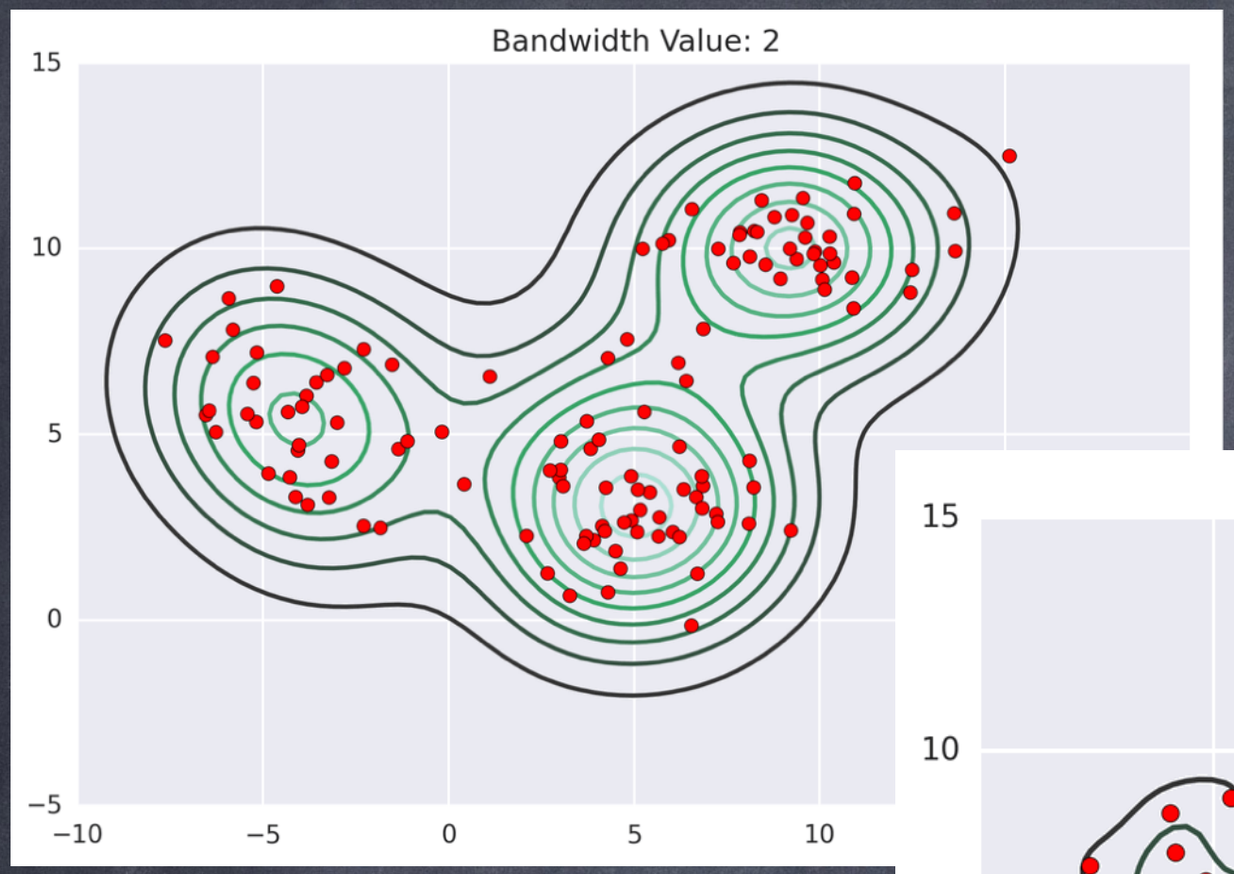
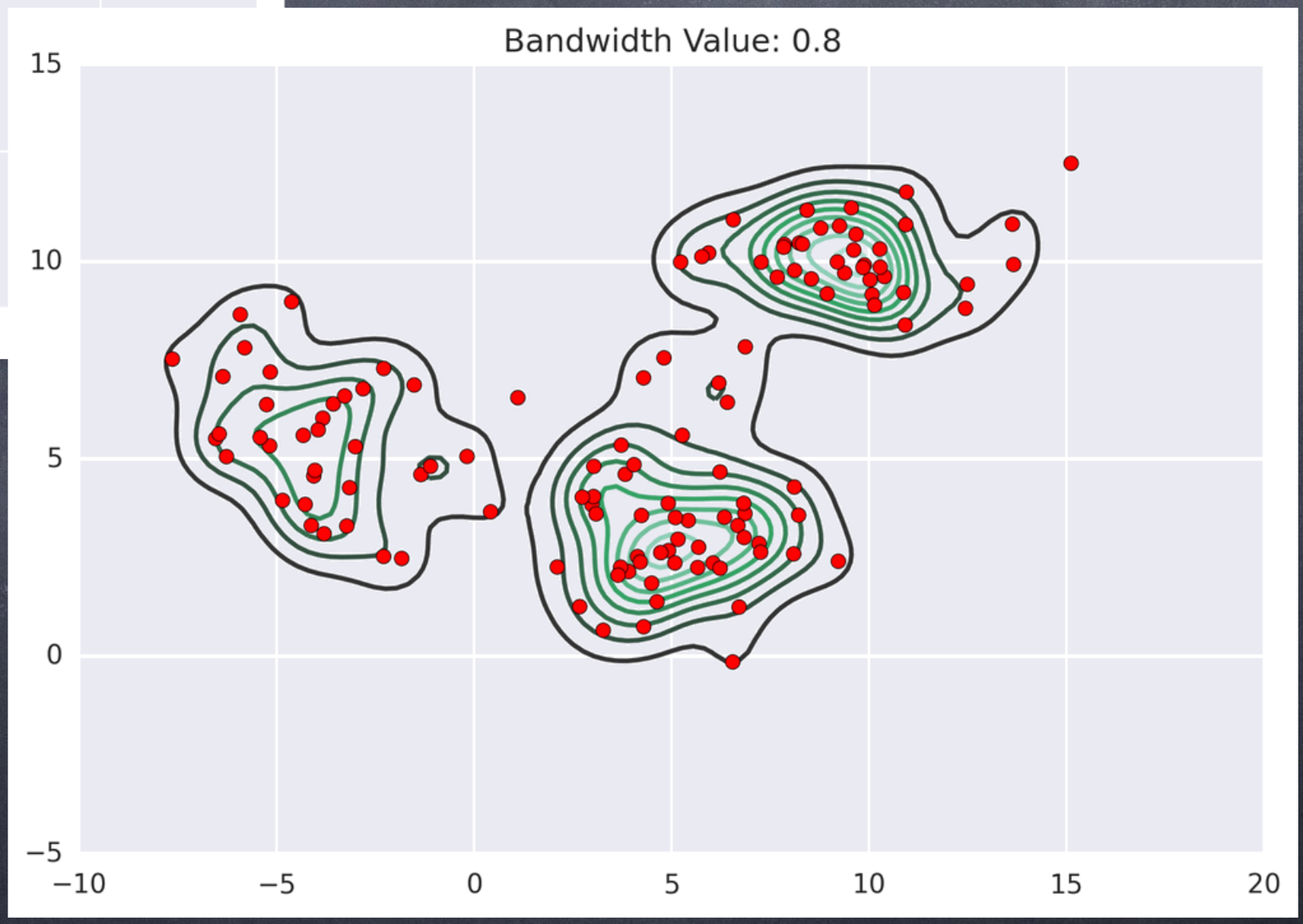
Machine Learning II
Week #3

