-2.9e-07	1.1e-06	1.9e-07	2e
alpha_0.0001	0.96	0.56	0.55	-0.13	-0.026	-0.0028	-0.00011	4.1e-05	1.5e-05	3.7e-06	7.4e-07	1.3e-07	1.9e-08	1.
alpha_0.001	1	0.82	0.31	-0.087	-0.02	-0.0028	-0.00022	1.8e-05	1.2e-05	3.4e-06	7.3e-07	1.3e-07	1.9e-08	1.
alpha_0.01	1.4	1.3	-0.088	-0.052	-0.01	-0.0014	-0.00013	7.2e-07	4.1e-06	1.3e-06	3e-07	5.6e-08	9e-09	1.
alpha_1	5.6	0.97	-0.14	-0.019	-0.003	-0.00047	-7e-05	-9.9e-06	-1.3e-06	-1.4e-07	-9.3e-09	1.3e-09	7.8e-10	2.
alpha_5	14	0.55	-0.059	-0.0085	-0.0014	-0.00024	-4.1e-05	-6.9e-06	-1.1e-06	-1.9e-07	-3.1e-08	-5.1e-09	-8.2e-10	-1
alpha_10	18	0.4	-0.037	-0.0055	-0.00095	-0.00017	-3e-05	-5.2e-06	-9.2e-07	-1.6e-07	-2.9e-08	-5.1e-09	-9.1e-10	-1
alpha_20	23	0.28	-0.022	-0.0034	-0.0006	-0.00011	-2e-05	-3.6e-06	-6.6e-07	-1.2e-07	-2.2e-08	-4e-09	-7.5e-10	-1

Courtesy: K. Jain

LASSO regression (regularization)



Courtesy: K. Jain

Fit coefficients & Residuals

(OLS Minimization with L1 penalty = LASSO regularization)

λ

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	coe
alpha_1e-15	0.96	0.22	1.1	-0.37	0.00089	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-10	0.96	0.22	1.1	-0.37	0.00088	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-08	0.96	0.22	1.1	-0.37	0.00077	0.0016	-0.00011	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.3
alpha_1e-05	0.96	0.5	0.6	-0.13	-0.038	-0	0	0	0	7.7e-06	1e-06	7.7e-08	0	0
alpha_0.0001	1	0.9	0.17	-0	-0.048	-0	-0	0	0	9.5e-06	5.1e-07	0	0	0
alpha_0.001	1.7	1.3	-0	-0.13	-0	-0	-0	0	0	0	0	0	1.5e-08	7.5
alpha_0.01	3.6	1.8	-0.55	-0.00056	-0	-0	-0	-0	-0	-0	-0	0	0	0
alpha_1	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_5	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_10	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0

HIGH SPARSITY

Courtesy: K. Jain

Machine Learning II
Week #4

Jan Nagler

Deep Dynamics Group
Centre for Human and Machine Intelligence (HMI)
Frankfurt School of Finance & Management

(Details in lecture!)

Frankfurt, October 2019

Outline for today

Final comment to mentioned book in week #3

Solution & Presentation for Assignments #4

MCMC: Theory and Implementation for
Markov Chains, and Metropolis Hastings

Solution & Presentation for Assignments #3
in lecture (10 min)

—Award ceremony—
(1330-1345)

Hints for final exam

SVM and Kernels

Hyperparameters

New and Views in Machine Learning

Assignment 5: Kernel methods

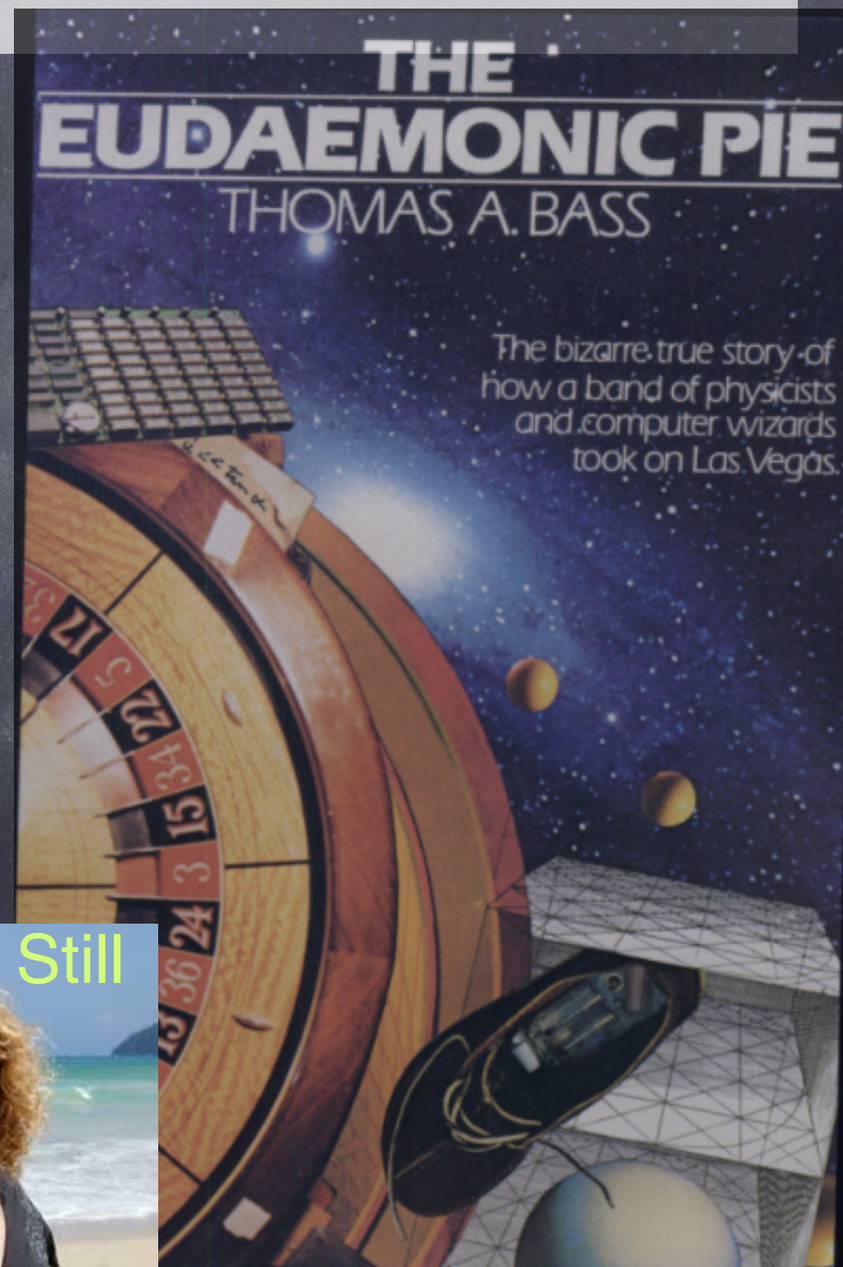
Final comment to mentioned book in week #3

The Eudaemonic Pie: The Bizarre True Story of How a Band of Physicists and Computer Wizards Took on Las Vegas

Audible Audiobook – Unabridged

Thomas A. Bass (Author, Publisher), Stephen Tupper (Narrator)

★★★★☆ 18 customer reviews



Metropolis-Hastings sampling

Sampling algorithm for distributions from which it is difficult to sample directly
(due to intractable (conditional pdfs) integrals, normalization constants,
high computational costs for high dimensional distributions)

The sequence (called Monte Carlo chain) can be used to approximate
the distribution (e.g. from the histogram) or to compute integrals
(e.g. an expected value).

The core of the algorithm lies in a distribution that is an
acceptance probability
for the next proposed candidate of the Markov Chain

$$\alpha = A(x \to x') = \min\left(1, \frac{P(x')Q(x' \to x)}{P(x)Q(x \to x')}\right)$$

$P(x)$ Sample distribution (must not be normalized)

x is current state ———- next state is x'

$Q(x \to x')$ is suggested transition probability

Metropolis-Hastings sampling

Transition probability = Suggested transition Q

$$T(x \to x') = Q(x \to x')$$

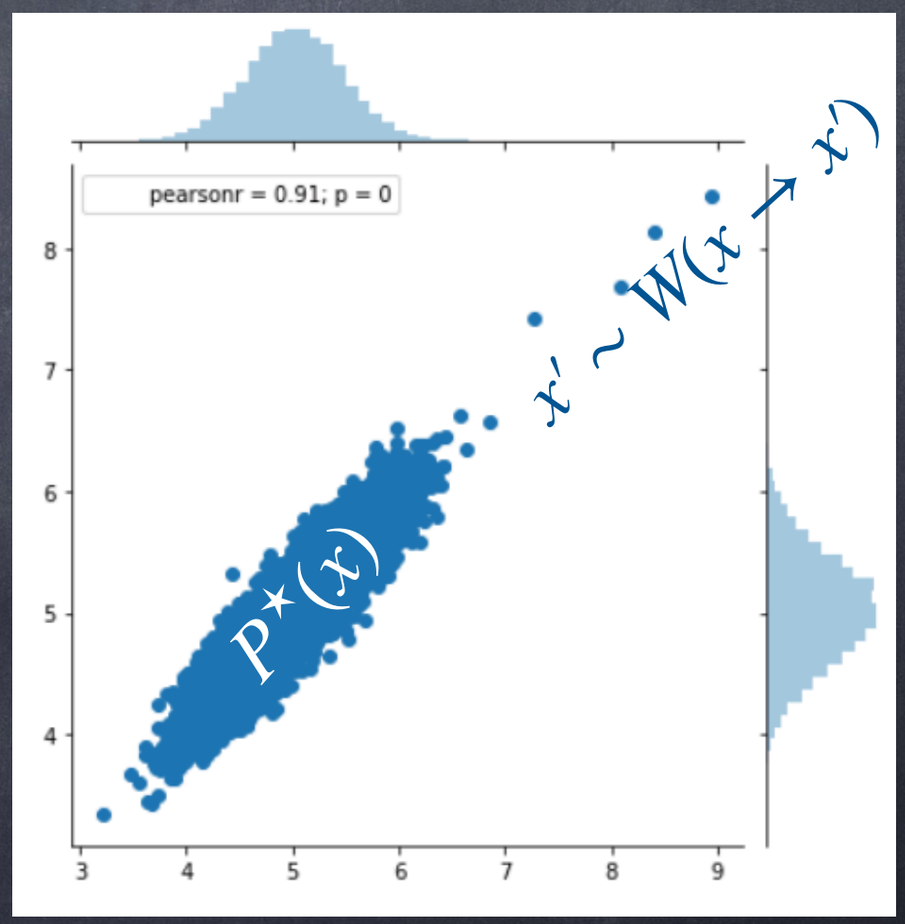
Probability (Wahrscheinlichkeit) of Markov Chain

$$W(x \to x') = Q(x \to x')A(x \to x')$$

Gibbs: W=Q

Target distribution or equilibrium- or stationary distribution

$$P^\star(x)$$



Lecture: Theory and implementation in Python

Markov chains

General

Normalization of Markov Chain

$$\sum_{x'} W(x \to x') = 1$$

Ergodicity of Markov Chain $W(x \to x') > 0$

(any x' must be reached from any other x with non-zero probability in a finite number of steps)

Special

Homogeneity of Markov Chain

$$\sum_{x'} P^\star(x') W(x' \to x) = P^\star(x)$$

A reversible Markov Chain (special property) exhibits

$$Q(x \to x') = Q(x' \to x)$$

Markov chains

Master equation of Markov Chain

$$\frac{dP(x,t)}{dt} = \sum_{x'} P(x')W(x' \to x) - \sum_{x'} P(x)W(x \to x')$$

Stationary distribution of Markov Chain

$P^\star(x)$ is solution of master equation for

$$\frac{dP^\star(x,t)}{dt} = 0 \to P^\star(x)$$

Detailed balance of Markov Chain

$$\to \sum_{x'} P^\star(x')W(x' \to x) = \sum_{x'} P^\star(x)W(x \to x')$$

$$= P^\star(x) \sum_{x'} W(x \to x') = P^\star(x)$$

Markov chains

Detailed balance of Markov Chain

$$\rightarrow \sum_{x'} P^{\star}(x')W(x' \rightarrow x) = \sum_{x'} P^{\star}(x)W(x \rightarrow x')$$

In MCMC detailed balance is enforced by the choosing W such that

$$P(x')W(x' \rightarrow x) = P(x)W(x \rightarrow x')$$

This justifies the Metropolis-Hastings acceptance probability (proof in lecture)

$$\alpha = A(x \rightarrow x') = \min\left(1, \frac{P(x')Q(x' \rightarrow x)}{P(x)Q(x \rightarrow x')}\right)$$

given $W(x \rightarrow x') = Q(x \rightarrow x')A(x \rightarrow x')$

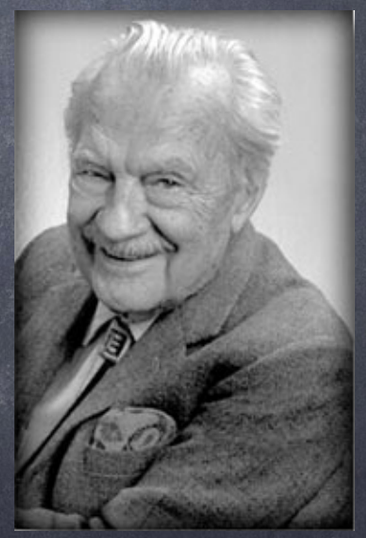
Metropolis and Metropolis-Hastings

Metropolis-Hastings acceptance probability

$$\alpha = A(x \to x') = \min\left(1, \frac{P(x')Q(x' \to x)}{P(x)Q(x \to x')}\right)$$