Add on: Mean Shift Clustering method

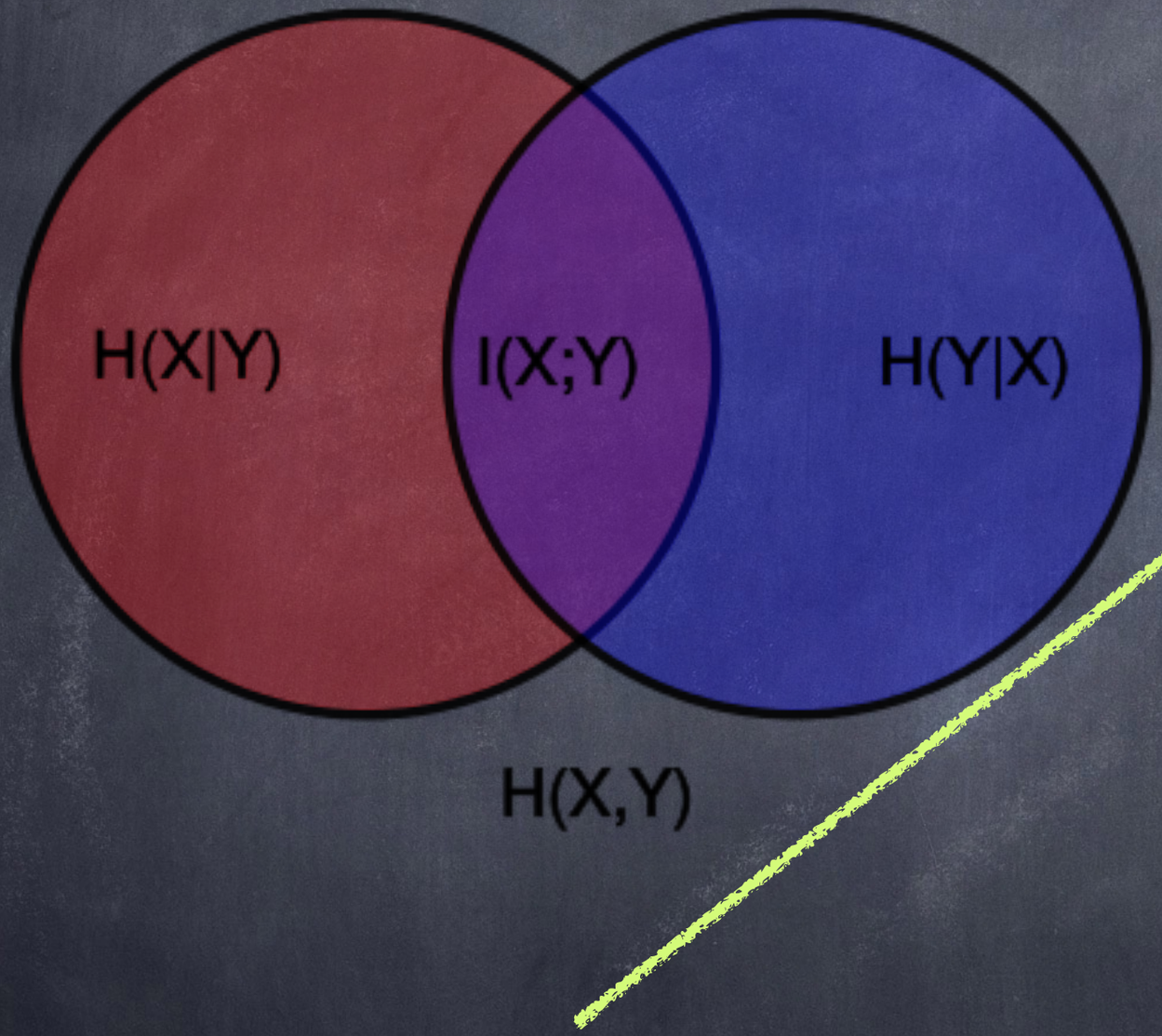




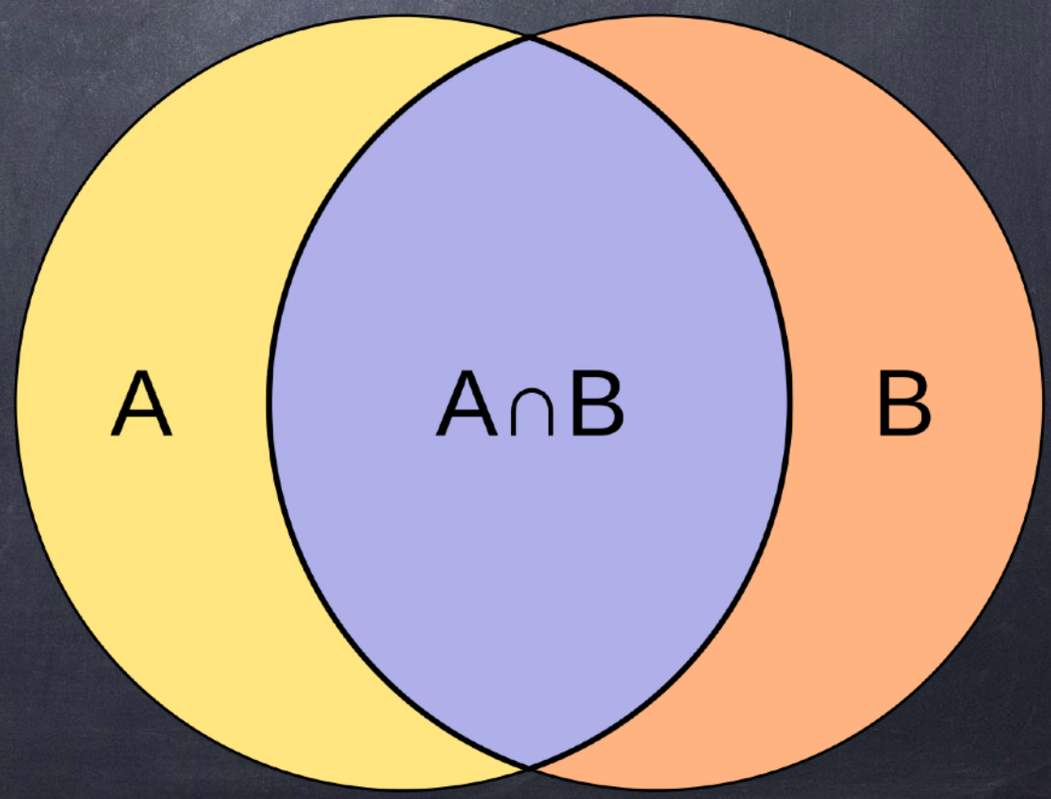
credit M. Nedrich

Add on: Mutual information (I) — Jaccard sim. (J)
entropy-based versus number-of-element based

$$I(X;Y) = H(X,Y) - H(X|Y) - H(Y|X)$$



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



Very different concepts!

No “silver bullet” approach to dimensionality reduction
or clustering

… but random projections
of high-dimensional data

Random projections: Johnson-Lindenstrauss lemma

For any data $X = \{x_1, x_2, \ldots x_n\}$ $\quad x_i \in \mathbb{R}^d$

Let $\varepsilon \in (0, \frac{1}{2})$

$k \geq C\varepsilon^{-2} \log n$

(for some constant C, e.g. C=20)

Then there is a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ such that

(f can be a matrix)

$$(1-\varepsilon)||x_i - x_j||_2^2 \leq ||f(x_i) - f(x_j)||_2^2 \leq (1+\varepsilon)||x_i - x_j||_2^2$$

for all pairs x_i, x_j where $||x_i - x_j||$ denotes the Euklidean norm

D. Achlioptas. 2003. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. Journal of Computer and System Sciences 66, 671 (2003)

Johnson-Lindenstrauss lemma

Every set of n points in a Euclidean space of any dimension can be "flattened" to dimension of only

$$O(\varepsilon^{-2} \log n)$$

in such a way that all distances between the points are preserved up to a factor between 1 – ε and 1 + ε

The dimensionality reduction (the map f) from d to k, with

$$k \sim \varepsilon^{-2} \log n$$

with the chosen error tolerance ε

w.h.p. does not change the squared L2 distance between any two points to more than $1 \pm \varepsilon$

Theoretical limitation:
This only holds for squared L2 norm not for L1 norm

Johnson Lindenstrauss lemma

Then there is a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ ....

ok!

But how to find this mapping ?

Answer: Random matrix A / Gaussian ensembles (python)

$$f = \frac{1}{k} A \in \mathbb{R}^{k \times d}$$

Random projections, wrong conclusions, e.g., at Scikit-Learn

Further reading:
Kane & Nelson paper (much sparser)
Fast Johnson-Lindenstrauss (next slide)

Warning:
Almost all secondary literature is wrong, typos, misconception,
even some papers are wrong, use the literature as given in this lecture

Warning

https://scikit-learn.org/stable/auto_examples/plot_johnson_lindenstrauss_bound.html

Remarks

According to the JL lemma, projecting 500 samples without too much distortion will require at least several thousands dimensions, irrespective of the number of features of the original dataset.

Hence using random projections on the digits dataset which only has 64 features in the input space does not make sense: it does not allow for dimensionality reduction in this case.

On the twenty newsgroups on the other hand the dimensionality can be decreased from 56436 down to 10000 while reasonably preserving pairwise distances.

Random projections, further reading

Johnson, William B.; Lindenstrauss, Joram (1984). Extensions of Lipschitz mappings into a Hilbert space". In Beals, Richard; Beck, Anatole; Bellow, Alexandra; et al. (eds.). Conference in modern analysis and probability (New Haven, Conn., 1982). Contemporary Mathematics. 26. Providence, RI: American Mathematical Society. pp. 189–206. doi:10.1090/conm/026/737400.

Dasgupta, Sanjoy; Gupta, Anupam (2003), An elementary proof of a theorem of Johnson and Lindenstrauss", Random Structures & Algorithms, 22 (1): 60–65, doi:10.1002/rsa.10073

Ailon, Nir; Chazelle, Bernard (2006). Approximate nearest neighbors and the Fast Johnson–Lindenstrauss transform. Proceedings of the 38th Annual ACM Symposium on Theory of Computing. New York: ACM Press. pp. 557–563. doi:10.1145/1132516.1132597

Ping Li, Trevor J. Hastie, and Kenneth W. Church. 2006. Very sparse random projections. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '06). ACM, New York, NY, USA, 287-296

Kane, Daniel M.; Nelson, Jelani (2014). "Sparser Johnson-Lindenstrauss Transforms". Journal of the ACM. 61 (1), arXiv:1012.1577. doi:10.1145/2559902

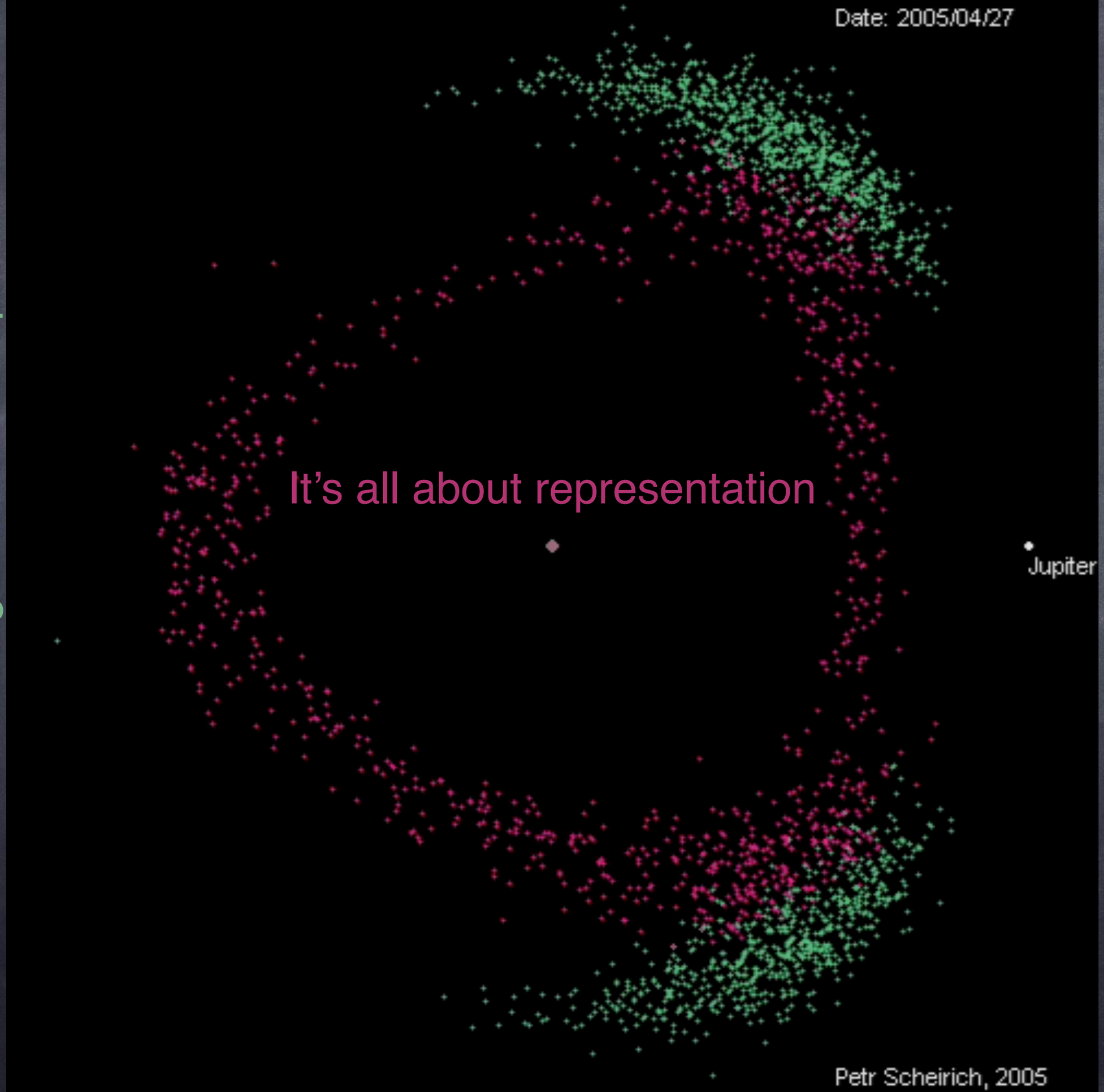
Kernel Methods

It’s all about representation

Heliocentrism

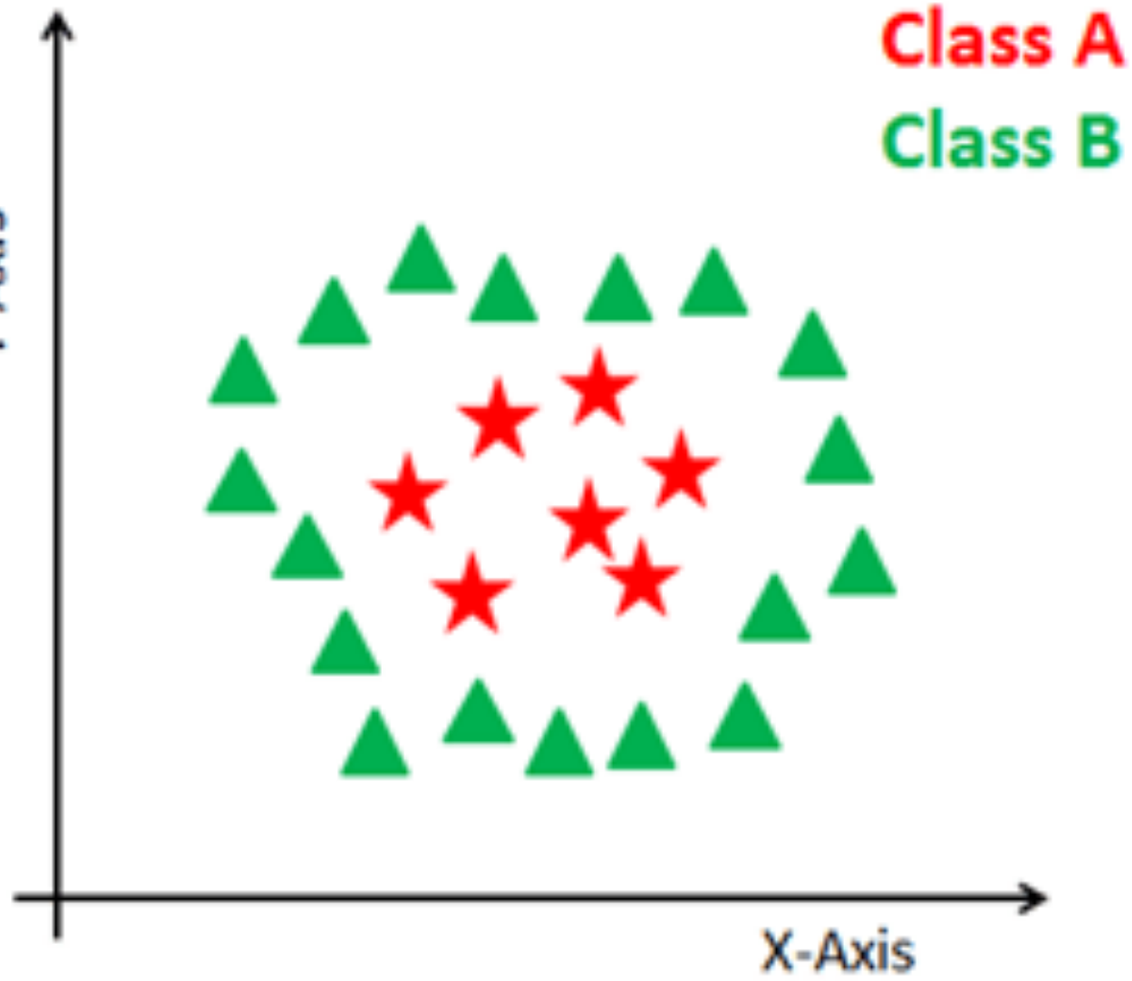
Geocentrism

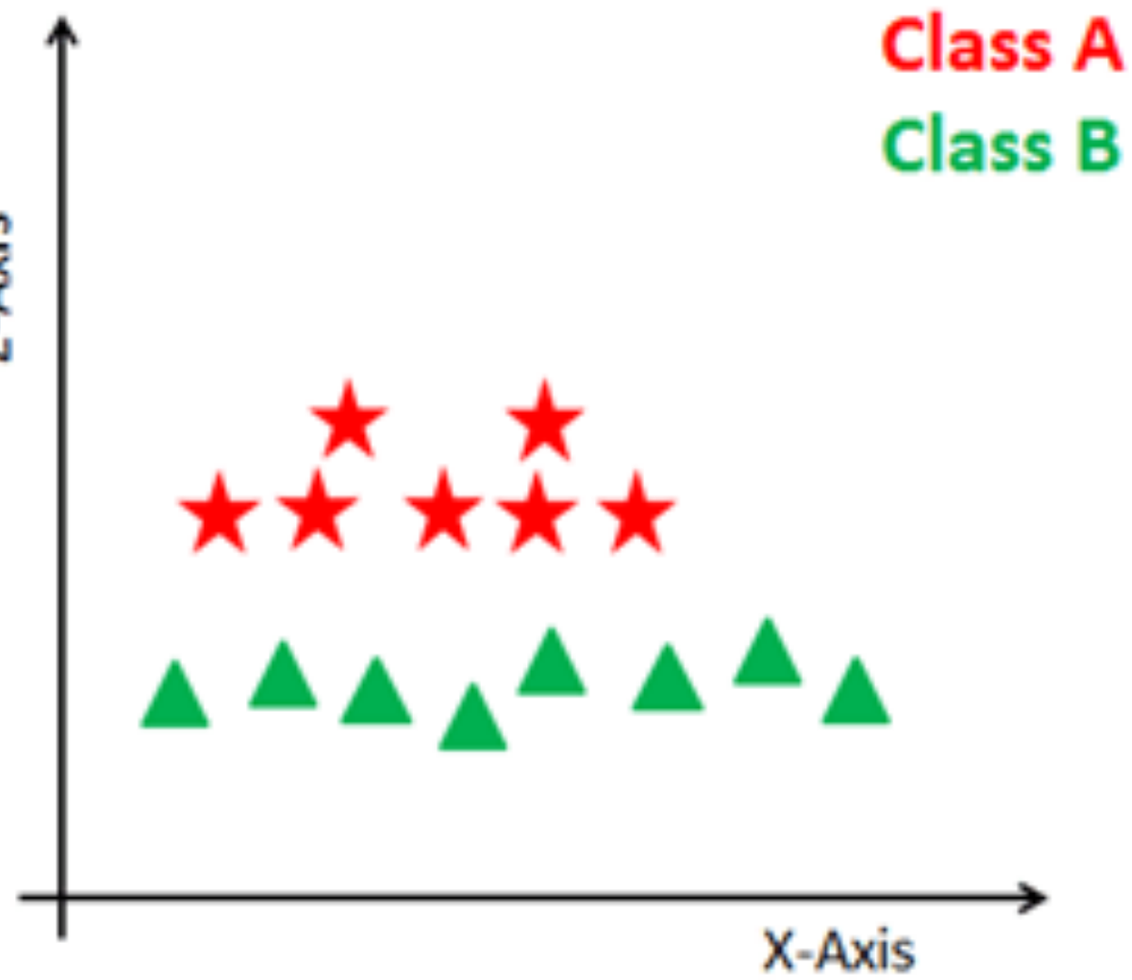
Corotating frame Sun-Jupiter



It’s all about representation

Representation matters: Kernel Methods



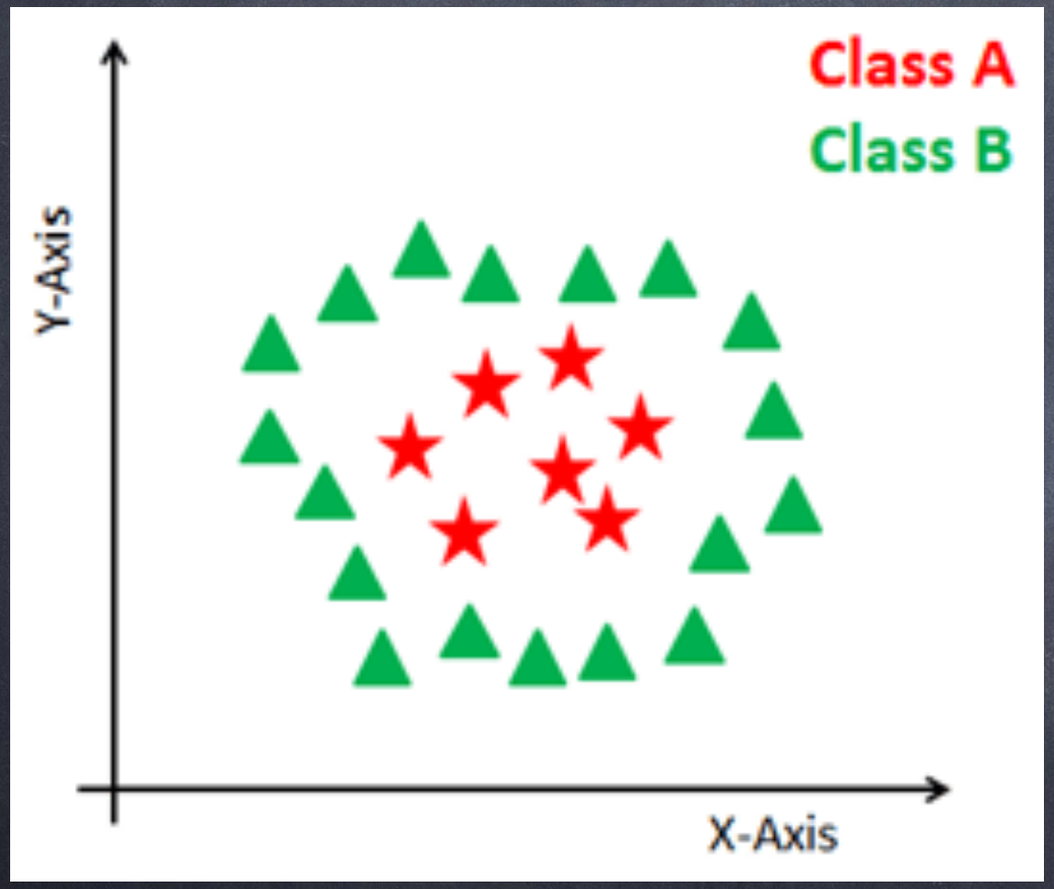


$$(x, y) \rightarrow (r, \varphi)$$

$$x = r\cos(\varphi)$$
$$y = r\sin(\varphi)$$

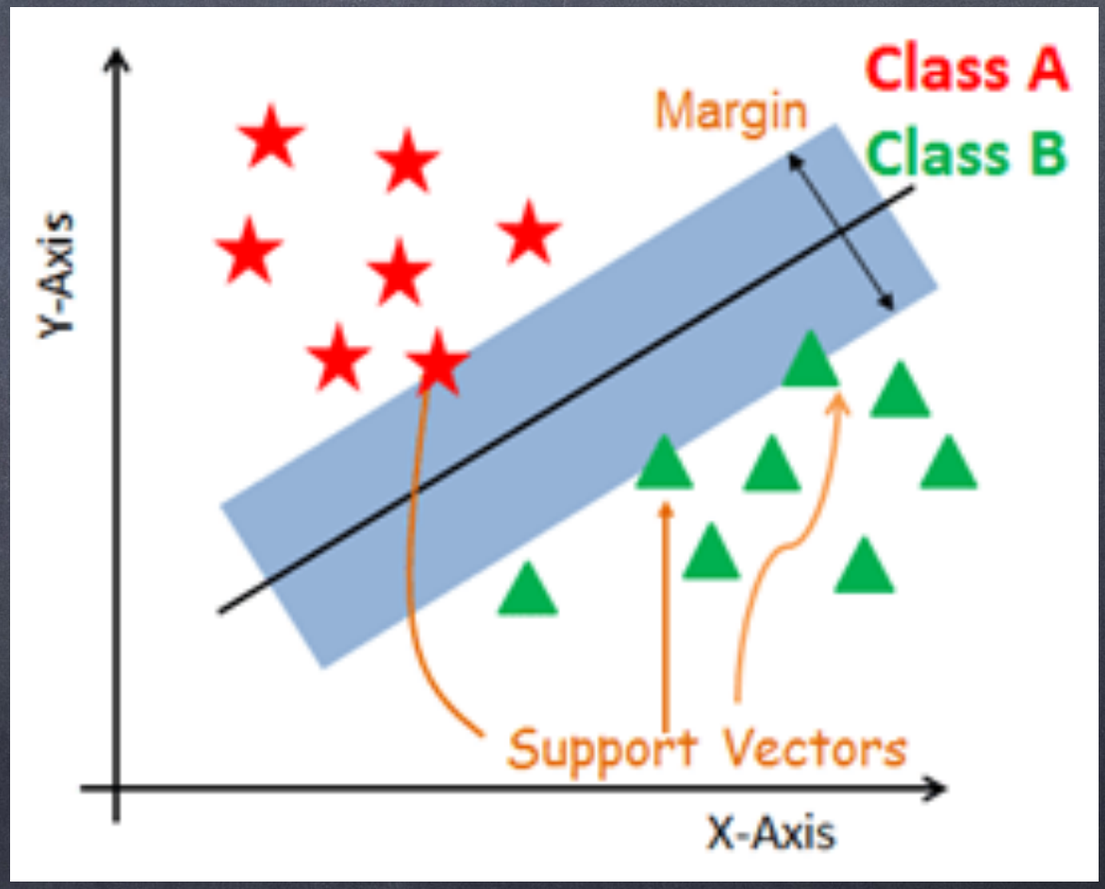
Nonlinear kernel Support Vector Machines