Metropolis acceptance probability for symmetric Q

$$\alpha = A(x \to x') = \min\left(1, \frac{P(x')}{P(x)}\right)$$



Historically: First Metropolis then Hastings generalised

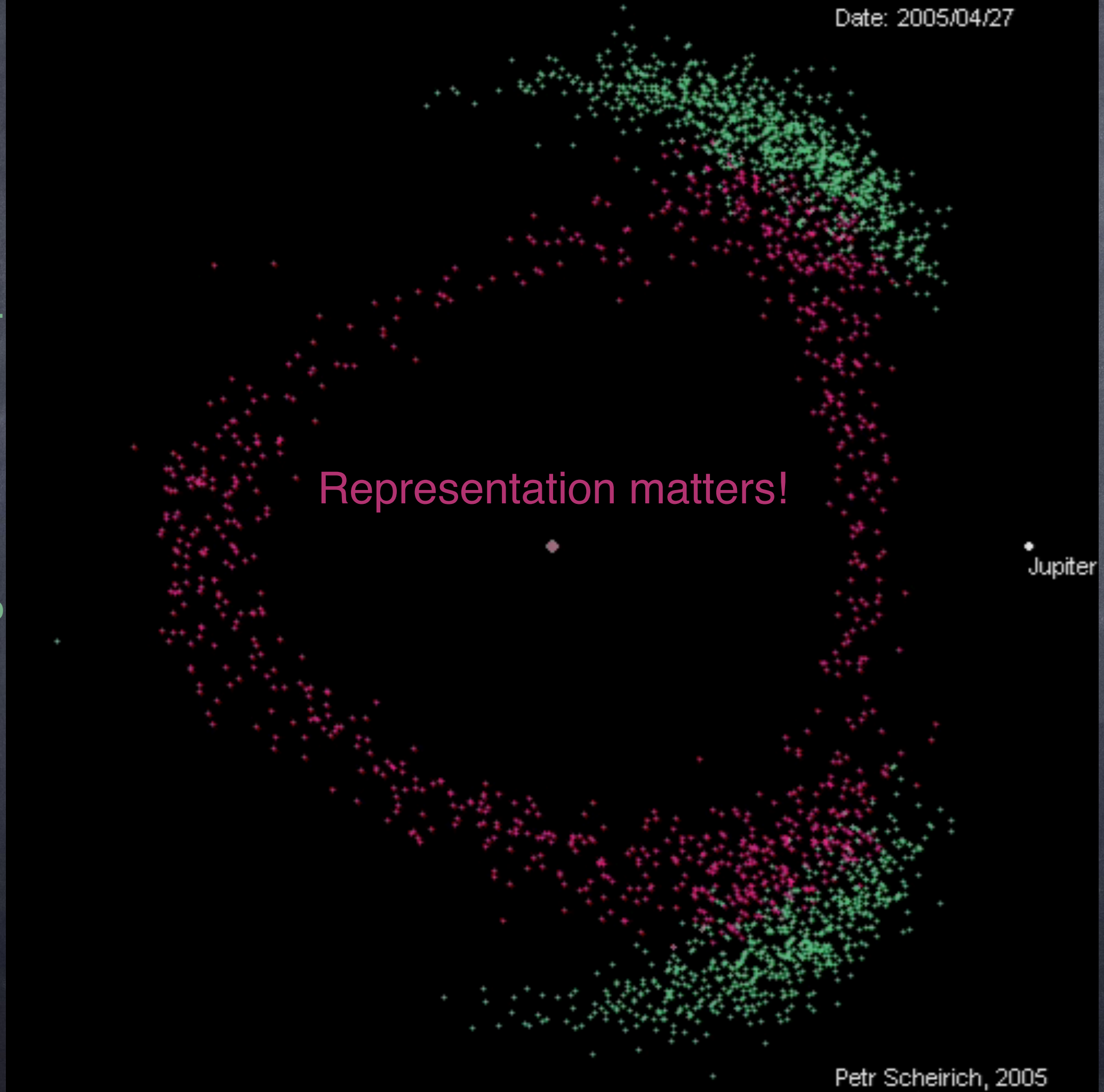
Kernel Methods

Representation matters

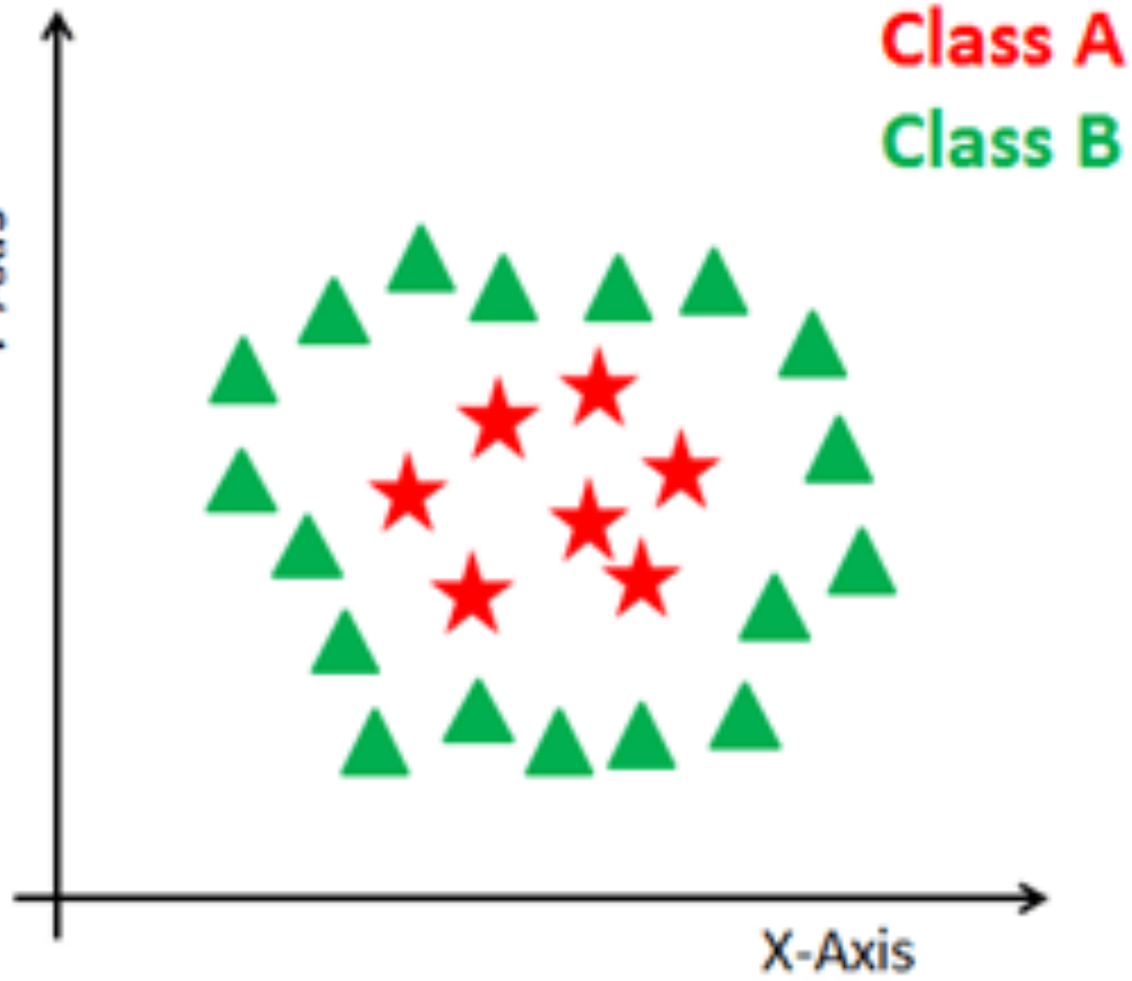
Heliocentrism

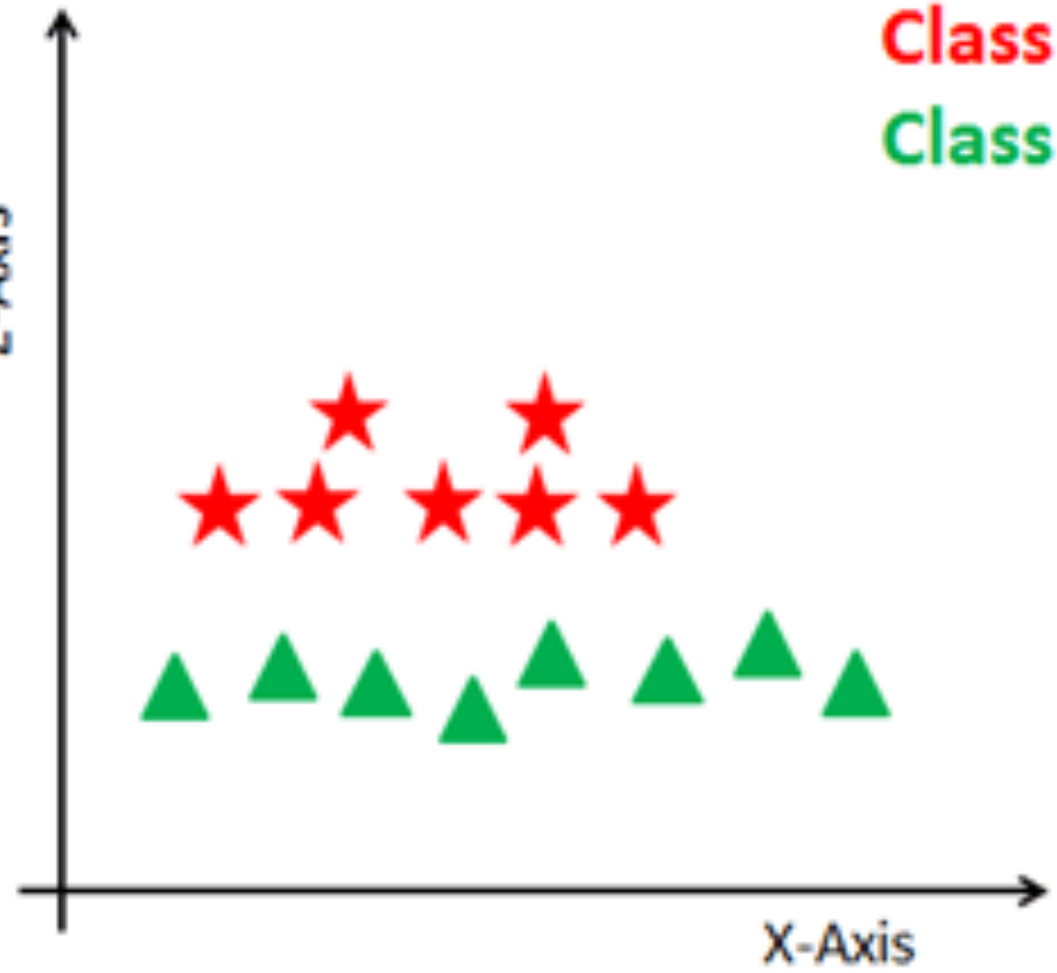
Geocentrism

Corotating frame Sun-Jupiter



Representation matters: Kernel Methods

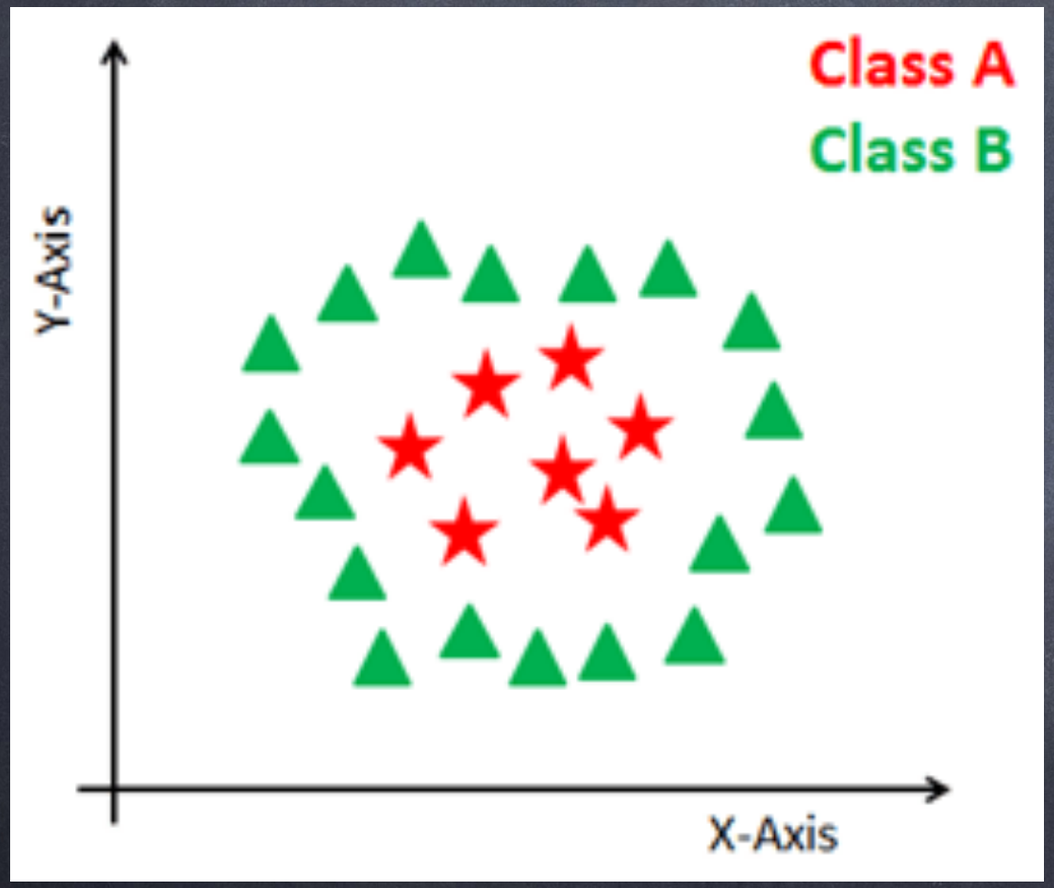




$$(x, y) \rightarrow (r, \varphi) \qquad x = r\cos(\varphi) \qquad y = r\sin(\varphi)$$

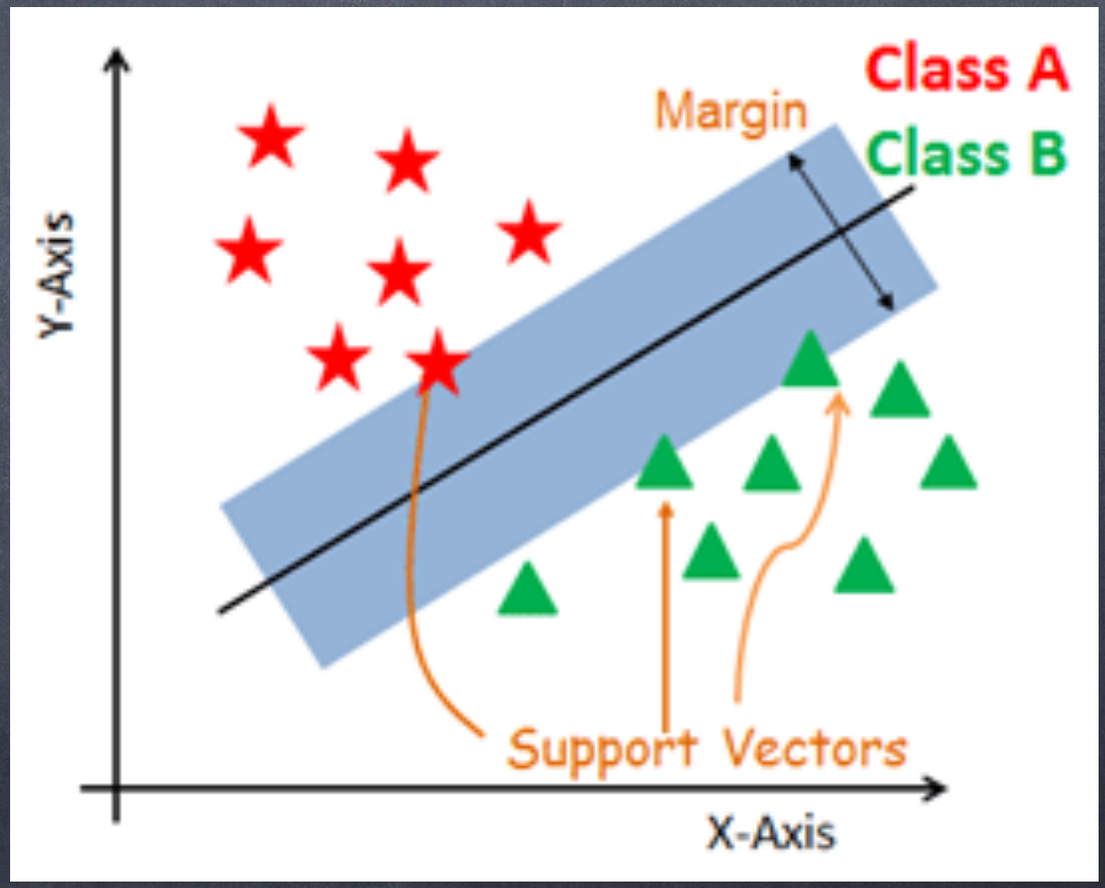
Nonlinear kernel Support Vector Machines

Objective: Map such that sample data are divided
by a gap, also called margin that is as wide as possible