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



Linearly inseparable

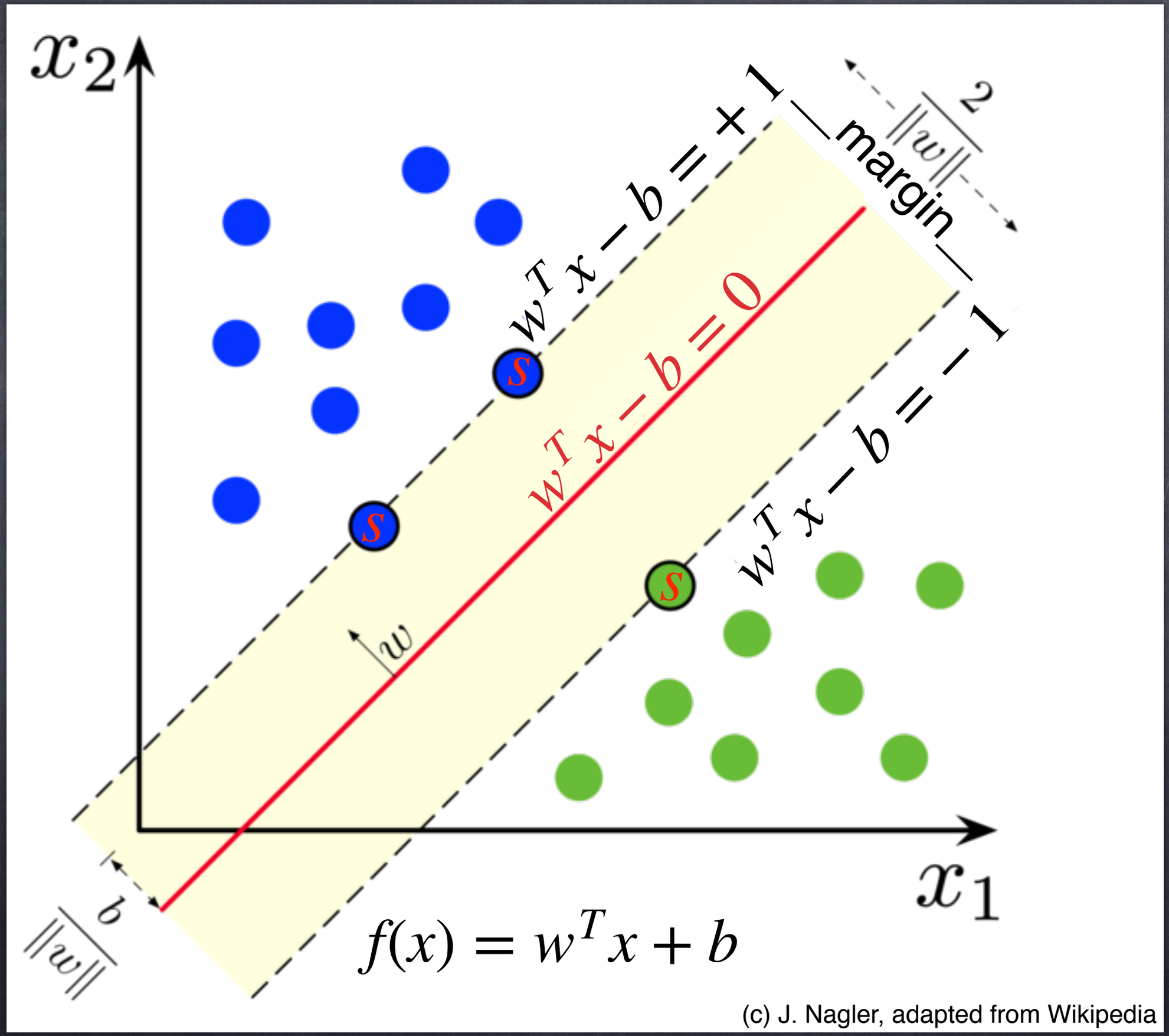
Map



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

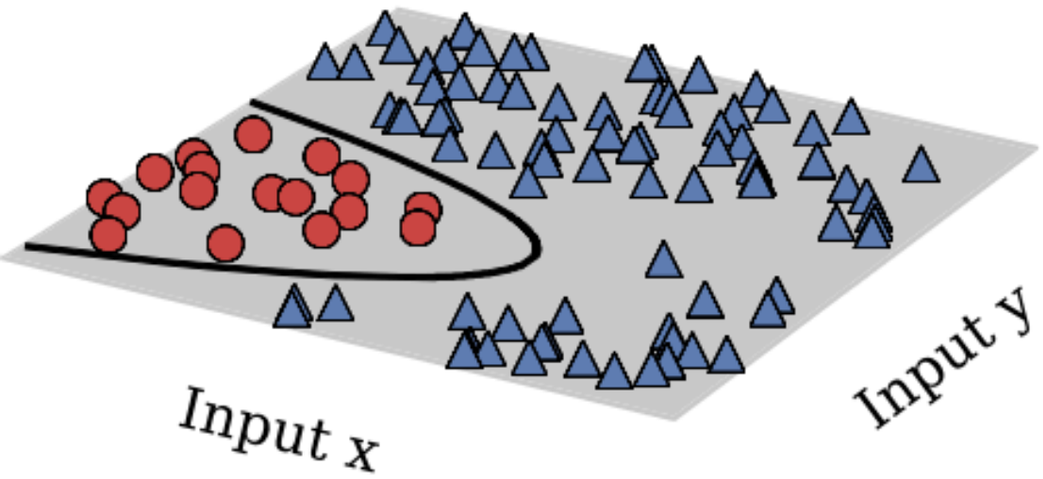
w, b : from support vectors

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

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

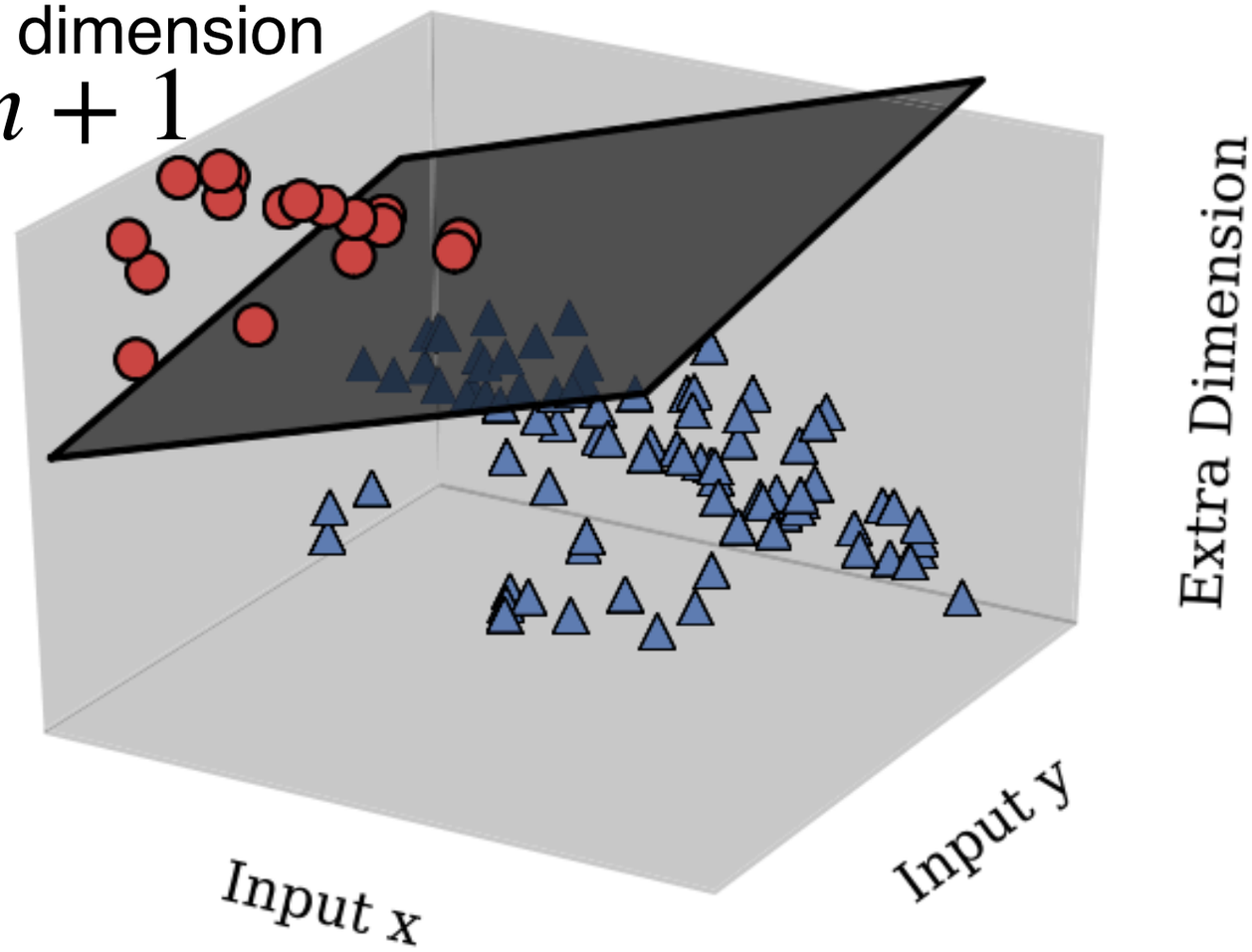
No extra dimension
but nonlinear feature map