Linearly inseparable

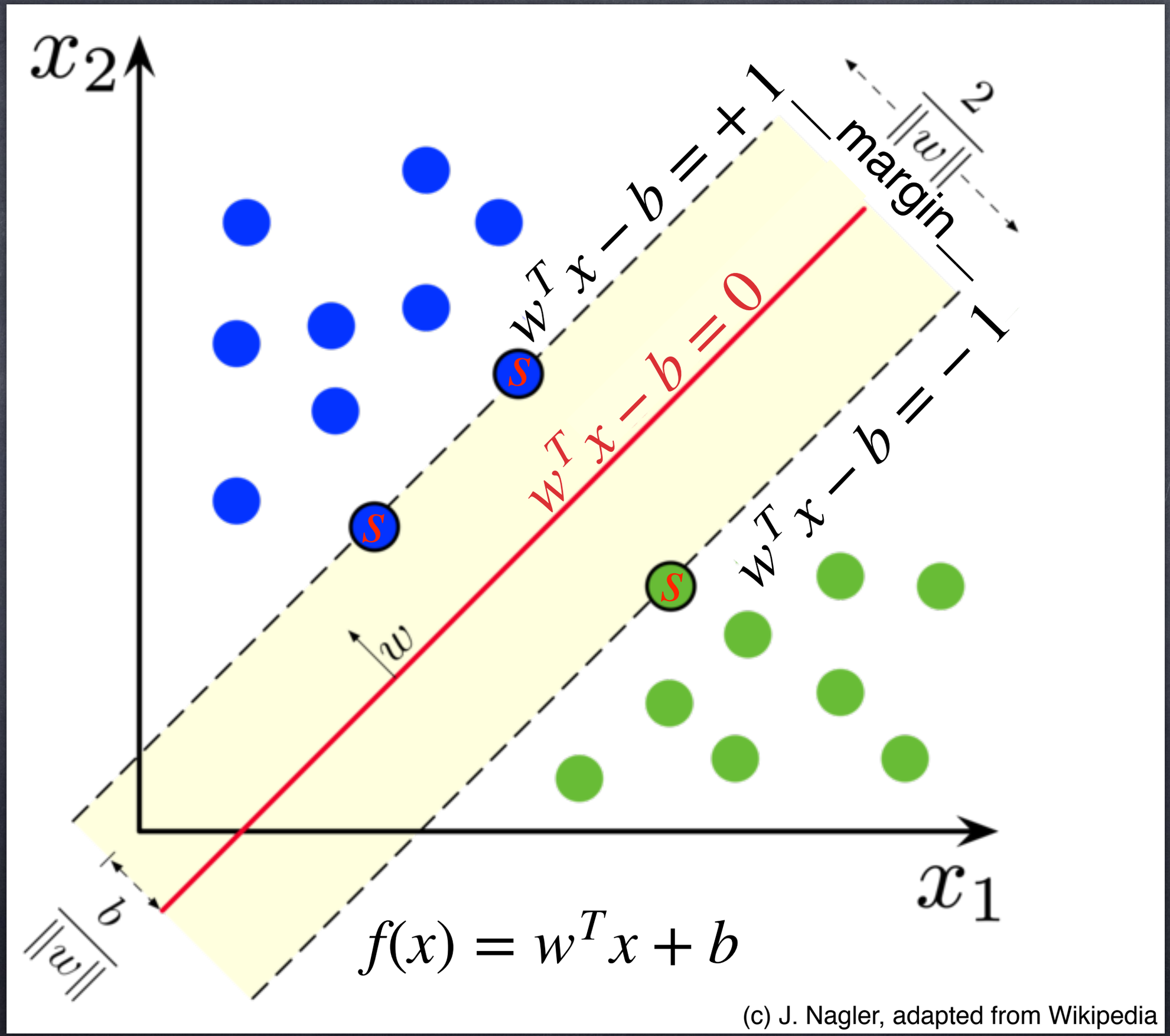
Map



Linearly separable

Support vectors are the data points which are closest to the hyperplane

Linear SVM Classifier method for binary classes -1 and +1



(c) J. Nagler, adapted from Wikipedia

Linear classifier $f(x) = w^T x + b$

x : samples

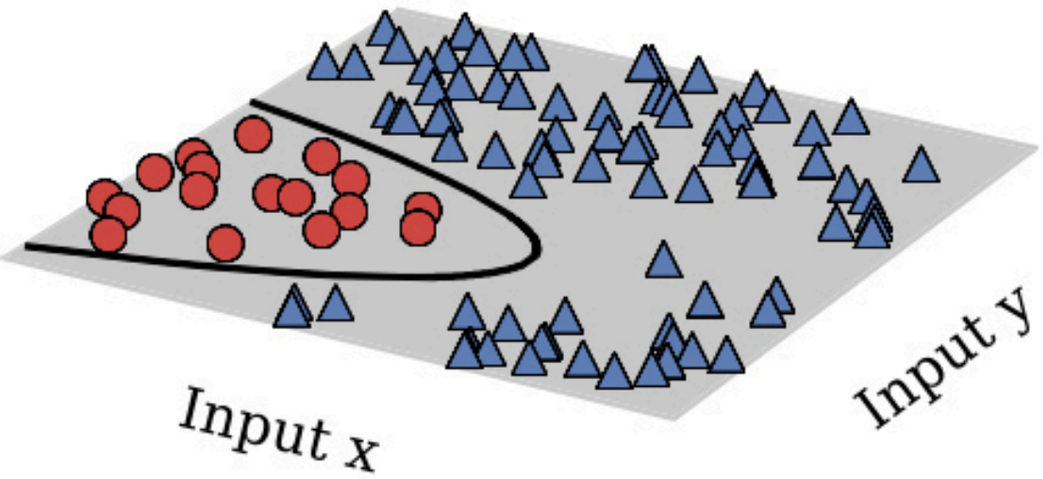
w, b : from support vectors

Feature map $\phi : \mathbb{R}^n \to \mathbb{R}^m \qquad \phi : x \to \phi(x)$

Nonlinear classifier $f(x) = w^T \phi(x) + b$

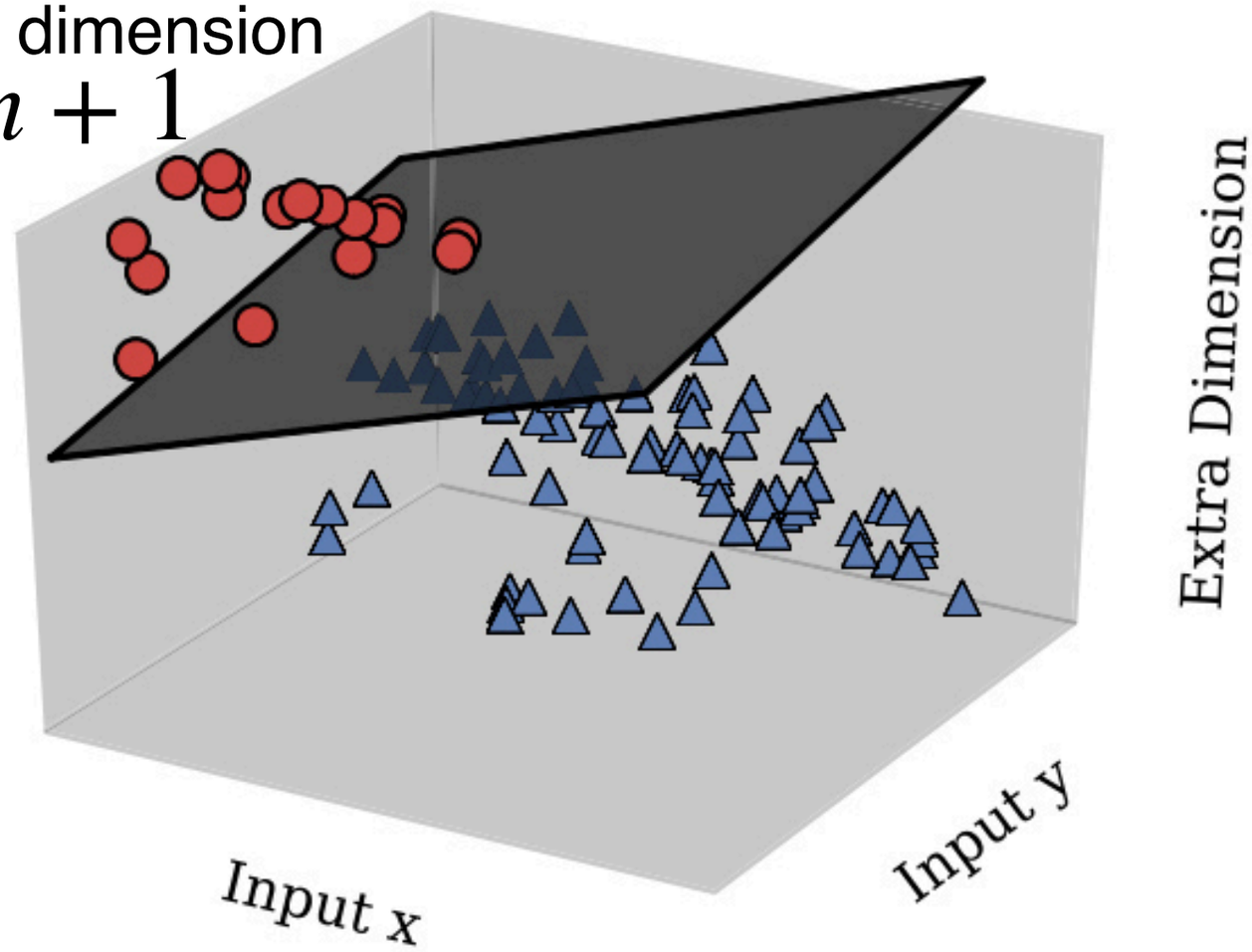
No extra dimension
but nonlinear feature map

$m = n$



With extra dimension

$m = n + 1$



Ntampaka et al., A MACHINE LEARNING APPROACH FOR DYNAMICAL MASS MEASUREMENTS OF GALAXY CLUSTERS, Astrophysical Journal 803 (2015)

Linear classifier

$$f(x) = w^T x + b$$

Nonlinear classifier

$$f(x) = w^T \phi(x) + b$$

Decision function $sign[f(x)]$

Learning objective (w/ and w/o kernels)

$$\min_{w \in \mathbb{R}^n} \left[||w||^2 + C \sum_{i \leq N} \max[0, 1 - y_i f(x_i)] \right]$$

x_i support vectors

$y_i \in \{-1, +1\}$

C: Soft margin parameter

Dual space problem

$$f(x) = \sum_i \lambda_i y_i x_i^T x + b$$

$$f(x) = b + \sum_i \lambda_i y_i \phi(x_i^T) \phi(x)$$

Kernel

with dual learning objective (cumbersome)

Kernel “use cases”

Linear kernel $k(x, x') = x^T x'$

Polynomial kernel $k(x, x') = (1 + \gamma x^T x')^d$

Gaussian kernel, or radial base function (rbf)

$$k(x, x') = e^{-||x-x'||^2/(2\sigma^2)} = \exp\left(-\gamma||x - x'||^2\right)$$

$\gamma = 1/(2\sigma^2)$

The feature space of the rbf kernel is infinite dimensional as

$$e^{-\gamma||x-x'||^2} = \sum_k^{\infty} \frac{(x^T x')^k}{k!} \exp\left(-\gamma||x||^2\right) \exp\left(-\gamma||x'||^2\right)$$

for $\gamma = 1/2$

Radial base function (rbf) in *dual space representation*

$$f(x) = \sum_{i \leq N} \lambda_i y_i \exp(-\frac{||x - x_i||^2}{2\sigma^2}) + b$$

Kernel Trick

Quadratic 2d->3d kernel

$$\phi : (x_1, x_2) \rightarrow (x_1^2, x_2^2, \sqrt{2}x_1x_2), \quad \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

Kernel trick

$$\phi(x)^T\phi(x') = (x_1^2, x_2^2, \sqrt{2}x_1x_2)^T({x'_1}^2, {x'_2}^2, \sqrt{2}x'_1x'_2) = (x^Tx')^2$$

Classifier can be learnt and applied
without computational expensive kernel evaluations,
c.f., quadratic programming + other advantages

Further literature to the Kernel Trick

Mercer's Theorem applied

Computation relies on a computational fast representation (theorem)

Theorem. Suppose K is a continuous symmetric non-negative definite kernel. Then there is an orthonormal basis $\{e_i\}_i$ of $L^2[a, b]$ consisting of eigenfunctions of T_K such that the corresponding sequence of eigenvalues $\{\lambda_j\}_j$ is nonnegative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $[a, b]$ and K has the representation

$$K(s,t) = \sum_{j=1}^{\infty} \lambda_j \, e_j(s) \, e_j(t)$$

where the convergence is absolute and uniform.

Valid kernels K(,) are defined via

K is said to be *non-negative definite* (or positive semidefinite) if and only if

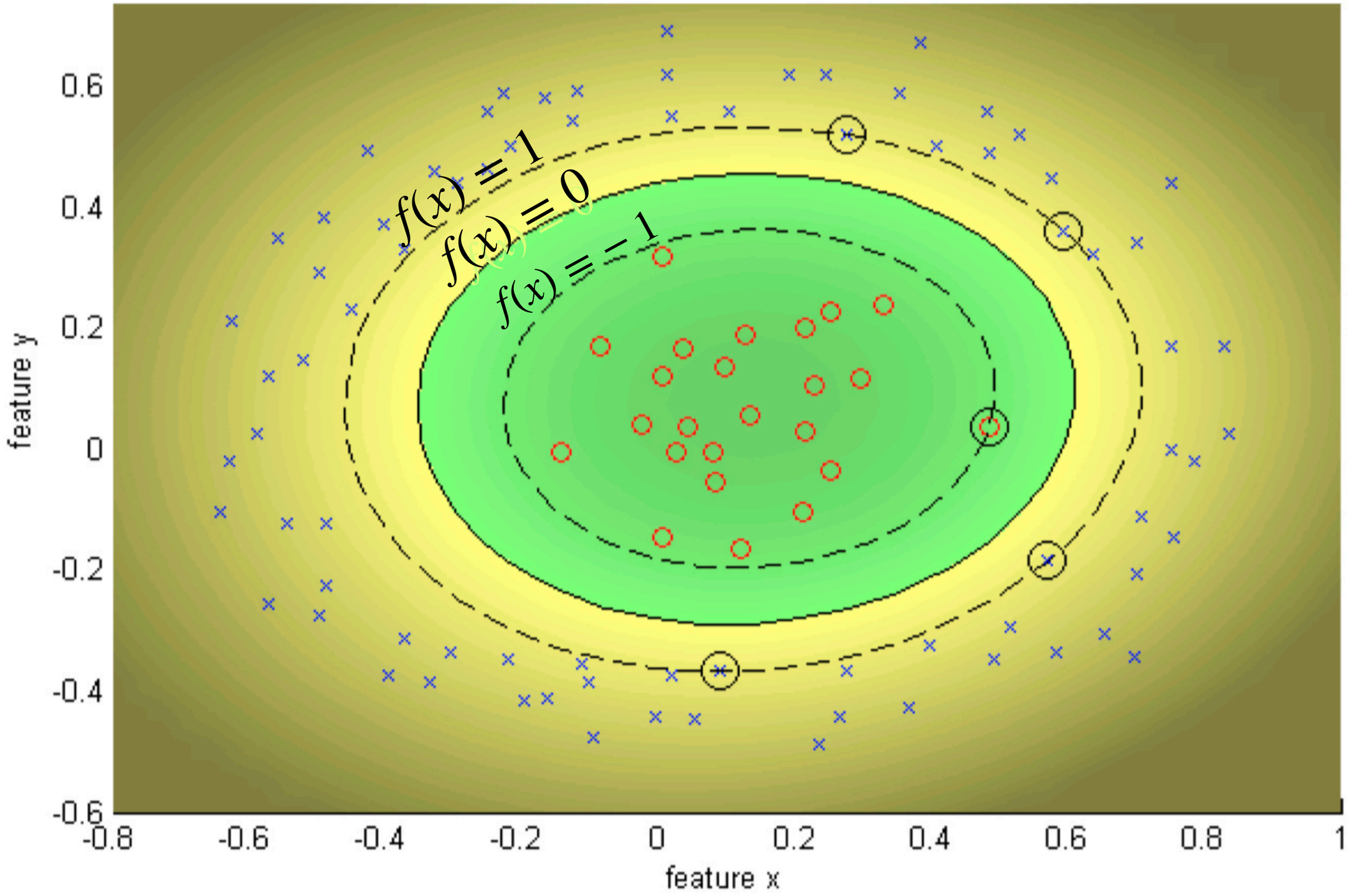
$$\sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j) c_i c_j \geq 0$$

for all finite sequences of points x_1, ..., x_n of $[a, b]$ and all choices of real numbers c_1, ..., c_n (cf. positive-definite kernel).