$m = n$



With extra dimension

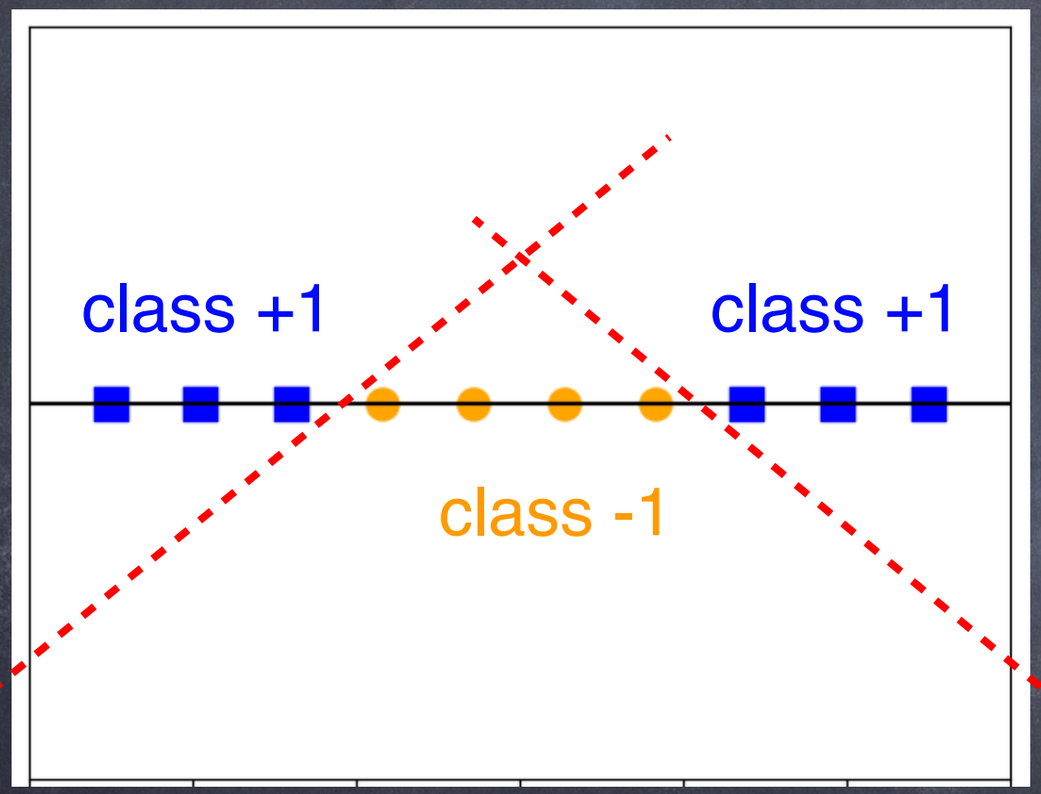
$m = n + 1$



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

Kernel Trick: Square it example #1

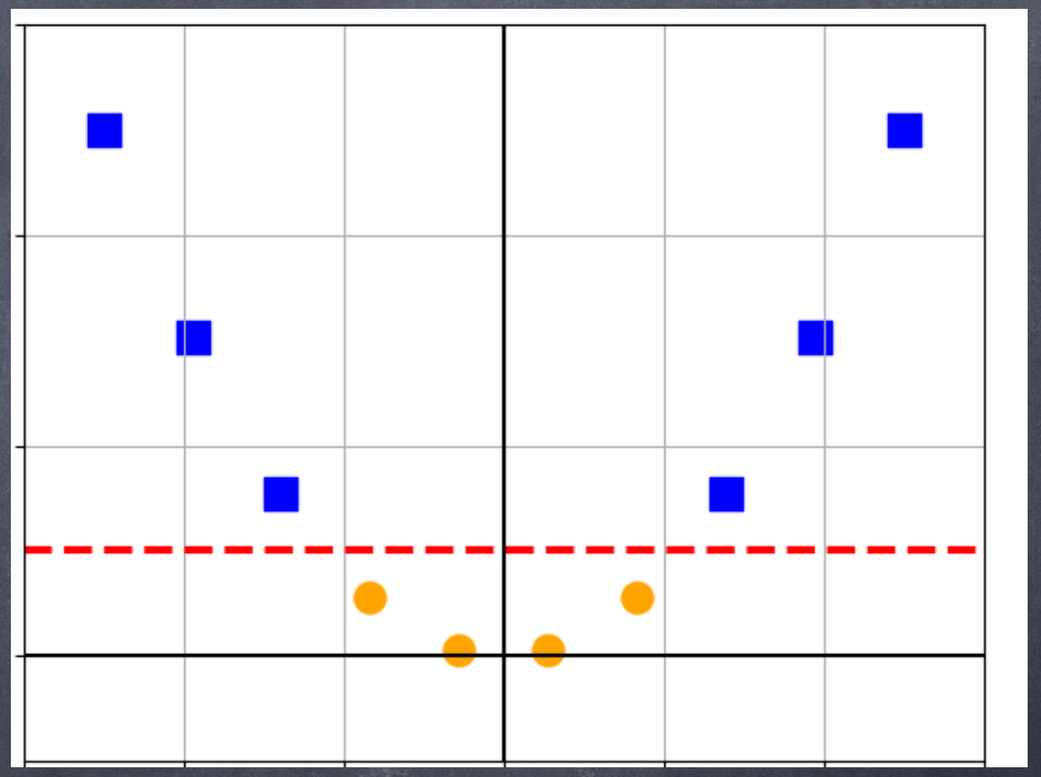
1d



Single linear classifier in x
does not exist

2d

extra dimension sqr(x)



Red line: Linear classifier in [x, phi(x)]

Kernel Trick

Example #2: Quadratic 2d->6d kernel (iPad)

$$\phi(\mathbf{x})\phi(\mathbf{z}) = 1 + \mathbf{x}^{\mathrm{T}}\mathbf{z}$$

Example #3: 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 (here name x'=z) iPad

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

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

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)

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

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

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

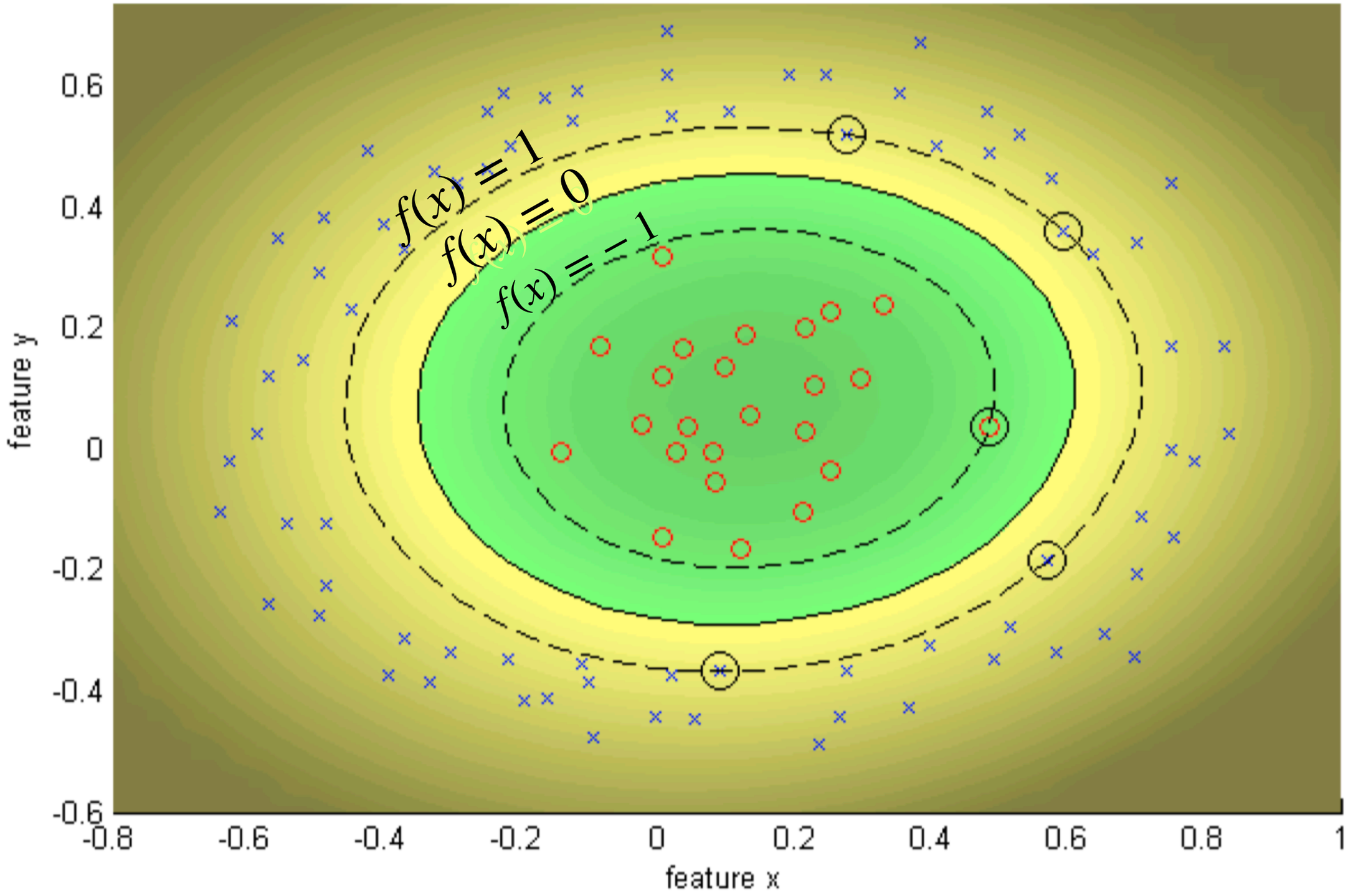
(No finite superposition of Gaussians to build target Gaussian) for $\gamma = 1/2$

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

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

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

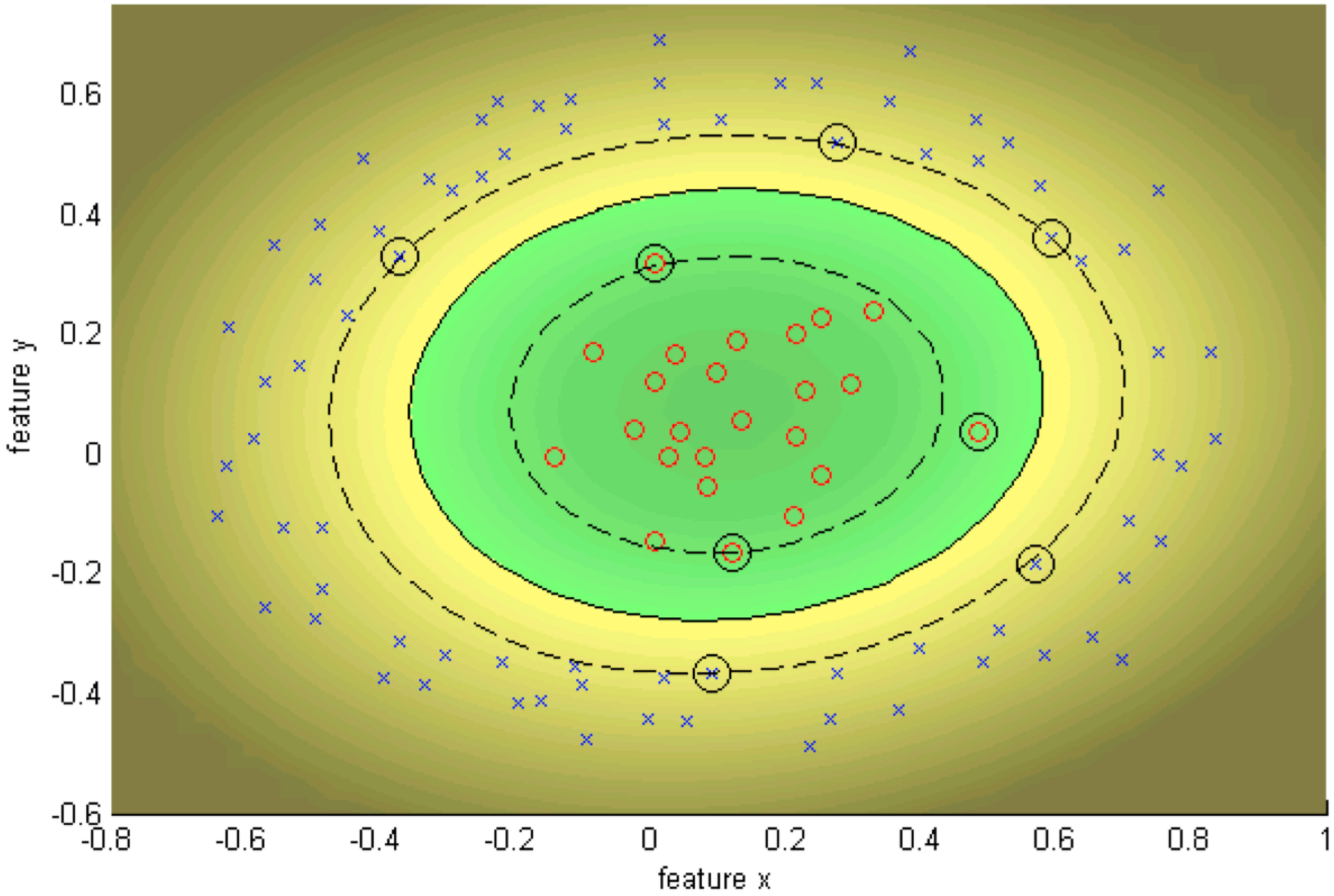
nonlinear decision boundaries



$\sigma = 1.0$ $C = 100$

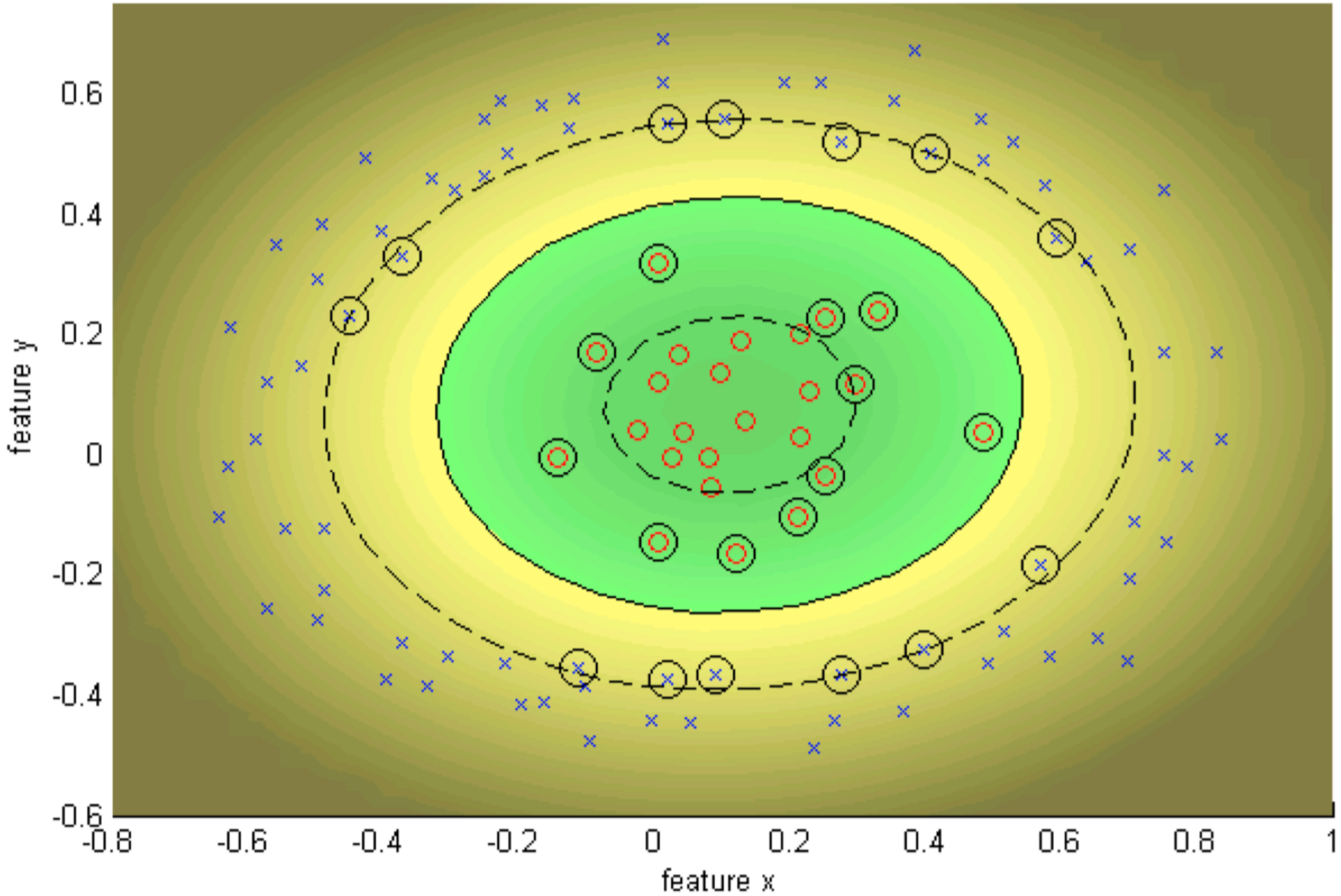
C: Soft margin parameter

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

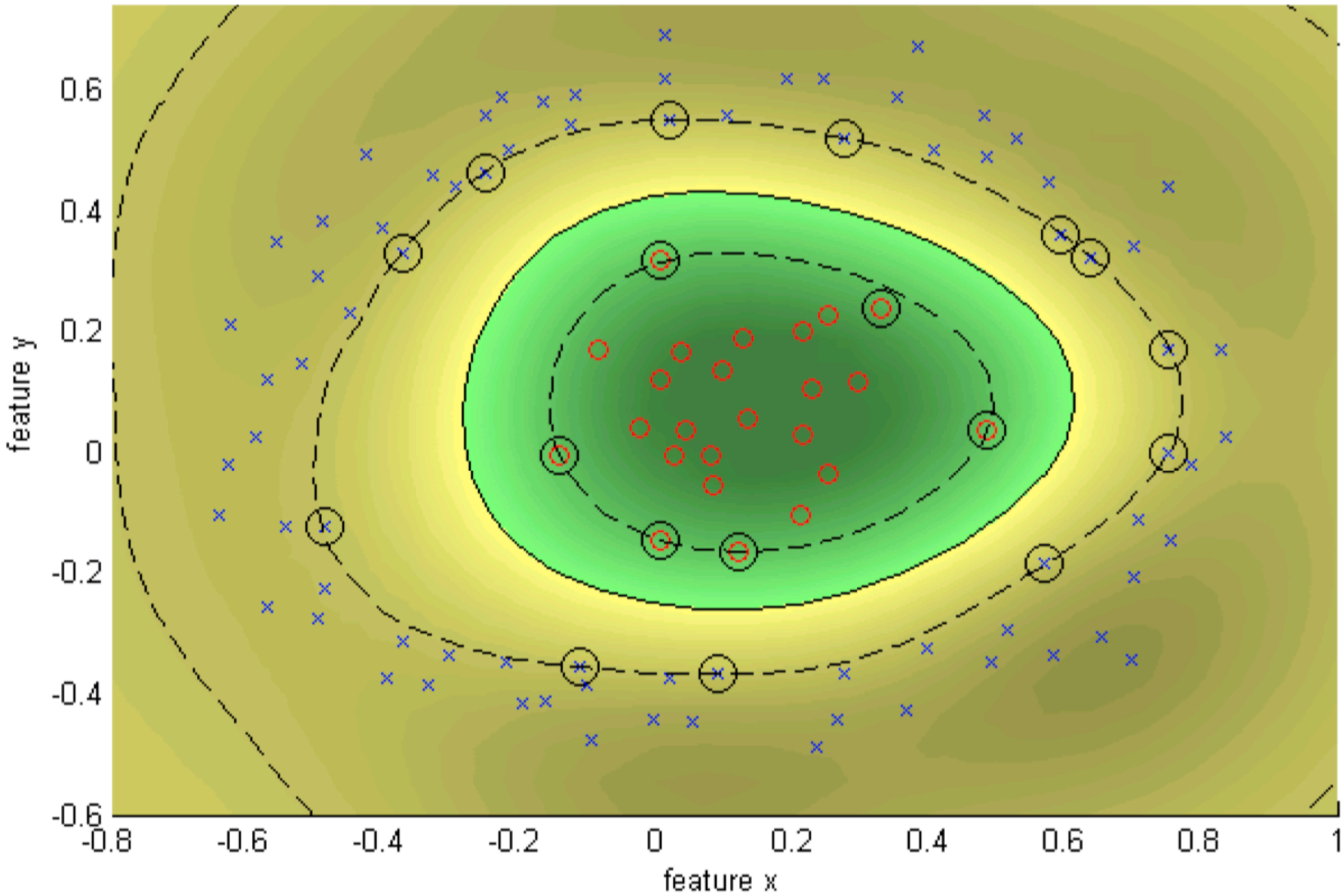
$$\gamma = 1/(2\sigma^2)$$

Omega: Vice versa

C: Soft margin parameter (as shown)

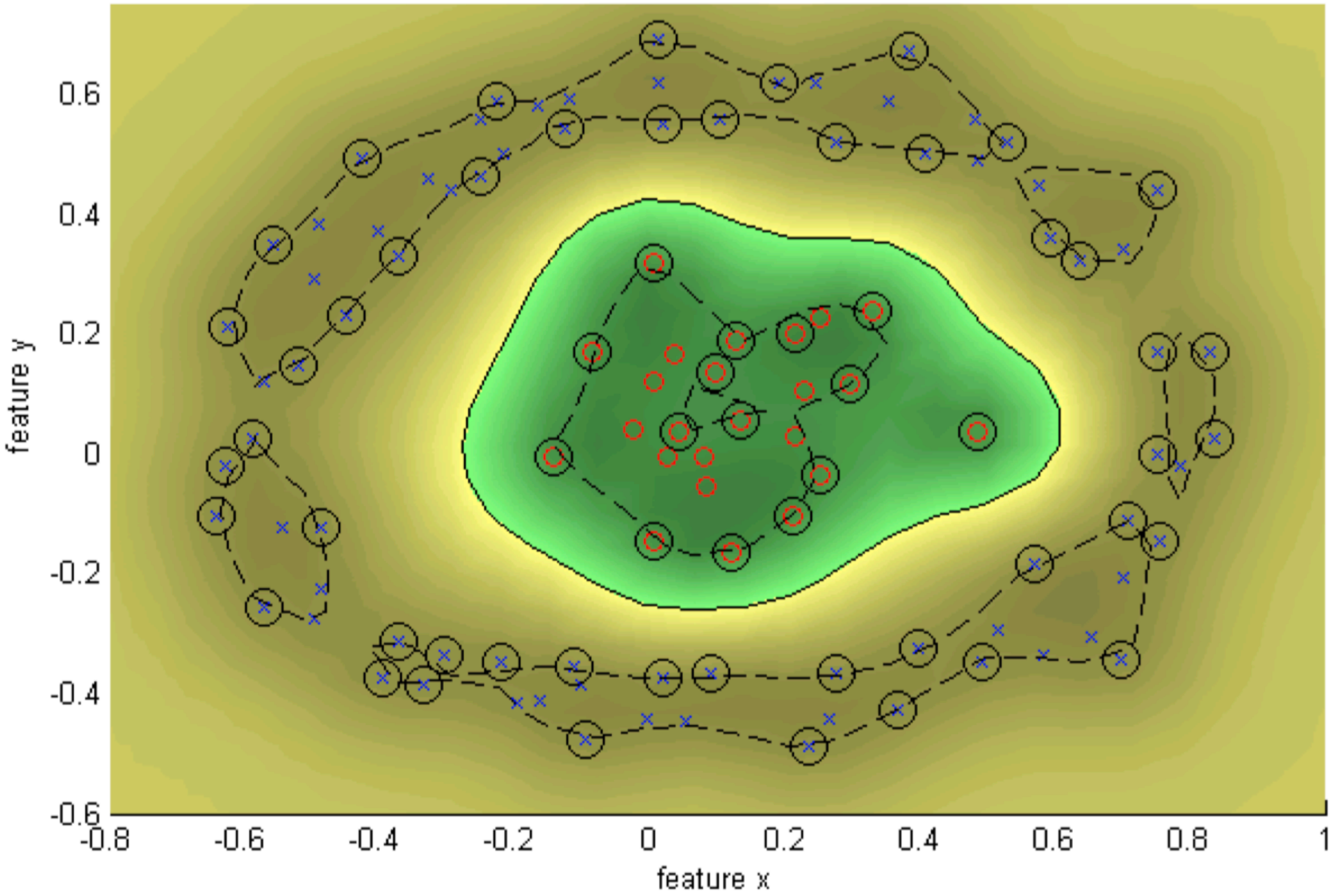
$\sigma = 0.25 \quad C = \infty$

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:

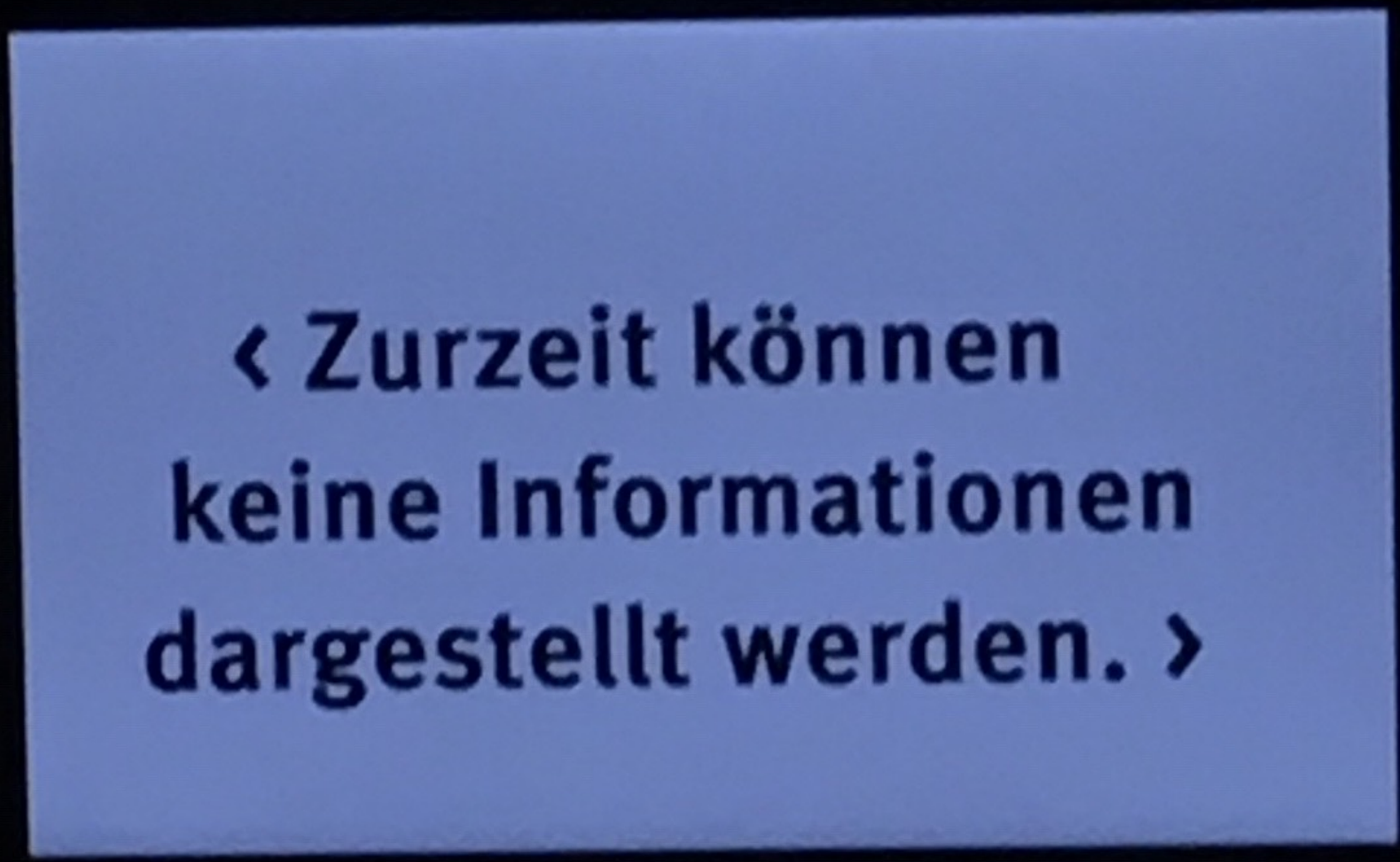
Slow for large datasets because of its high training time

Works poorly with overlapping classes

Sensitive to the type of kernel used

Expectation Maximization (EM)

Expectation Maximization



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

Expectation Maximization (EM)

Sort of soft clustering approach, based on information theory

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])

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

Expectation Maximization (EM)

Goal: Maximize log likelihood $\theta_{MLE} = \texttt{argmax}_\theta \log p(x \mid \theta)$

(instead of maximize pdf itself,
avoid problem of underflow small prob by using log,
analytically: easier to handle sums than products)

Problem: Hard to do in closed form, e.g., many local maxima,
gradient methods, Newton-based maybe hard to compute or to implement

Expectation Maximization (EM)

Theory based on variational methods,
i.e., free distribution is introduced for obtaining
lower bound error in optimization step

Advantages:

Conceptual simplicity, simple implementation, each iteration leads to improvement

Improvement (rate of convergence) fast in the beginning

EM works best when the fraction of missing information is small

Disadvantages:

EM can require many iterations
Higher dimensionality may substantially
slow down the E-step and reduce improvement

Expectation Maximization (EM)

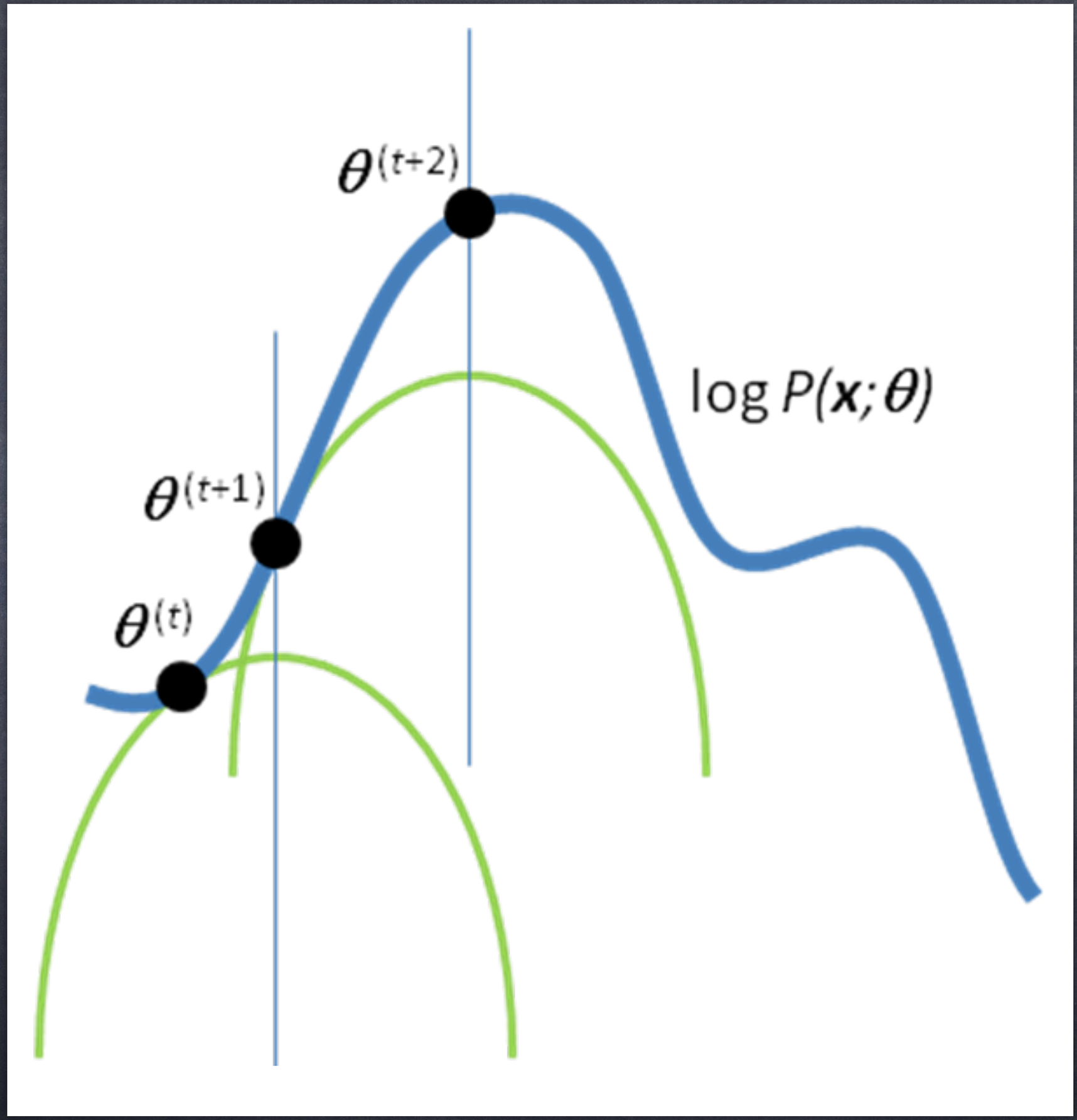
#0: General theoretical framework

Example #1:
Coin toss (Primer pdf + iPad)

Example #2:
Gaussian Mixure Model

iPad

Expectation Maximization



Primer pdf

Expectation Maximization (EM) — Coin flips — M Step

$$Q = Q_i(z_i) := p(z_i, x_i \,|\, \theta)$$

$$= E\left[\log \Pi_{i \le n} [p_A \; p^{x_i} (1-p)^{1-x_i}]^{z_i} [p_B \; q^{x_i} (1-q)^{1-x_i}]^{1-z_i}\right]$$

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

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

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

iPad / exercise

Expectation Maximization

EM is a statistical inference method

Relations between

Maximum Likelihood Estimation (MLE)

Maximum A posteriori (MAP) estimate

and Expectation-Maximization (EM) algorithm

iPad / exercise / Gaussian mix: next week


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.

Regularization

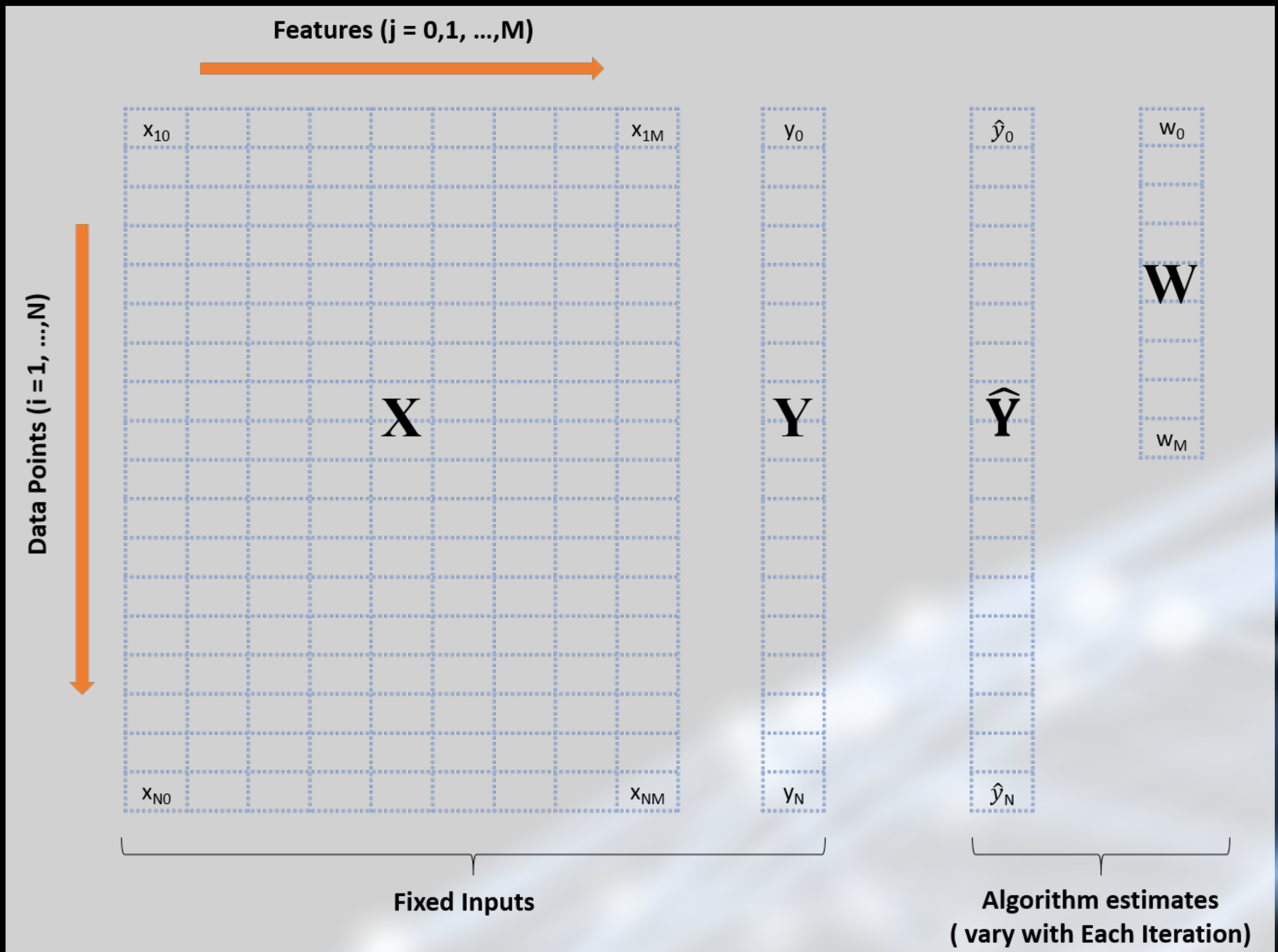
Ridge Regression:
Performs L2 regularization,
i.e. adds penalty equivalent to square of the magnitude of coefficients,
with
minimization objective =
sum of squared residuals + factor * (sum of square of coefficients)