Wikipedia

$$\sigma = 1.0 \quad C = \infty$$

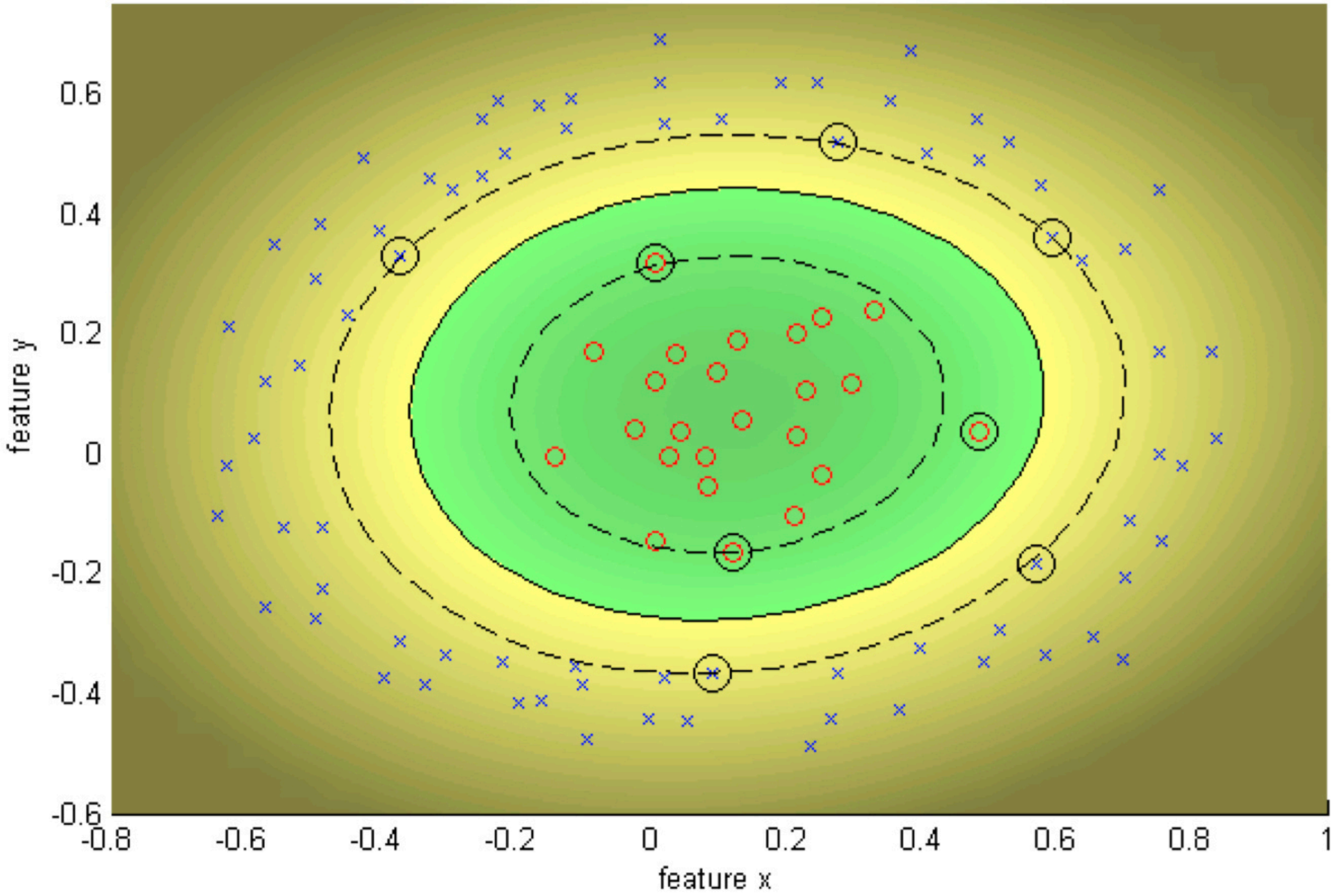
nonlinear decision boundaries



$\sigma = 1.0$ $C = 100$

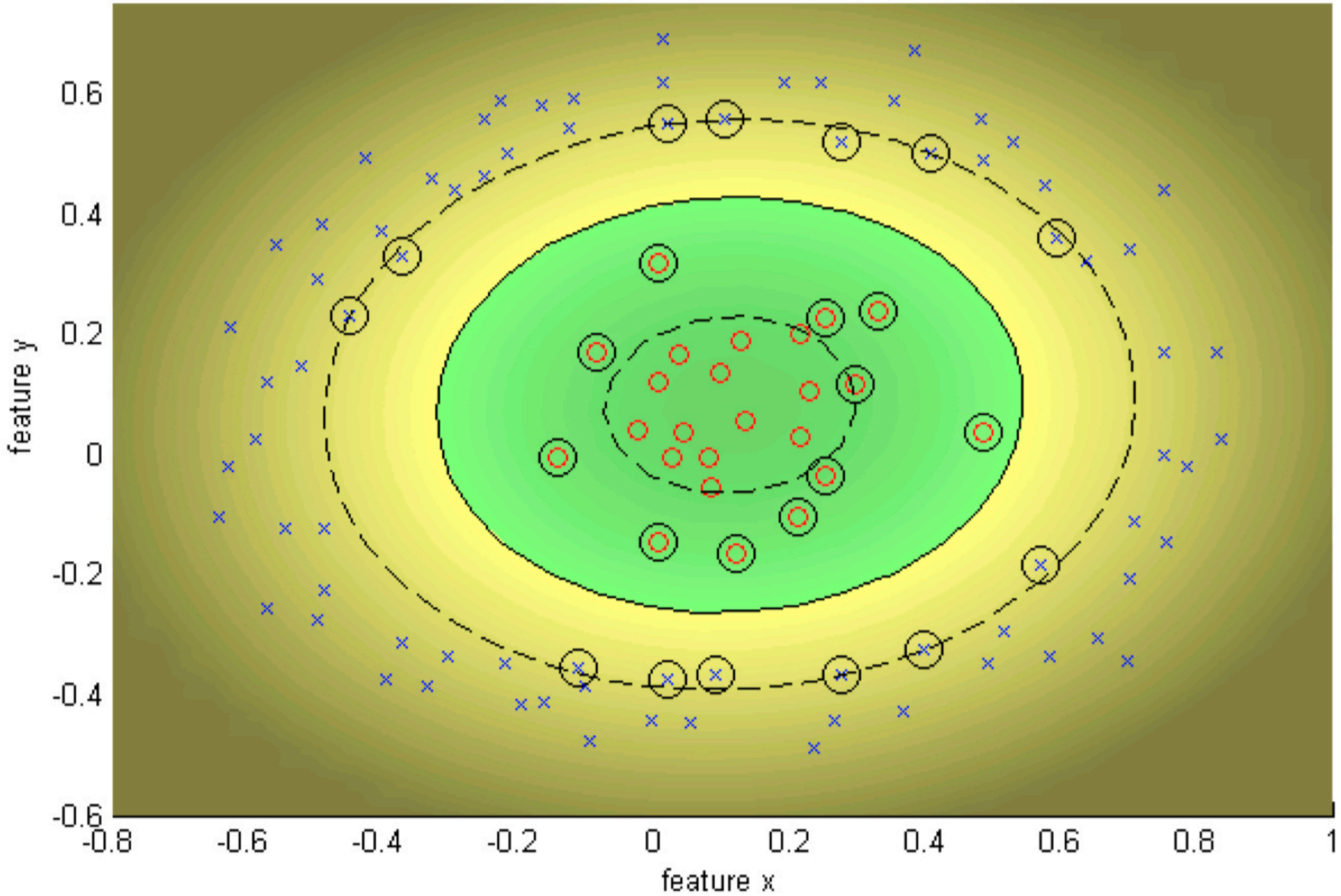
C: Soft margin parameter

ok



$\sigma = 1.0 \quad C = 10$

too loose



Kernel hyperparameters

Gamma: A too low value of gamma will under-fit the training dataset, whereas a too high value of gamma will cause overfitting

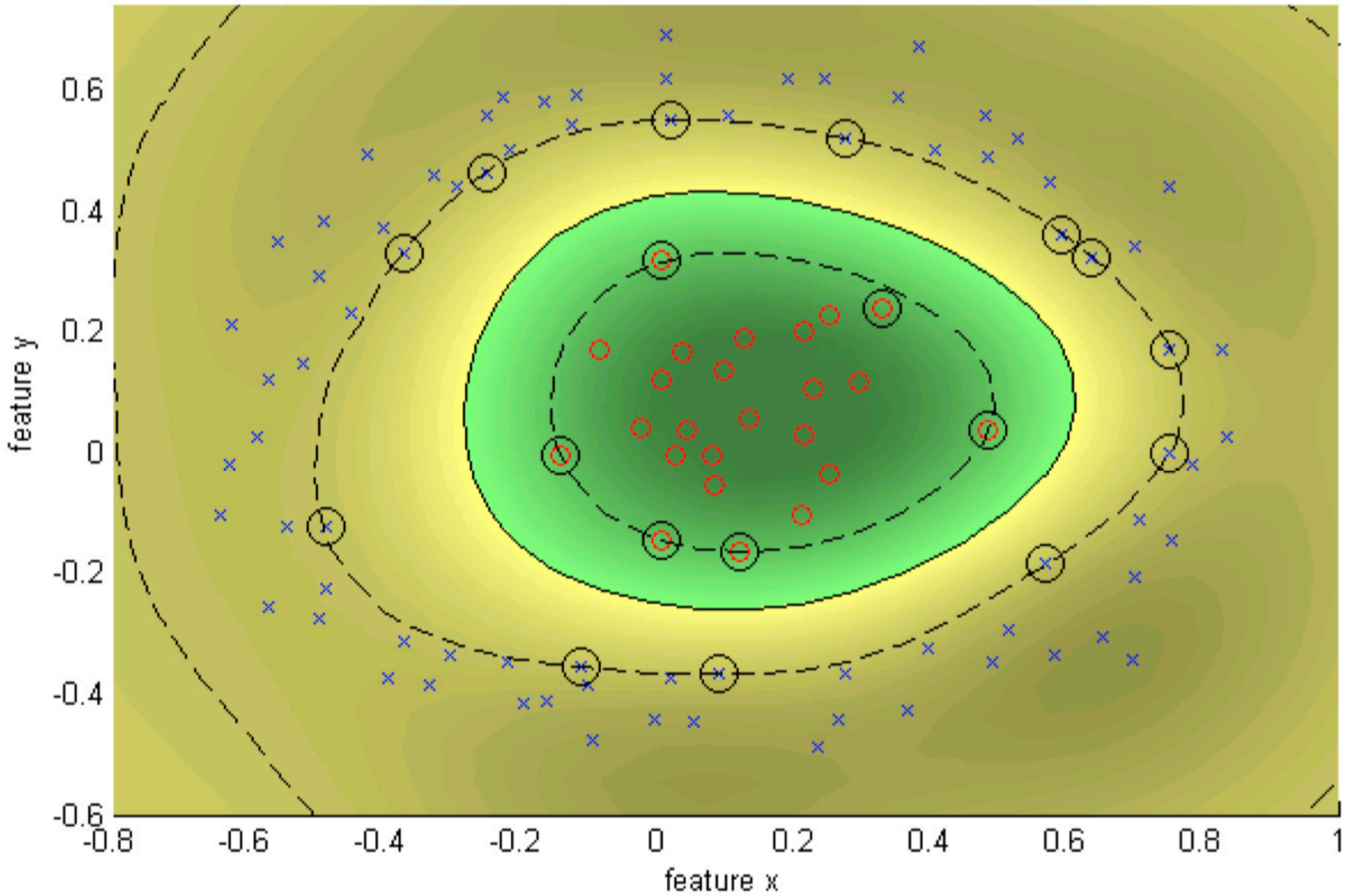
$$\gamma = 1/(2\sigma^2)$$

Omega: Vice versa

C: Soft margin parameter (as shown)

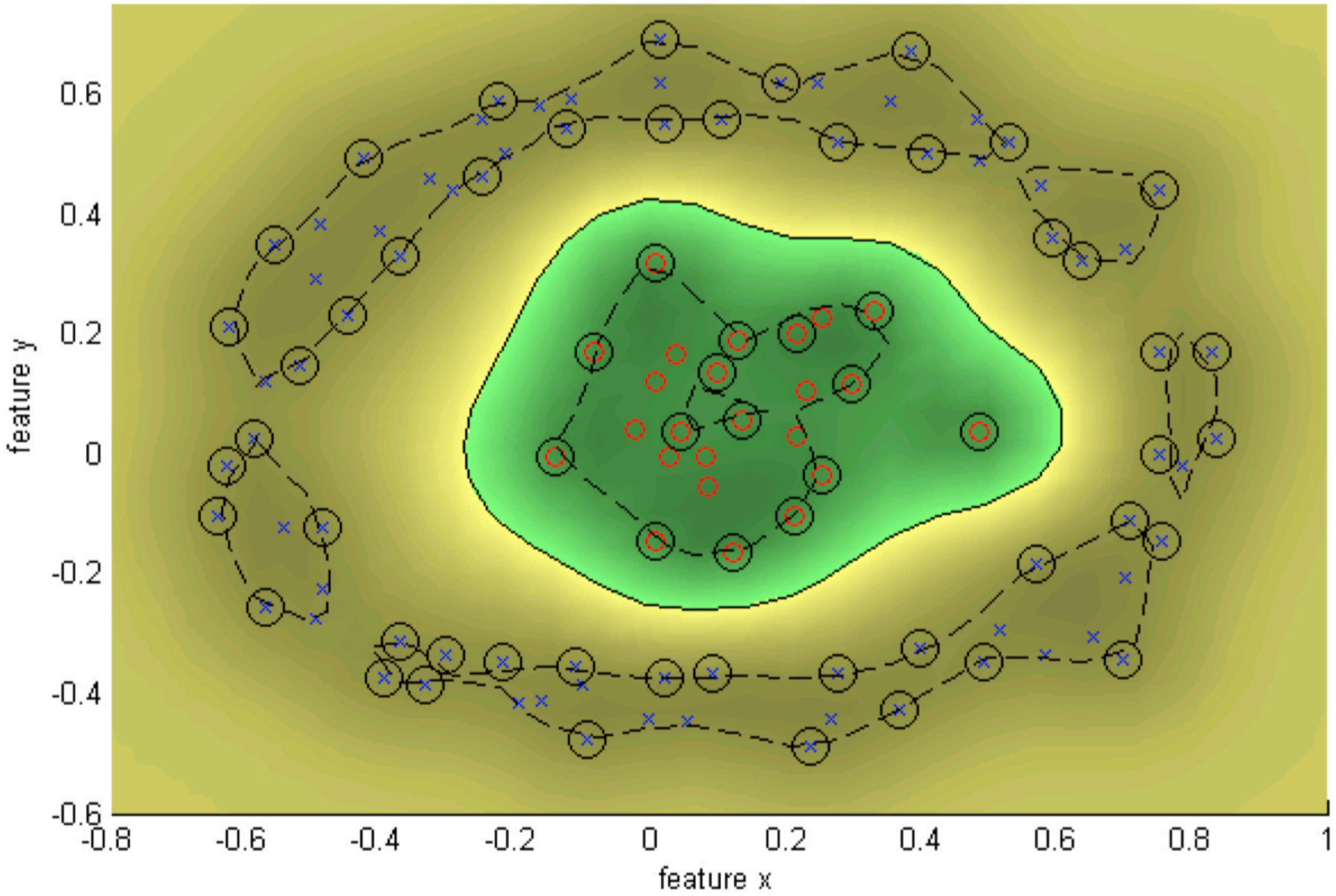
$\sigma = 0.25 \quad C = \infty$

nonlinear effect



$\sigma = 0.1 \quad C = \infty$

overfitting



Pros & Cons for Kernel SVM

Kernel: Linear, higher-order polynomial, and radial basis function (rbf), …

Pro SVM:

Typically good accuracy

Need small memory

(because use of only a subset of training points in the decision phase)

Work well with a clear margin of separation and with high dimensional space

Con SVM:

Slow for large datasets because of its high training time

Works poorly with overlapping classes

Sensitive to the type of kernel used

Hyperparameters from Bayesian Optimization

Is everybody on the floor?
We put some energy into this place
I want to ask you something
Are you ready for the sound of Scooter?
I want to see you sweat
I said, I want to see you sweat, yeah
Hyper, Hyper
Hyper, Hyper
Hyper, Hyper
We need the bass drum
Come on
Hyper, Hyper
Hyper, Hyper
It's so beautiful to see your hands in the air
Put your hands in the air
Come on
This is Scooter
We want to sing a big shout to U.S. and to all ravers in the world
And to Westbam, Marusha, Stevie Mason, The Mystic Man, DJ Dick
Carl Cox, The Hooligan, Cosmic, Kid Paul, Dag, Mike Van Dyke
Jens Lissat, Lenny D., Sven Väth, Mark Spoon, Marco Zaffarano
Hell, Paul Elstak, Mate Galic, Roland Casper, Sylvie, Miss Djax
Jens Mahlstedt, Tanith, Laurent Garnier, Special, Pascal F.E.O.S.
Gary D., Scotty, Gizmo and to all DJs all over the world
Hyper, Hyper
Sit there
Be good
Bye-bye

Hyperparameters from Bayesian Optimization

Grid Search:
Systematic but stupid

Random Search:
Samples from given distributions

Q: Alternatives to GridSearch and Random Search?

A: State-of-the-art is Bayesian Optimization

Tree Parzen Estimator (TPE)
is based on surrogate objective function

A hyperparameter λ_i can be continuous, integer-valued or categorical

$$\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$$

Hyperparameter domain

$$\Lambda = (\Lambda_1, \Lambda_2, \ldots, \Lambda_n)$$

Lossfunction

$$\mathscr{L}(\lambda, \mathscr{D}_{\texttt{train}}, \mathscr{D}_{\texttt{test}})$$

Data

$\mathscr{D}_{\texttt{train}}$

$\mathscr{D}_{\texttt{test}}$

Objective from, e.g., k-fold cross-validation

$$f(\lambda) = \frac{1}{k} \sum_i \mathscr{L}(\lambda, \mathscr{D}^{(i)}_{\texttt{train}}, \mathscr{D}^{(i)}_{\texttt{test}})$$

Optimal hyperparameters

$$\lambda^\star = \texttt{argmin}_\lambda f(\lambda)$$

Model	**Hyperparameter optimization**	Implementation
Gradient Boosting	Random search: `max_features`, `min_samples_leaf`, `max_depth`, `learning_rate`, `n_estimators`	SCIKIT-LEARN
Random Forest	Random search: `min_samples_split`, `n_estimators`, `max_features`	SCIKIT-LEARN
Gaussian Process	MCMC sampling over hyperparameters	SPEARMINT
SVR	Random search: `C` and `gamma`	SCIKIT-LEARN
NuSVR	Random search: `C`, `gamma` and `nu`	SCIKIT-LEARN
k-nearest-neighbours	Random search: `n_neighbors`	SCIKIT-LEARN
Linear Regression	None	SCIKIT-LEARN
Ridge Regression	Random search: `alpha`	SCIKIT-LEARN

K. Eggensperher et al., *Efficient Benchmarking of Hyperparameter Optimizers via Surrogates* (2015)

Tree Parzen Estimator (TPE)

based on surrogate objective function

Generally: The surrogate function is a probabilistic MCMC that maps hyperparameters to a probability of a score on the objective function
(Bayesian updating of the prior)

Tree Parzen Estimator (TPE) is based on
tree-structured Parzen density estimators

TPE is a generalist:
Can handle continuous, categorical, and conditional parameters

`Python example in lecture`

Recommended reading:
`https://sebastianraschka.com/Articles/2014_kernel_density_est.html`

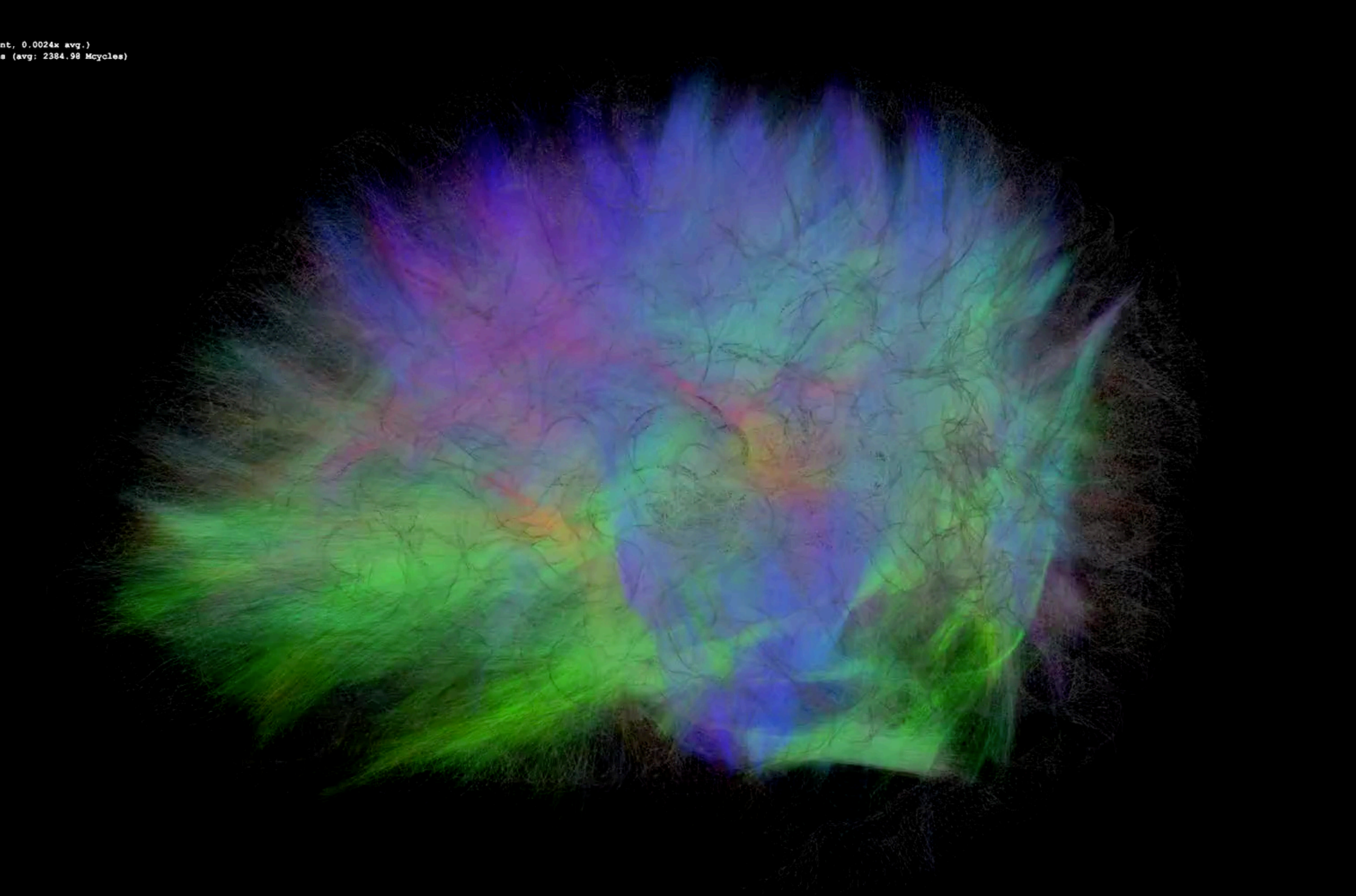
Final part:
News, Views & Threats in Machine Learning

Jan Nagler

Deep Dynamics Group
Centre for Human and Machine Intelligence (HMI)
Frankfurt School of Finance & Management

Frankfurt, October 2019

nt, 0.0024x avg.)
s (avg: 2384.98 Mcycles)

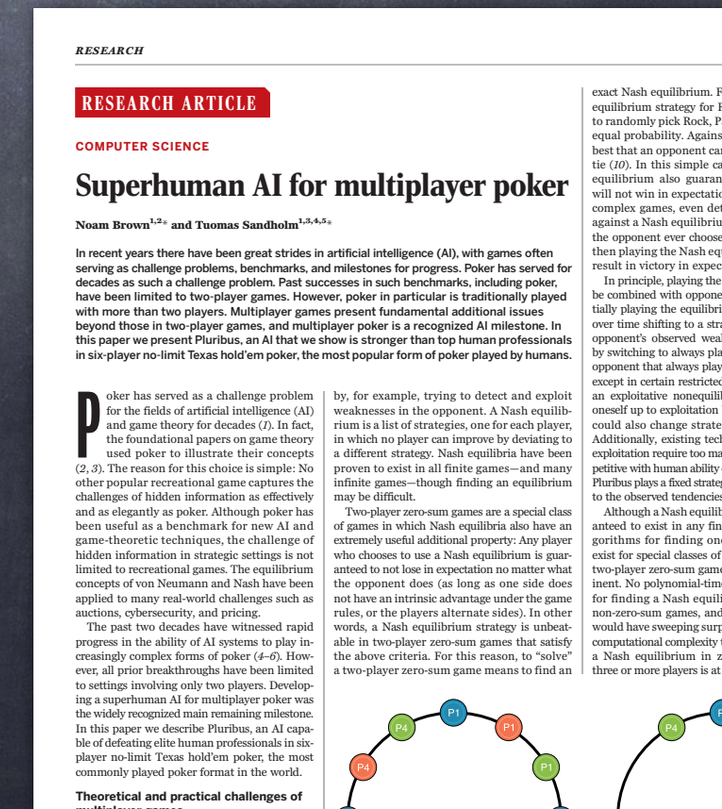


Simulated Thalamocortical Brain Network, 3 Million Neurons, 476M Synapses, Ivan Dimkovic

Selected Milestones Machine Learning

1763/1812: Bayesian Inference
1913: Marko Chains
1943: Neural Networks
1957: Perceptron
1970: Backpropagation & ’AI Winter'
1989: Reinforcement Learning
1995: SVM & Random Forrests
1997: LSTM (Hofreiter & Schmidhuber)
1998: MNIST (Yann LeCun)
2006: Netflix prize (DBN)
2009: ImageNet (Fei-Fei Li)
2010: Deep Learning rises
2012: Dropout
2011: Jeopardy (IBM Watson, NLP)
2014: DeepFace, GAN (Facebook)
2015: ResNet (Human lose in image clf)
2016: AlphaGo (Google)
2017: AlphaZero (Google)
2018: BERT (Google, NLP++)
2018: Turing Award for Bengio, Hinton, LeCun
2019: Superhuman AI for multiplayer Poker

Not shown:
Hardware, architecture development,
CPU, GPU, TPU (Moore’s law)
ML Libraries and Ecosystem
(Tensorflow, Keras, …)



RESEARCH

RESEARCH ARTICLE

COMPUTER SCIENCE

Superhuman AI for multiplayer poker

Noam Brown[1,2]* and Tuomas Sandholm[1,3,4,5]*

In recent years there have been great strides in artificial intelligence (AI), with games often serving as challenge problems, benchmarks, and milestones for progress. Poker has served for decades as such a challenge problem. Past successes in such benchmarks, including poker, have been limited to two-player games. However, poker in particular is traditionally played with more than two players. Multiplayer games present fundamental additional issues beyond those in two-player games, and multiplayer poker is a recognized AI milestone. In this paper we present Pluribus, an AI that we show is stronger than top human professionals in six-player no-limit Texas hold’em poker, the most popular form of poker played by humans.

Poker has served as a challenge problem for the fields of artificial intelligence (AI) and game theory for decades (*1*). In fact, the foundational papers on game theory used poker to illustrate their concepts (*2*, *3*). The reason for this choice is simple: No other popular recreational game captures the challenges of hidden information as effectively and as elegantly as poker. Although poker has been useful as a benchmark for new AI and game-theoretic techniques, the challenge of hidden information in strategic settings is not limited to recreational games. The equilibrium concepts of von Neumann and Nash have been applied to many real-world challenges such as auctions, cybersecurity, and pricing.

The past two decades have witnessed rapid progress in the ability of AI systems to play increasingly complex forms of poker (*4*–*6*). However, all prior breakthroughs have been limited to settings involving only two players. Developing a superhuman AI for multiplayer poker was the widely recognized main remaining milestone. In this paper we describe Pluribus, an AI capable of defeating elite human professionals in six-player no-limit Texas hold’em poker, the most commonly played poker format in the world.

Theoretical and practical challenges of

by, for example, trying to detect and exploit weaknesses in the opponent. A Nash equilibrium is a list of strategies, one for each player, in which no player can improve by deviating to a different strategy. Nash equilibria have been proven to exist in all finite games—and many infinite games—though finding an equilibrium may be difficult.

Two-player zero-sum games are a special class of games in which Nash equilibria also have an extremely useful additional property: Any player who chooses to use a Nash equilibrium is guaranteed to not lose in expectation no matter what the opponent does (as long as one side does not have an intrinsic advantage under the game rules, or the players alternate sides). In other words, a Nash equilibrium strategy is unbeatable in two-player zero-sum games that satisfy the above criteria. For this reason, to “solve” a two-player zero-sum game means to find an

exact Nash equilibrium. F
equilibrium strategy for
to randomly pick Rock, P
equal probability. Agains
best that an opponent ca
tie (*10*). In this simple ca
equilibrium also guaran
will not win in expectatio
complex games, even de
against a Nash equilibriu
the opponent ever choose
then playing the Nash eq
result in victory in expec

In principle, playing the
be combined with oppone
tially playing the equilibri
over time shifting to a str
opponent’s observed wea
by switching to always pla
opponent that always play
except in certain restricted
an exploitative nonequili
oneself up to exploitation
could also change strate
Additionally, existing tech
exploitation require too ma
petitive with human ability
Pluribus plays a fixed strate
to the observed tendencies

Although a Nash equilib
anteed to exist in any fin
gorithms for finding on
exist for special classes of
two-player zero-sum game
inent. No polynomial-tim
for finding a Nash equili
non-zero-sum games, and
would have sweeping surp
computational complexity
a Nash equilibrium in z
three or more players is at

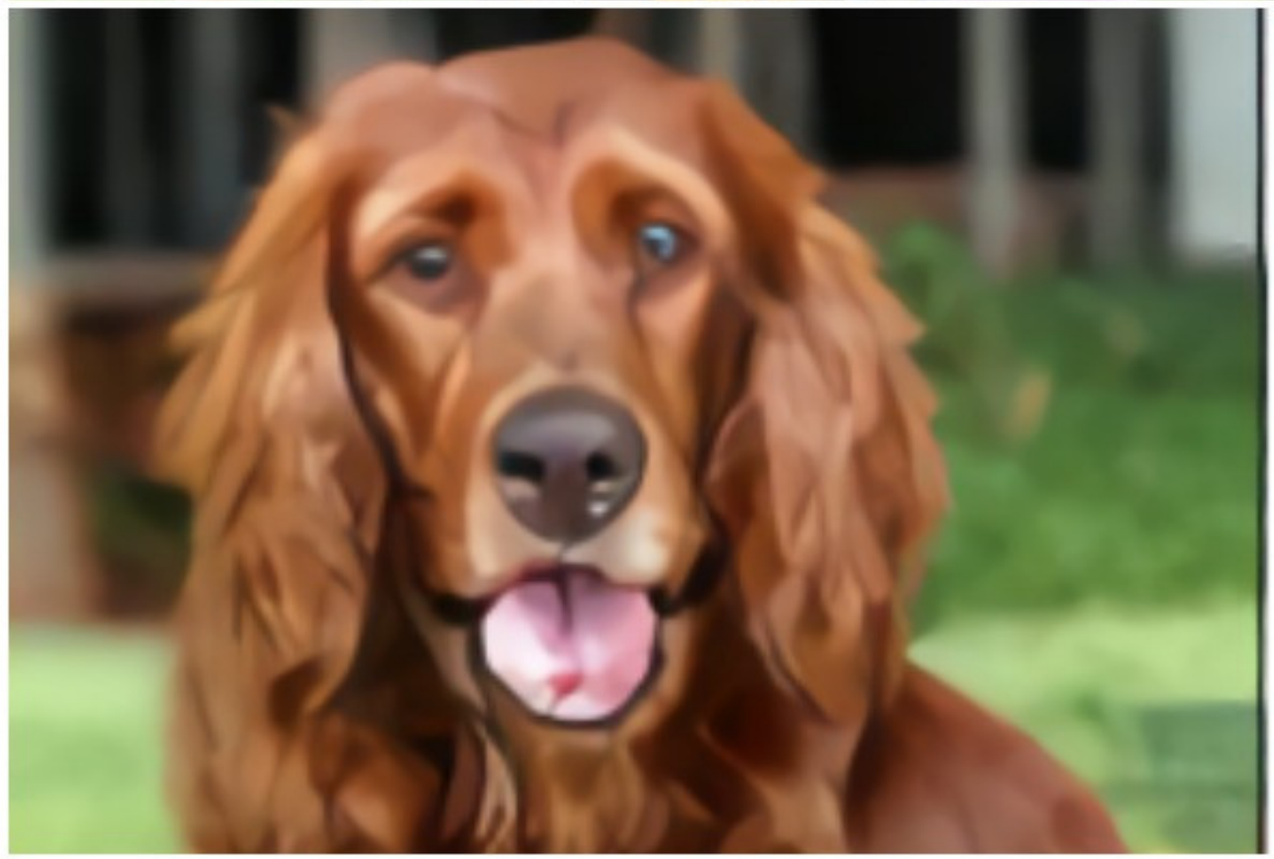
Introducing CoastRunner…

Reinforcement learning (RL)

Misspecified reward functions causing odd RL behavior in *CoastRunners*

Surrogates generated by Generative Adversarial Networks at DeepMind

Surrogates generated by Generative Adversarial Networks at DeepMind



Surrogates generated by Generative Adversarial Networks at DeepMind

Surrogates generated by Generative Adversarial Networks, Kaggle Dog Competition 2019

Real threats

Deep learning removes human decisions
(Autonomous driving, Trolley dilemma)

Warfare

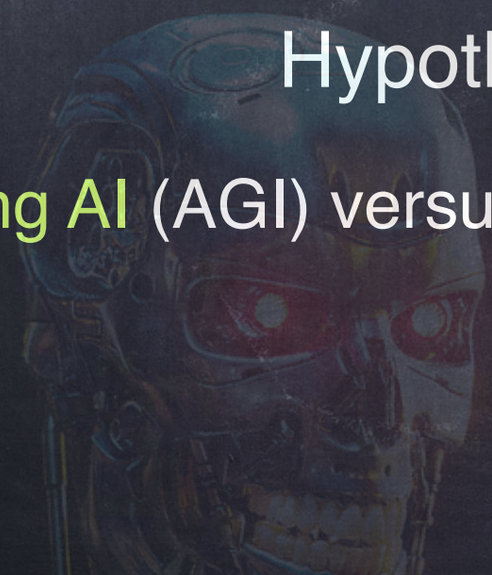
Drones (with and w/o face recognition)

Undermining democratic values

Deception and Influencing in Online Social Networks
Echo Chambers, Bots
Fake Pictures, fake videos & fake news

Hypothetical threats

Strong AI (AGI) versus risk assessment and panic



Autonomous killer drones set to be used by Turkey in Syria

TECHNOLOGY 20 September 2019

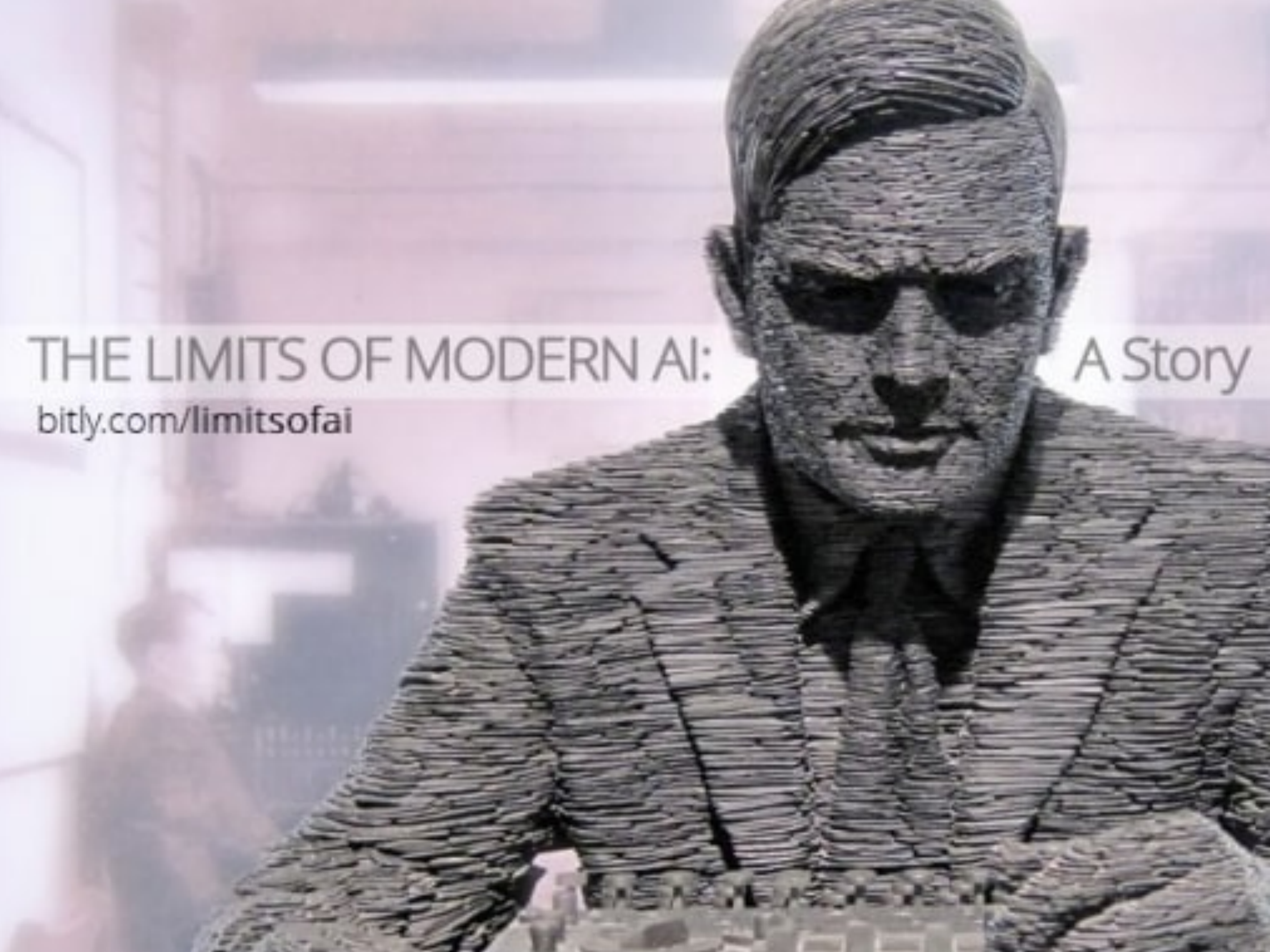
By David Hambling

The Turkish army is increasingly using drones in combat
Soner Kilinc/Anadolu Agency/Getty

Turkey is to become the first nation to use drones able to find, track and kill people without human intervention.

THE LIMITS OF MODERN AI: A Story

bitly.com/**limitsofai**



Limits of Machine Learning

Artificial General Intelligence (AGI)?
(Elon Musk, Sam Harris)

Common sense gap!
(Gary Marcus, Francois Chollet)

Currently many mundane limitations:
No advanced natural scene understanding,
lack of trivial understanding of relations, physics, irony,
of what is actually really going on,
what matters, what is next, …

thesis

The limits of machine prediction

Ten years ago, scientists trained an algorithm (M. Schmidt and H. Lipson, *Science* **324**, 81–85; 2009) to seek patterns in the erratic behaviour of a chaotic double pendulum. The algorithm, using a learning strategy inspired by evolution, was soon able to discover the laws of motion on its own, despite being given no scientific knowledge by the researchers. "Without any prior knowledge about physics, kinematics, or geometry," they noted, "the algorithm discovered Hamiltonians, Lagrangians, and other laws of geometric and momentum conservation." A machine had, apparently, deduced laws of physics.

This kind of success fuels an increasingly widely held belief that the explosive rise in technology for data gathering and analysis may soon make the scientific method unnecessary. After several centuries of science driven by the profitable interplay of observation and theory, the idea goes, we're now moving into a new era in which theory and conceptual understanding will play a smaller role, if any role at all. Forget models and hypotheses, all we will need are smart machine-learning algorithms devouring huge datasets. Through fully automated learning, we'll come to make vastly more accurate predictions, and face fewer real-world surprises. Machines will do science for us.

Some computer scientists even believe we may one day find what they call the master algorithm — an ultimate data-processing device that, running quite independently from human oversight, will organize human politics to "speed poverty's decline" and make everyone's lives "longer, happier and more productive" (P. Domingos, *The Master Algorithm*; 2015).

Of course, we're all rapidly growing accustomed to machine-learning tools impinging on all aspects of our lives. They make decisions on bank loans and credit risks, determine the advertisements and prices we see when shopping online, and recommend medical diagnoses. Artificial intelligence is already changing scientific practice too — from materials scientists using algorithms to explore the space of possible substances to physicists using them to design quantum networks. But is the bigger vision at all plausible? Probably not, and not only because of the unreasonable optimism of the technology's enthusiasts. For all its likely power and value, big data will only ever be a part of the scientific enterprise. On its own, it probably won't even lead to better predictions. In many cases, we can expect that more data may actually lead to worse predictions.

That point was made in a recent essay by physicists Hykel Hosni and Angelo Vulpiani (preprint at https://arxiv.org/abs/1705.11186; 2017). The conclusion follows by noting some of the lessons to be learned from simple examples, including weather forecasting. The main lesson is that the best predictions generally come from a judicious trade-off between modelling and quantitative analysis. Conceptual insight counts as much as quantity of data. 'Big data need big theory too', as the title of another paper making a similar point in the context of biology goes (P. V. Coveney, E. R. Dougherty and R. R. Highfield, *Phil. Trans. R. Soc. A* https://doi.org/10.1098/rsta.2016.0153; 2016).

The best predictions generally come from a judicious trade-off between modelling and quantitative analysis.

One expectation about limits on the predictive accuracy of big data comes from dynamical systems theory in the context of high-dimensional systems — the norm for most real-world applications. Some degree of predictability is guaranteed by the notion of Poincaré recurrence, which applies to any Hamiltonian system, as well as to the dynamics on the attractor of any dissipative system. Poincaré's insight was that, given any current state of a system, one can be sure that the system will return to a neighbourhood of this state some time in the future. How long we should expect this time to be is the crucial question.

As Hosni and Vulpiani note, it's easy to estimate this time — it is proportional to ε^{-D}, with ε being some small number reflecting the precision of the prediction and D being the system dimension. This means that, with sufficient data specifying the current system state, useful predictions can be made, at least for low-dimensional systems. But if D is large — 10 or larger, for example — then the time of recurrence is astronomically large even if the aim is for predictions of fairly low precision.

Hence, the possibility of prediction from data alone may be guaranteed in theory, but it is often impossible in practice — even considering the clean situation in which one has perfectly precise information about the system. It's the system dimension at fault. Making useful predictions in high-dimensional systems therefore requires something more than data — theory or conceptual insight. "Nothing is more practical," as the Austrian physicist Ludwig Boltzmann once wrote, "than a good theory."

Indeed, this is precisely how weather forecasters make useful predictions — by giving up on some data, and working with less detailed models. As Hosni and Vulpiani review the history, it was a few decades after Lewis Fry Richardson pioneered numerical weather forecasting that von Neumann and Charney came to the conclusion that the equations Richardson had proposed, though correct from the standpoint of fluid dynamics and atmospheric physics, were not practically useful. They were, in fact, too accurate, as they included details on high-frequency wave motions having no importance for weather forecasting. Accuracy was achieved in numerical weather forecasting by scientists realizing that too much data can be a bad thing — and it took insight to build simpler and more capable models.

In 2008, the editor of the technology magazine *Wired* made the famous claim that "the data deluge makes the scientific method obsolete." Since then, this view has gained significant backing — no doubt, in large part, because of the commercial benefits accruing to companies rolling out their algorithms into every corner of modern society. Among many others, Hosni and Vulpiani suggest we should think again, and reign in our expectations.

We should also work hard to spread understanding about the likely limits to machine prediction, and the many ways false belief may lead to misuse and painful mistakes.

Clearly, if the day comes that computers can do all the things human brains can do, including generating creative insights and conceptual revolutions, then the human era of science may be over. We're still a long way from that. ❐

Mark Buchanan

Published online: 1 April 2019
https://doi.org/10.1038/s41567-019-0489-5

Limits of Machine Learning: Reasoning

http://karpathy.github.io/2012/10/22/state-of-computer-vision/

Why theory?

PHILOSOPHICAL TRANSACTIONS A

rsta.royalsocietypublishing.org

Opinion piece

Cite this article: Coveney PV, Dougherty ER, Highfield RR. 2016 Big data need big theory too. *Phil. Trans. R. Soc. A* **374**: 20160153. http://dx.doi.org/10.1098/rsta.2016.0153

Accepted: 17 June 2016

One contribution of 17 to a theme issue 'Multiscale modelling at the physics–chemistry–biology interface'.

Subject Areas:

Big data need big theory too

Peter V. Coveney[1], Edward R. Dougherty[2] and Roger R. Highfield[3]

[1]Centre for Computational Science, University College London, Gordon Street, London WC1H 0AJ, UK
[2]Center for Bioinformatics and Genomic Systems Engineering, Texas A&M University, College Station, TX 77843-31283, USA
[3]Science Museum, Exhibition Road, London SW7 2DD, UK

PVC, 0000-0002-8787-7256

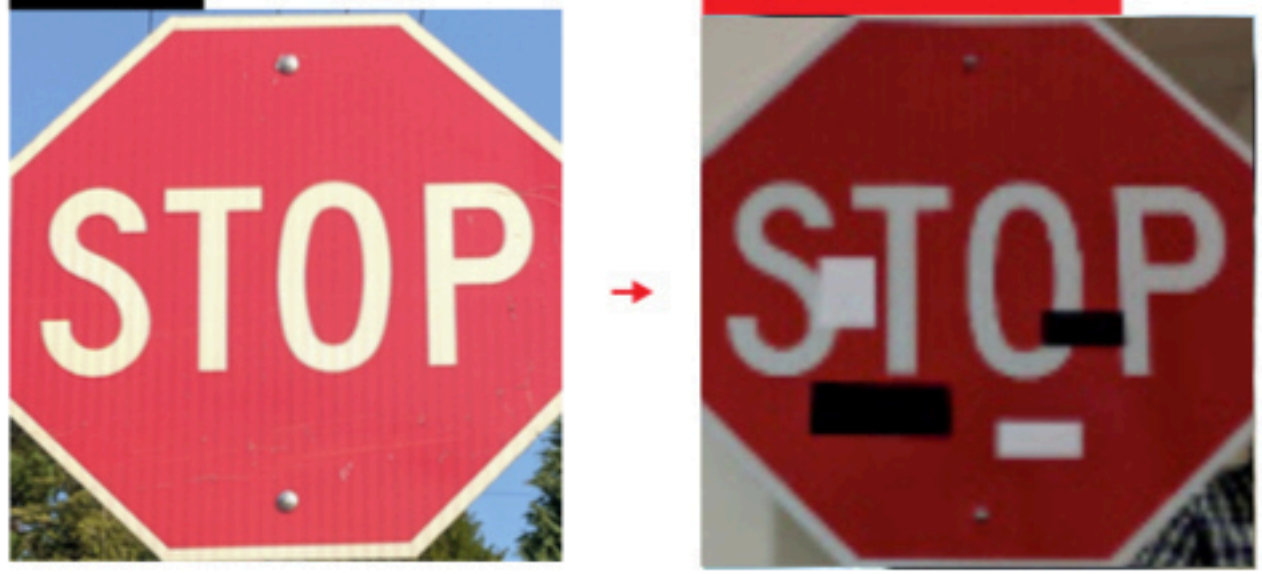
The current interest in big data, machine learning and data analytics has generated the widespread impression that such methods are capable of solving most problems without the need for conventional scientific methods of inquiry. Interest in these methods is intensifying, accelerated by the ease with which digitized data can be acquired in virtually all fields of endeavour, from science, healthcare and cybersecurity to economics, social sciences and the humanities. In multiscale modelling, machine learning appears

Source: Deep trouble for deep networks, 10 OCTOBER 2019 I VOL 574 I NATURE I 163

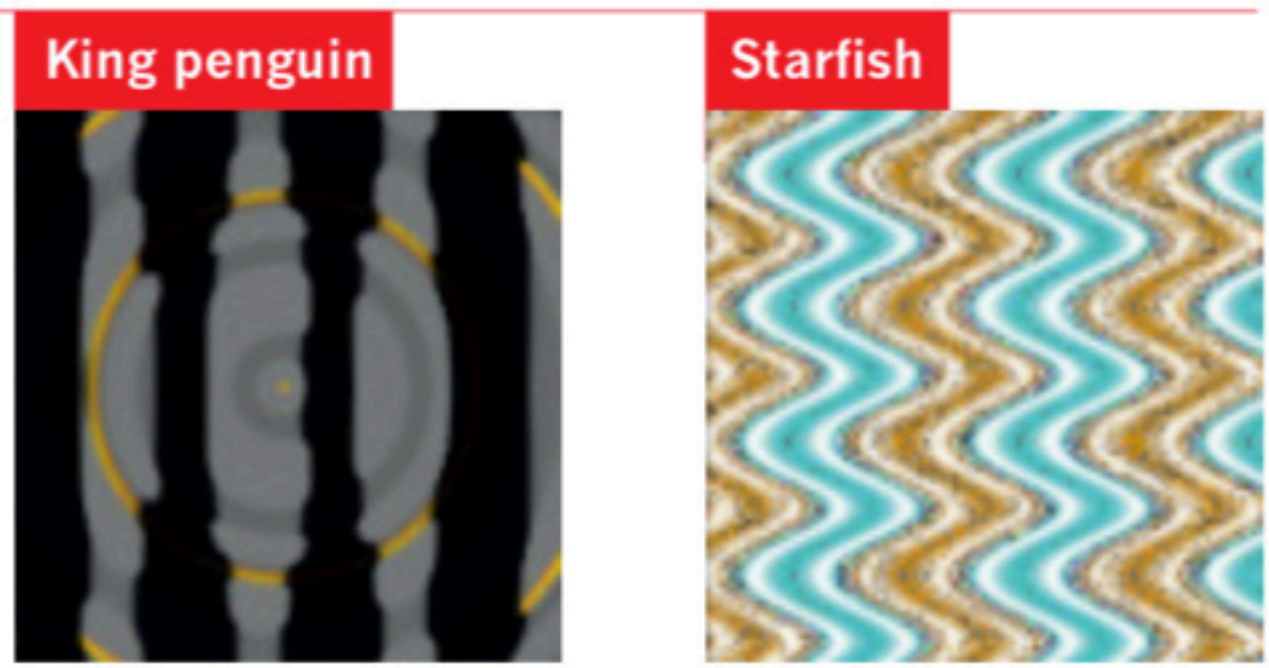
FOOLING THE AI

Deep neural networks (DNNs) are brilliant at image recognition — but they can be easily hacked.

These stickers made an artificial-intelligence system read this stop sign as 'speed limit 45'.



Scientists have evolved images that look like abstract patterns — but which DNNs see as familiar objects.



Source: Deep trouble for deep networks,
10 OCTOBER 2019 I VOL 574 I NATURE I 163

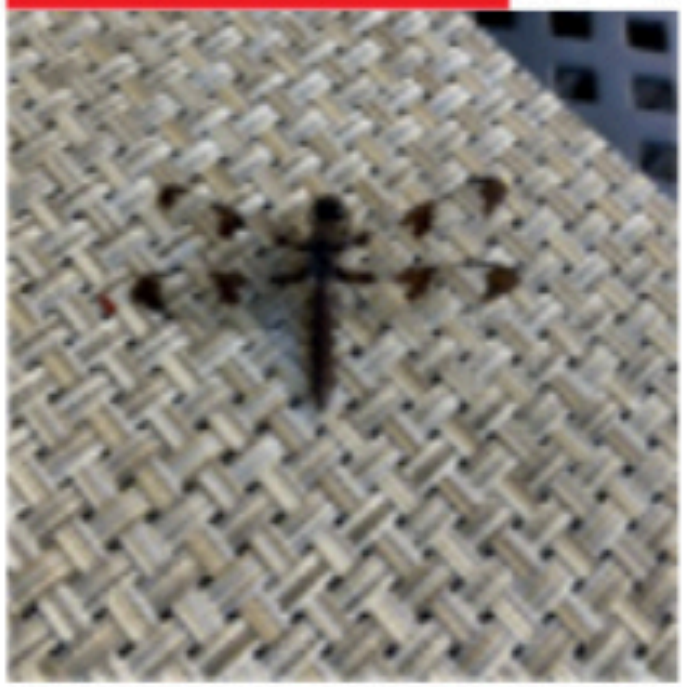
Adding carefully crafted noise to a picture can create a new image that people would see as identical, but which a DNN sees as utterly different.

In this way, any starting image can be tweaked so a DNN misclassifies it as any target image a researcher chooses.

Source: Deep trouble for deep networks, 10 OCTOBER 2019 I VOL 574 I NATURE I 163

Rotating objects in an image confuses DNNs, probably because they are too different from the types of image used to train the network.

Even natural images can fool a DNN, because it might focus on the picture's colour, texture or background rather than picking out the salient features a human would recognize.



Further reading

Deep trouble for deep networks,
10 OCTOBER 2019 I VOL 574 I NATURE I 163

Cheat Sheet by Fjodor van Veen

REVIEW

doi:10.1038/nature14539

Deep learning

Yann LeCun[1,2], Yoshua Bengio[3] & Geoffrey Hinton[4,5]

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input.

Representation learning is a set of methods that allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using a general-purpose learning procedure.

Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition[1–4] and speech recognition[5–7], it has beaten other machine-learning techniques at predicting the activity of potential drug molecules[8], analysing particle accelerator data[9,10], reconstructing brain circuits[11], and predicting the effects of mutations in non-coding DNA on gene expression and disease[12,13]. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding[14], particularly topic classification, sentiment analysis, question answering[15] and language translation[16,17].

We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

Supervised learning

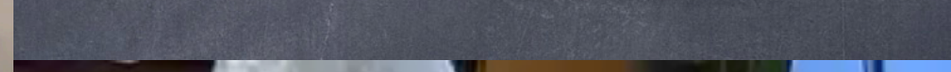
The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labelled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as 'knobs' that define the input–output function of the machine. In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labelled examples with which to train the machine.

To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector.

The objective function, averaged over all the training examples, can

[1]Facebook AI Research, 770 Broadway, New York, New York 10003 USA. [2]New York University, 715 Broadway, New York, New York 10003, USA. [3]Department of Computer Science and Operations Research Université de Montréal, Pavillon André-Aisenstadt, PO Box 6128 Centre-Ville STN Montréal, Quebec H3C 3J7, Canada. [4]Google, 1600 Amphitheatre Parkway, Mountain View, California 94043, USA. [5]Department of Computer Science, University of Toronto, 6 King's College Road, Toronto, Ontario M5S 3G4, Canada.

Thank you!



Reinforcement learning in cats

Clustering Methods

Zakariya Abu Grin, Viktor-Johannes Holl, Loa Marx,
Ananya Neogi, Lenka Ting
Master in Applied Data Science
Oktober 10, 2019

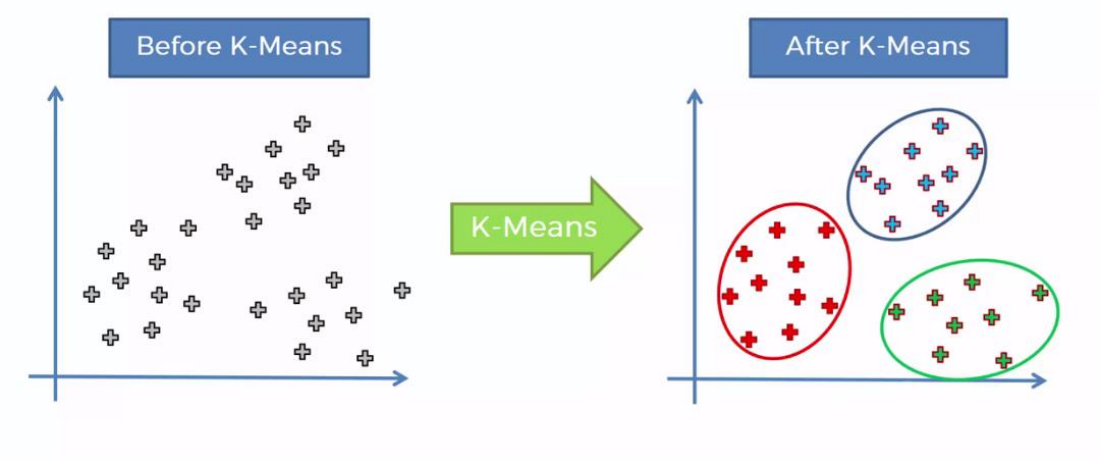
Overview

1. Introduction of 5 types of clustering methods
    - Partition Clustering
    - Model-based Clustering
    - Density-Based Clustering
    - Fuzzy Clustering
    - Hierarchical Clustering

1. Summarized Comparison

Partition Clustering

- Partitioning algorithms are clustering techniques that subdivide the data sets into a set of k groups, where k is the number of groups pre-specified.



Examples: k-means, k-mediods clustering or PAM(*Partitioning Around Medoids)*, Clara Algorithm.

Model-based Clustering

Approach:

- A probabilistic approach where data is assumed to be generated from some mixture of probability distributions.

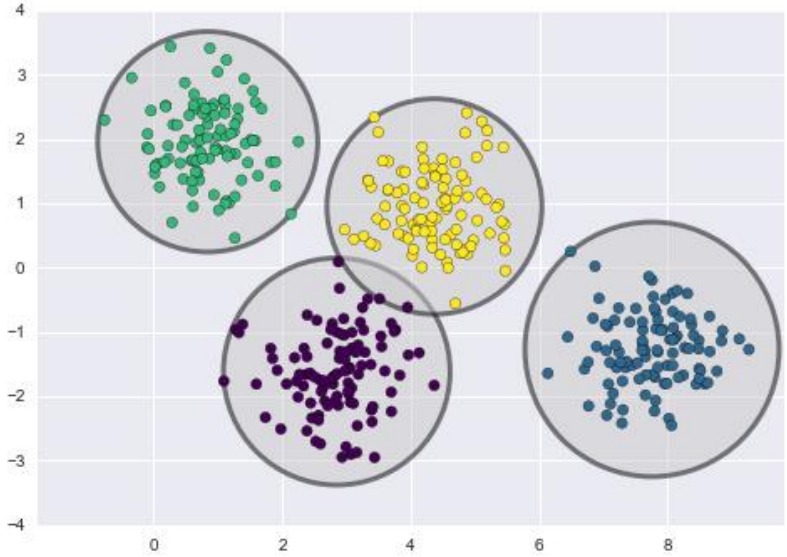
Outcome:

- The probability of each point data point belonging to each cluster.
- Data points are assigned based on the highest probability (**Soft Clustering**) or by 0 and 1 probability based on some rule/boundary (**Hard Clustering**).

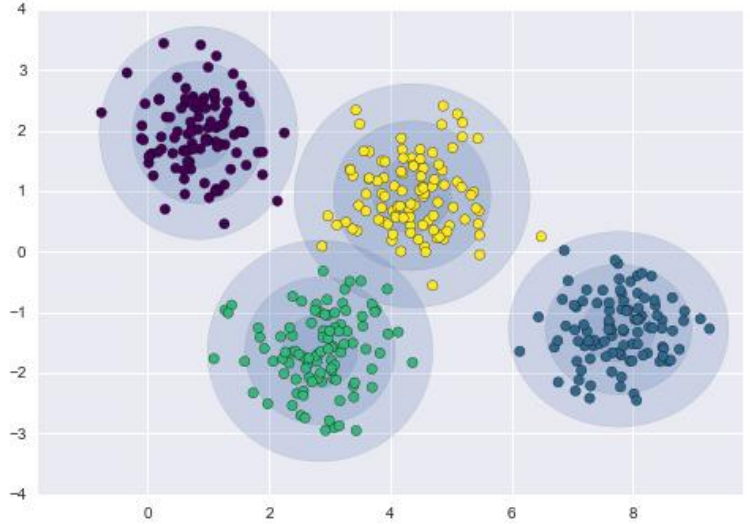
Reference: “Bayesian Analysis with Python”

Model-based Clustering

K-means

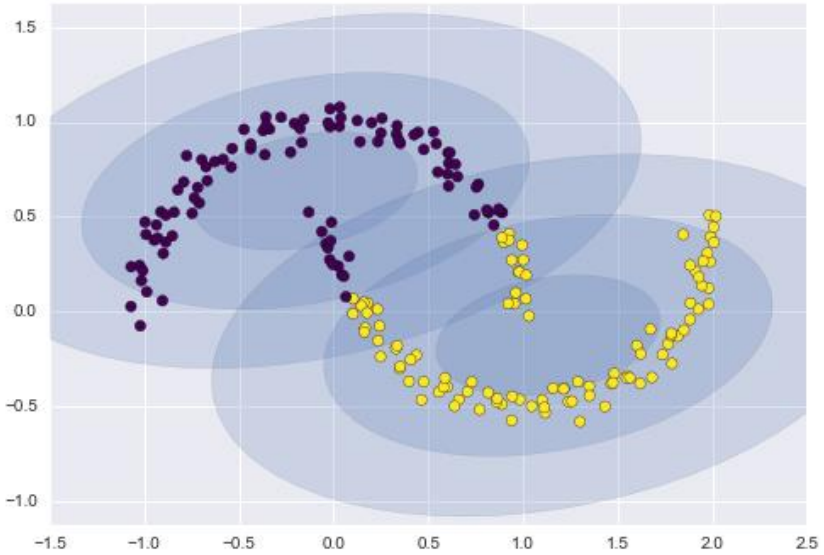


Gaussian Mixture Models (GMMs)



Model-based Clustering

Gaussian Mixture Models (GMMs)



Reference: Python Data Science Handbook

Density-Based Clustering

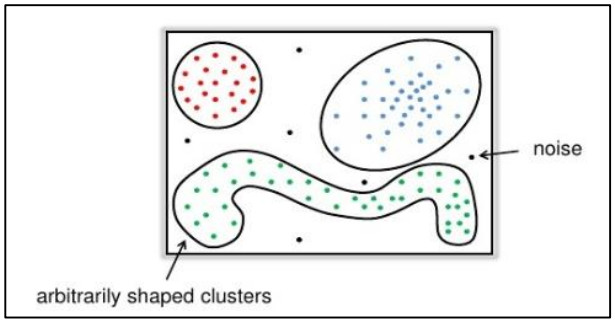
- Identify distinctive groups/clusters in a data set, based on the idea that a cluster in a data space is a continuous region of high point density, separated from other such clusters by contiguous regions of low point density
- Based on connectivity and density functions
- Locates regions of high density that are separated from one another by regions of low density
- Two parameters:
  - 1. maximum radius of the neighbourhood→Eps
  - 2. minimum number of points in the Eps neighbourhood of a point→MinPts

Pros:

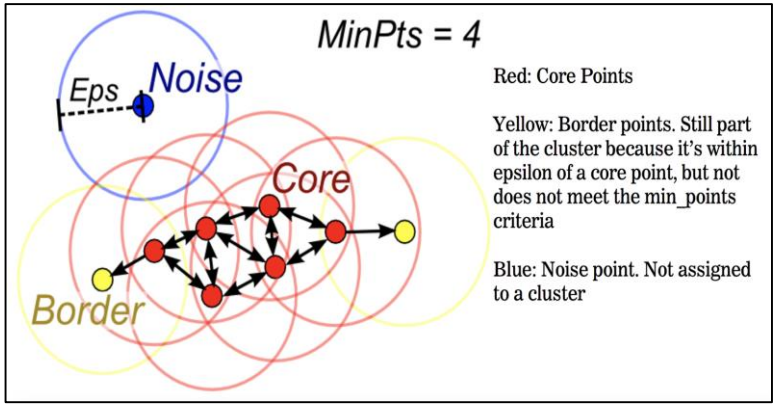
- Is great at separating clusters of high density from clusters of low density within a given dataset
- Is great with handling outliers within the dataset

Cons:

- struggles with clusters of similar density
- Sensitive to the settings of parameters



Density-Based Clustering



DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

Algorithm:

- Arbitrarily select a point p
- Retrieve all points density-reachable from p w.r.t. Eps and MinPts
  - If p is a core point, a cluster is formed
  - If p is a border point, no points are density-reachable from p, and DBSCAN goes to the next point of the dataset
- Continue the process until all of the points have been processed

Visualization Example

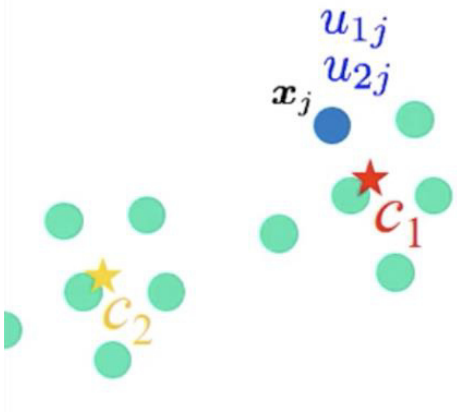
- https://www.youtube.com/watch?v=h53WMIImUuc

Fuzzy Clustering

- Soft Method
- An item can be part of more than one cluster
- Each has a set of membership coefficients corresponding to the degree of being in a given cluster.

Fuzzy C-means

- The centroid of a cluster is calculated as the mean of all points, weighted by their degree of belonging to the cluster.
- Data points close to center of a cluster -> get higher degree
  (total membership for a point must add to 1)



u1j + u2j =1

Fuzzy C-means steps

1. Specify k (number of clusters) and m.
2. Assign randomly to each point coefficients for being in the clusters.
3. Repeat the followings until (i) we reach the max number of iterations (ii) the algorithm has converged.

   a. Compute the centroid for each cluster.

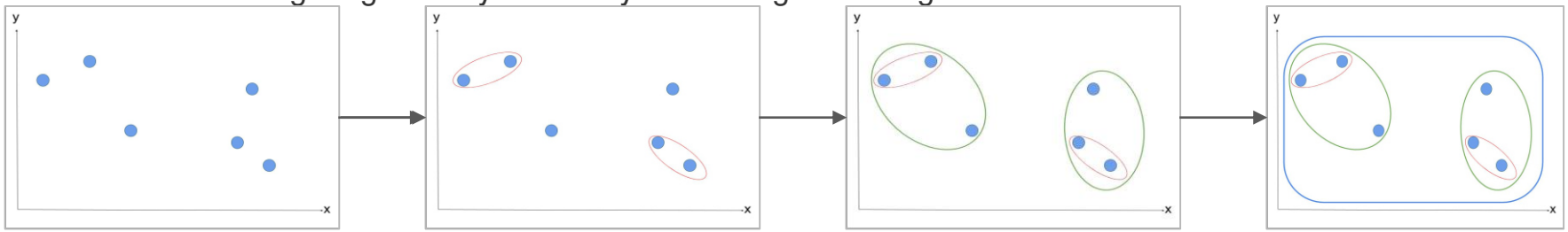
   b. For each point, compute its coefficients of being in the clusters.

Brief Comparison

	K-means	**Fuzzy C-means**
Calculation Speed	Fair	Low
Function	$\sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$	$\sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij}^m \|\mathbf{x}_i - \mathbf{c}_j\|^2$,
Implementation	Traditional case/ Limited	• Can handle uncertainty • Can be used in variety of clusters

Hierarchical Clustering

1. Each node is its own cluster
2. Clusters merge together by minimally increasing the linkage distance



Advantages:

- No need to specify “k” clusters
- Produces different levels of resolution
- Flexible regarding the metric used for merging

Disadvantages:

- Time efficiency is not optimal
- Suited for smaller datasets
- Decide which level of resolution is sensible

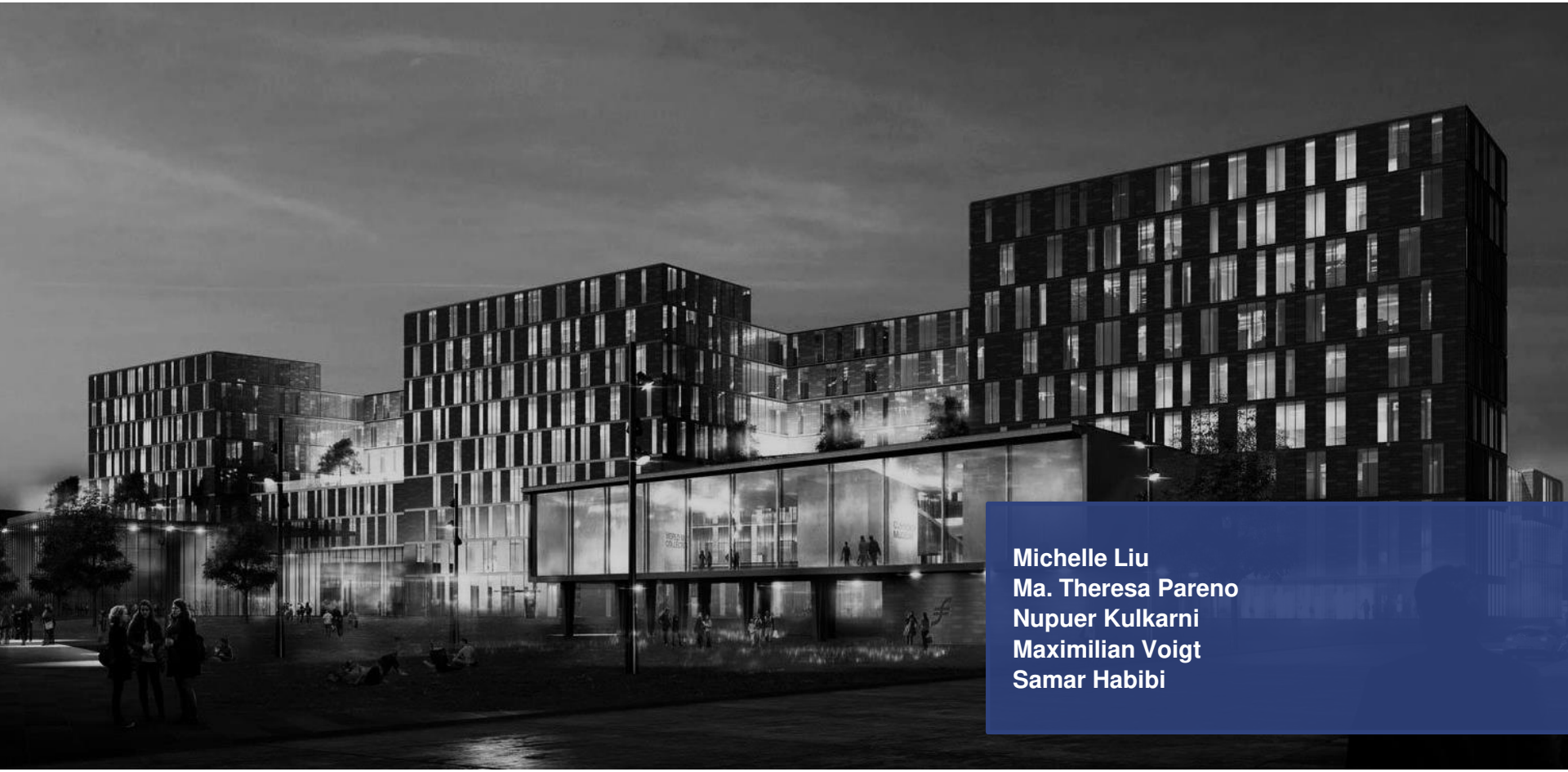
Comparative Study of the Clustering methods

	Partition	Hierarchical	Fuzzy	Model-based	Density-based
Core Concept	Datapoints are assigned to moving centroids	Datapoints are merged according to distance	Datapoints have multiple weighted memberships	Datapoints are assumed to belong to some distributions	Identify distinctive groups/clusters in a dataset
Advantages	- No overlap between the clusters - Good computational performance	- No specification of “k”	More applicable and robust	One of the fastest	- Separating clusters of high density - Handling outliers within the dataset
Limitations	- Sensitive to outliers - “k” must be specified	- Computationally expensive - Suited for small datasets	- Computationally expensive - Result depends on the initial choice of weights	Suffers with non-gaussian distributed data	Struggles with clusters of similiar density
Hard/Soft Clustering	Hard	Hard	Soft	Soft	Soft
Algorithms	k-means, Clara, k-mediods (PAM)	AggloC	Fuzzy C-Means	Gaussian Mixture Models	-DBSCAN

References

- https://hdbscan.readthedocs.io/en/latest/comparing_clustering_algorithms.html
- https://www.datanovia.com/en/blog/types-of-clustering-methods-overview-and-quick-start-r-code/#partitioning-clustering
- https://www.datanovia.com/en/courses/partitional-clustering-in-r-the-essentials/
- https://medium.com/predict/three-popular-clustering-methods-and-when-to-use-each-4227c80ba2b6
- https://ethz.ch/content/dam/ethz/special-interest/gess/computational-social-science-dam/documents/education/Spring2015/datascience/clustering2.pdf
- https://medium.com/@elutins/dbscan-what-is-it-when-to-use-it-how-to-use-it-8bd506293818
- https://elementtechnologies.net/life-sciences-services/clinical-data-services

Team Blue: Distance



Michelle Liu
Ma. Theresa Pareno
Nupuer Kulkarni
Maximilian Voigt
Samar Habibi

A motivation for distance measures

Clustering of data is useful for the extraction of information

In order to classify data, measures of similarity (or distance) are necessary

A distance measure should ideally fulfill:

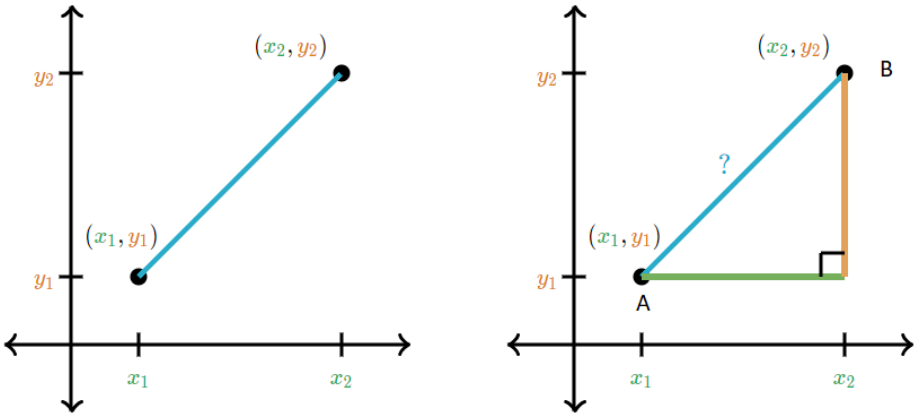
- Non-negativity: $d(a, b) \geq 0$
- Identity: $d(a, b) = 0 \leftrightarrow a = b$
- Symmetry: $d(a, b) = d(b, a)$
- Triangle inequality: $d(a, c) \leq d(a, b) + d(b, c)$

Euclidean Distance

Euclidean distance

Formula:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$



Property:

- It is a straight-line distance between two point, similar to using a ruler to actually measure the distance
- A generalized term for the Euclidean norm is the L2 norm

Drawbacks:

- Not a good metric in high dimensions
  - in high dimensional data, some variables are likely to be correlated, often highly correlated and Euclidean distance does not deal well with this.
  - due to the squared terms, is particular sensitive to noise
- Euclidean distance is the only metric that is the same in all direction, that is, rotation invariant.

Cosine Similarity

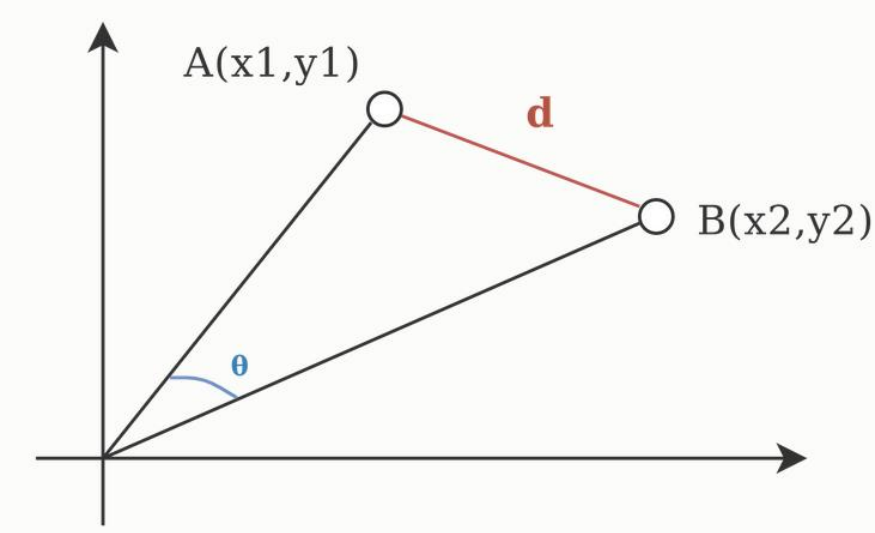
Cosine Similarity

Cosine similarity is a measure of similarity that measures the cosine of the angle between two non-zero vectors. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90° relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. The smaller the angle, higher the cosine similarity.

Our cosine similarity function can be defined as follows:

$$\cos\theta = \frac{x.y}{\|x\|\|y\|}$$

Where x and y are two vectors.



The cosine similarity is most used in:

- high-dimensional positive spaces
- for measuring distance when the magnitude of the vectors does not matter

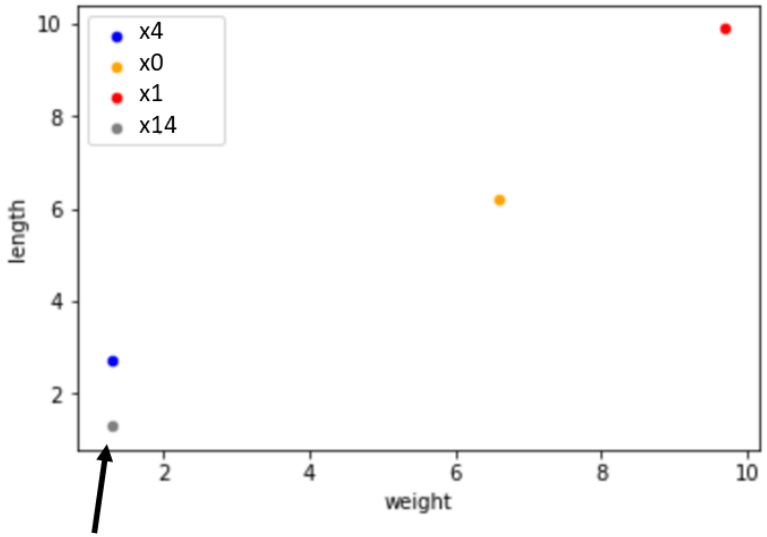
For example in:

- information retrieval
- text mining or working with text data represented by word counts

each term is notionally assigned a different dimension and a document is characterized by a vector where the value in each dimension corresponds to the number of times the term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter.

Source: XXX

Comparing Euclidean and Cosine



New data point x14

```
x0: [6.6 6.2]
x1: [9.7 9.9]
x4: [1.3 2.7]
x14: [1.3 1.3]
```

According to euclidean distance:
→ instance #14 is closest to #4

According to cosine similarity
→ instance #14 is closest to #1

Jaccard Similarity Index

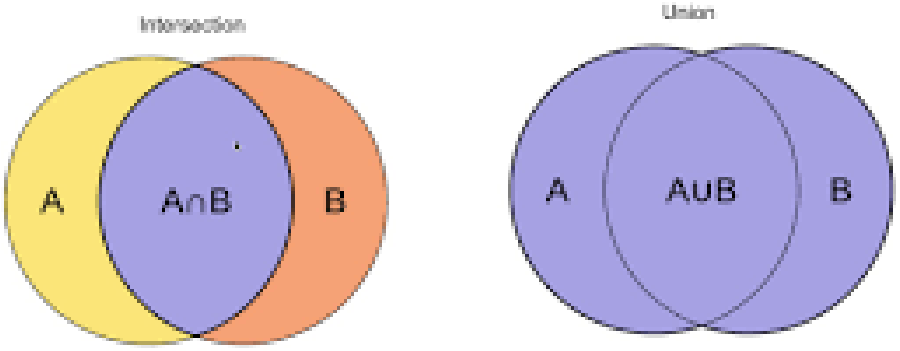
Jaccard Similarity Index

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

(If A and B are both empty, we define $J(A,B) = 1$.)

$$0 \leq J(A,B) \leq 1.$$

Jaccard coefficient



$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Jaccard Similarity Index

- Used for comparing similarity, dissimilarity, and distance of the data set.
- Only measures the similarity between finite sample sets.

Use Case:

- used for recommendation for similar items.
- Can be use for information retrieval

Issues:

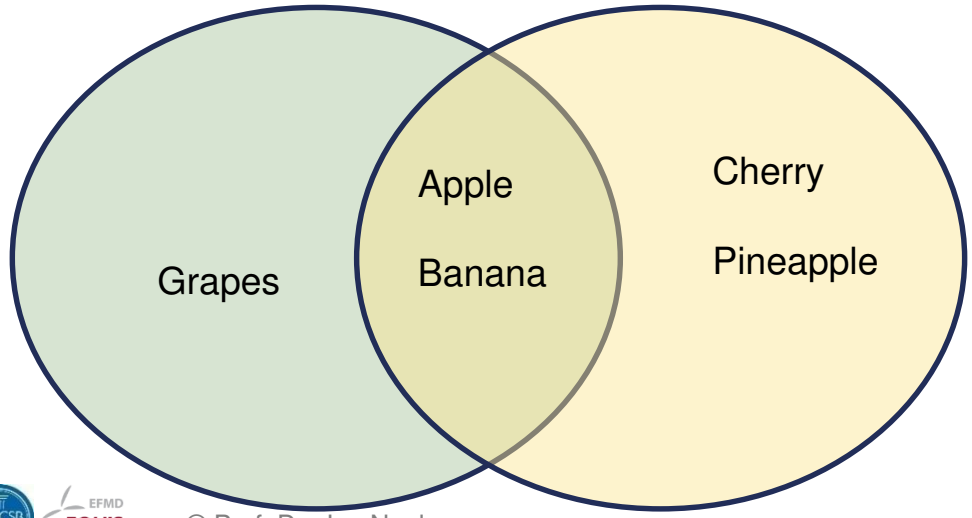
- Doesn't consider term frequency

Jaccard Similarity

Example:

Set A = {apple, banana, grapes}
Set B = {apple, banana, cherry, pineapple}

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



$$= \frac{2}{3 + 4 - 2}$$

= 0.4 or 40%
Similar

Jaccard Distance:

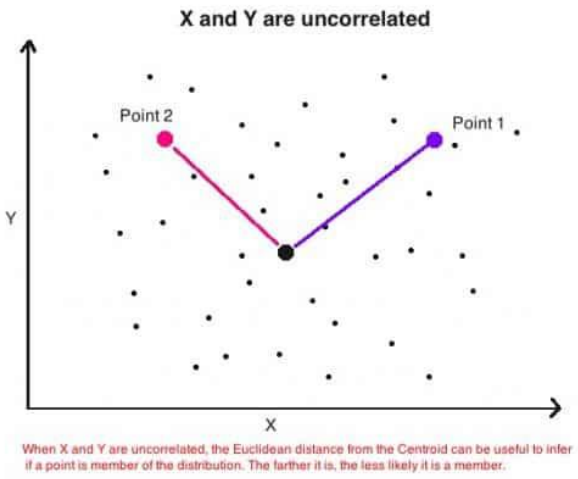
- Jaccard distance, is a measure of how *dissimilar* two sets are.

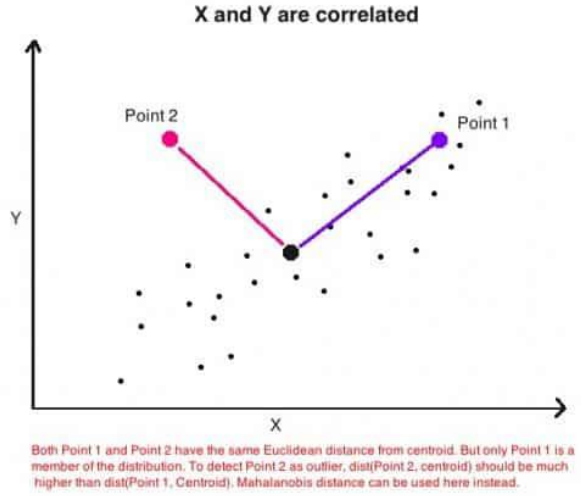
1 – 0.4 = 0.6

Mahalanobis distance

Problems with Euclidean distance

- Euclidean distance is the distance between two points
- Scaling of variables will affect distance
- If dimensions of the data are correlated, Euclidean distance does not yield information about closeness to the cluster (problem of non-spherical distributions)





- **Mahalanobis distance is a measure of the distance between a point P and a distribution D**
- **Mahalanobis distance is given by**

$$D^2 = (x - \mu)^T C^{-1} (x - \mu)$$

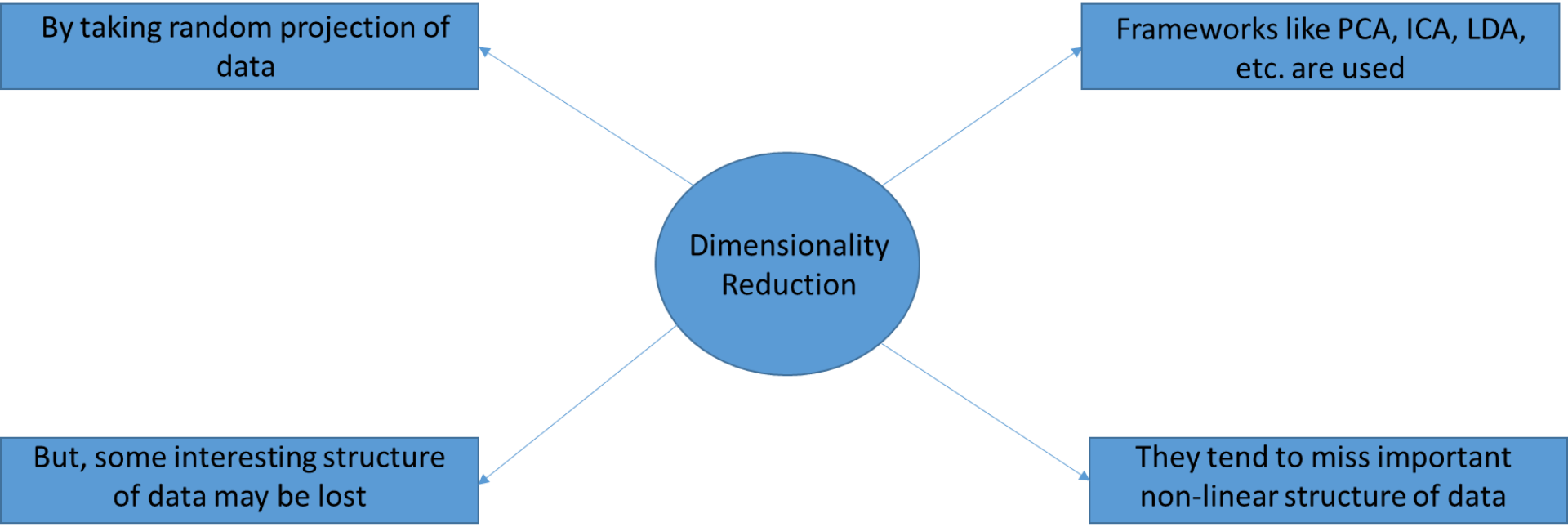
- **Similar to z-Score transformation, the distance from the observation to the location parameter is divided by the sample covariance**

- Usecases
  - Outlier detection
  - Classification
  - One-class classification

- Drawbacks
  - Needs grouped/ labelled data
  - Inverse of covariance matrix needed

Nearest k neighbors-manifold based: Isomap

to visualize the structure of data set



Manifold based dimensionality reduction tools

Mathematical space which on small scale resembles the Euclidean space of a specific dimension

An approach to non-linear dimensionality reduction

Used when linear models fail

Types: Multi-dimensional scaling, t-distributed Stochastic Neighbor Embedding (t-SNE), Isomap, etc.

Isometric Mapping

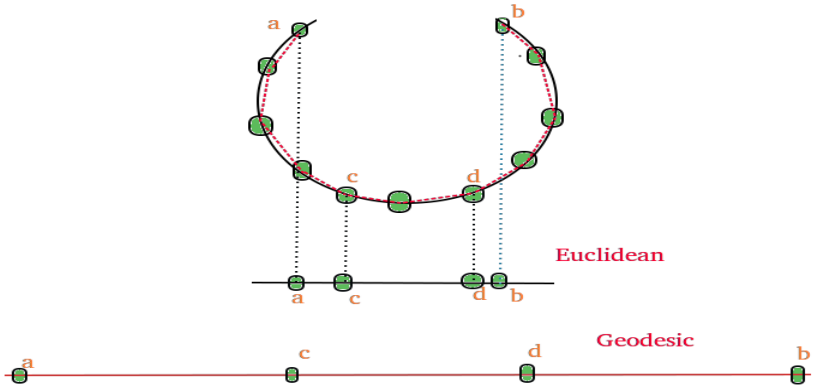
- Nearest neighbor search – Tries to create neighborhood network.
- Graph distance to the approximate geodesic distance between all pairs of points. (Distance between points along the manifold.)
- Through eigenvalue decomposition of the geodesic distance matrix, it finds the low dimensional embedding of the dataset
- Geodesic Distance - Number of edges in a shortest path connecting them is called the graph geodesic
- If we measure the distance between two points by following the manifold, we will have a better approximation of how far or near two points are.

Drawbacks of Isomap

- Isomap performs poorly when manifold is not well sampled and contains holes.
- Neighborhood graph creation is tricky and slightly wrong parameters can produce bad results.

Advantage

- Accurately captures the neighborhood relationships.



Isomap Algorithm

- Given input points $\{\mathbf{x}_i\}_{i=1}^n$, compute interpoint distances $\delta_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$
- Construct neighborhood graph, G, two possible ways:
  - edge $i, j \in G$ if i is is a K-nearest-neighbor of j
  - edge $i, j \in G$ if $\delta_{ij} < \epsilon$, ϵ some neighborhood size

  edge $i \leftrightarrow j$ gets weight δ_{ij}
- Compute shortest path distances d_{ij} between all pairs of nodes in G, using your favourite algorithm (Dijkstra, Floyd-Warshall)
- Use classical MDS on the shortest-path distances $\{d_{ij}\}$ to compute the images $\mathbf{y}_i$

Reference: http://www.cs.toronto.edu/~dross/411/16mds-4up.pdf

Thank you