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 nonlinear optimization (quadratic programming yields efficiently approximation)

LASSO has no closed-form solution (nonlinearity)

LASSO

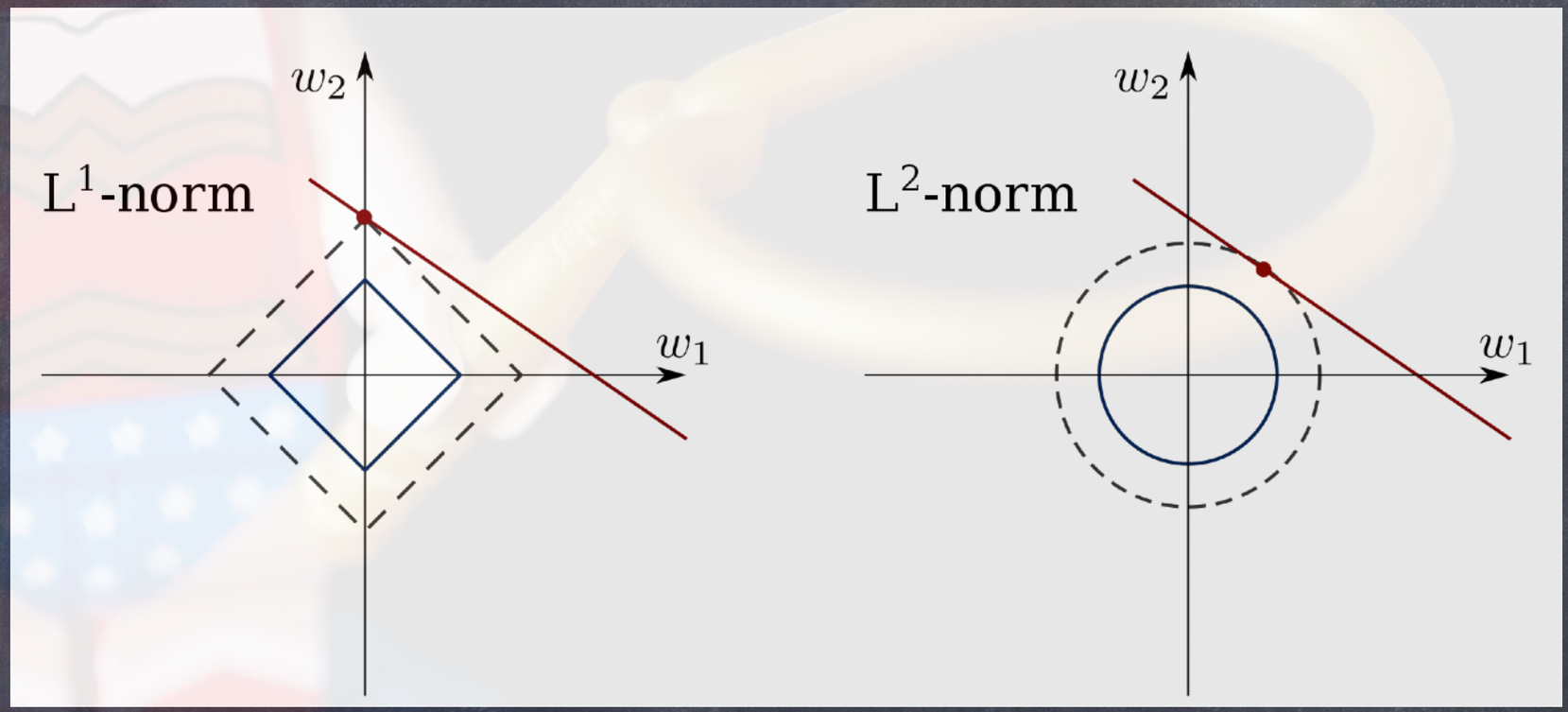
hard thresholding

$|w_1|+|w_2|< const$

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!

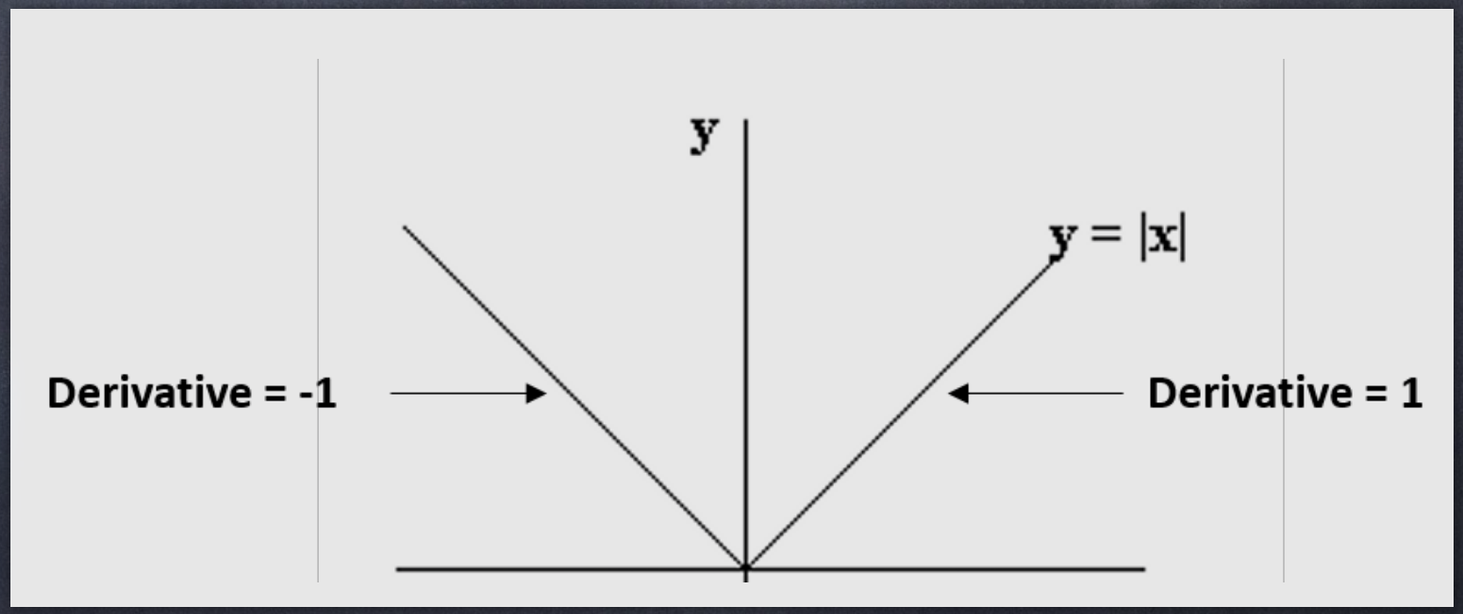
Lambda choice for ridge:
Information criterion or
min. prediction error / CV,
omega never zero

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?



iPad, theory, proof (live, similar to next 4 slides)

Proof idea : const $\longleftrightarrow$ λ

$$\min_{\underline{\omega}} \| \underline{y} - \underline{\underline{X}}\,\underline{\omega} \|^2 , \quad \|\underline{\omega}\|_2^2 \leq C = \text{const.}$$

Solution

$$g(\lambda) = \inf_{\underline{\omega}} \mathcal{L}(\underline{\omega}, \lambda)$$

Lagrange Method

$$= \inf_{\underline{\omega}} \left[\| \underline{y} - \underline{\underline{X}}\,\underline{\omega} \|^2 + \lambda \left(\|\underline{\omega}\|_2^2 - C \right) \right]$$

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

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

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

$$g(\lambda) = \| y - \underline{\underline{X}} (\underline{\underline{X}}^T \underline{\underline{X}} + \lambda \underline{\underline{I}})^{-1} \underline{\underline{X}}^T \underline{y} \|^2$$

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

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

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

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

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

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

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

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

Closed form solution of Ridge Problem

Elastic Net

LASSO — many variables useless

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

ssr + L1 penalty

Ridge — most variables useful

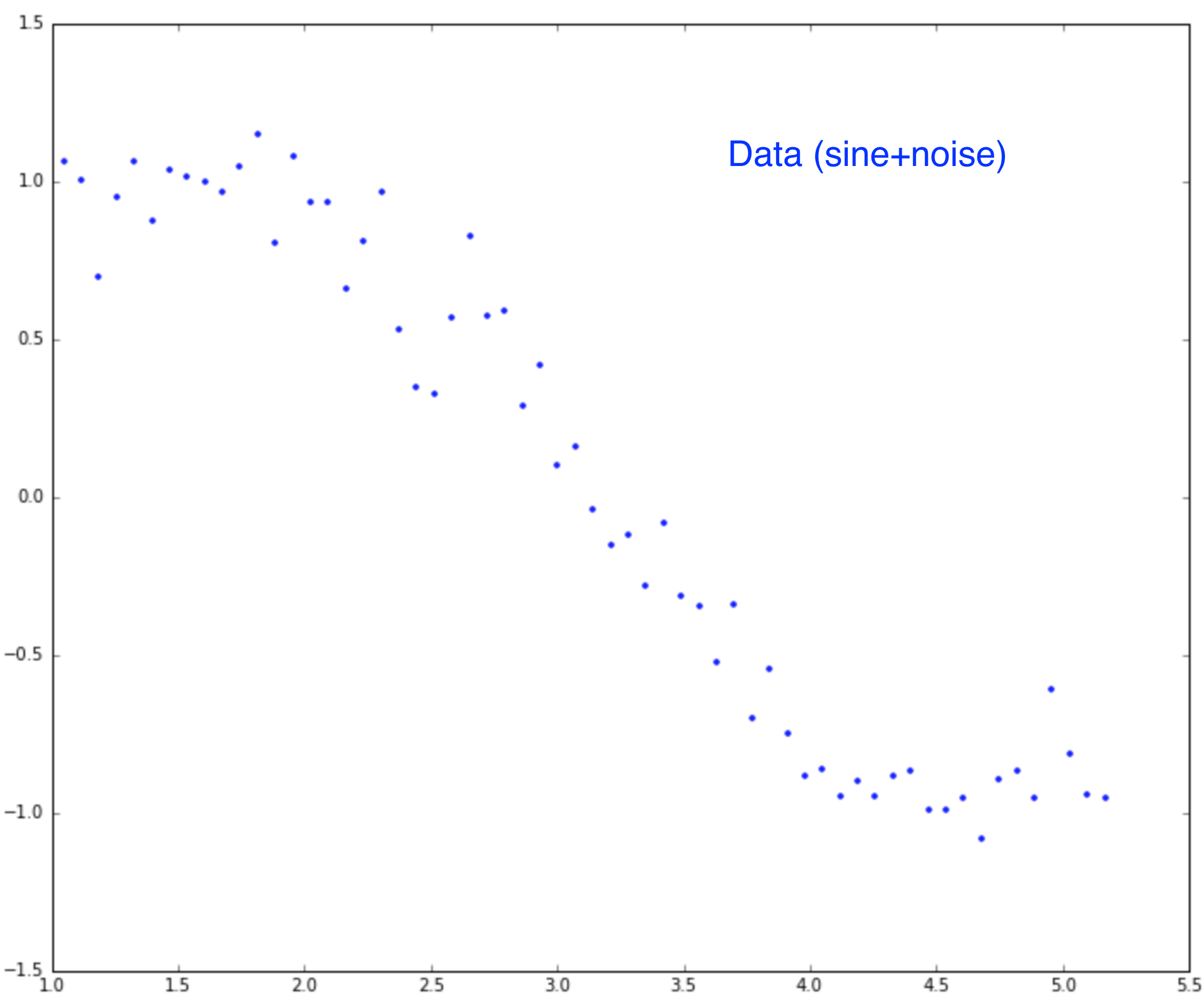
$$|w_1|^2+|w_2|^2< const$$

ssr + L2 penalty

makes Elastic Net

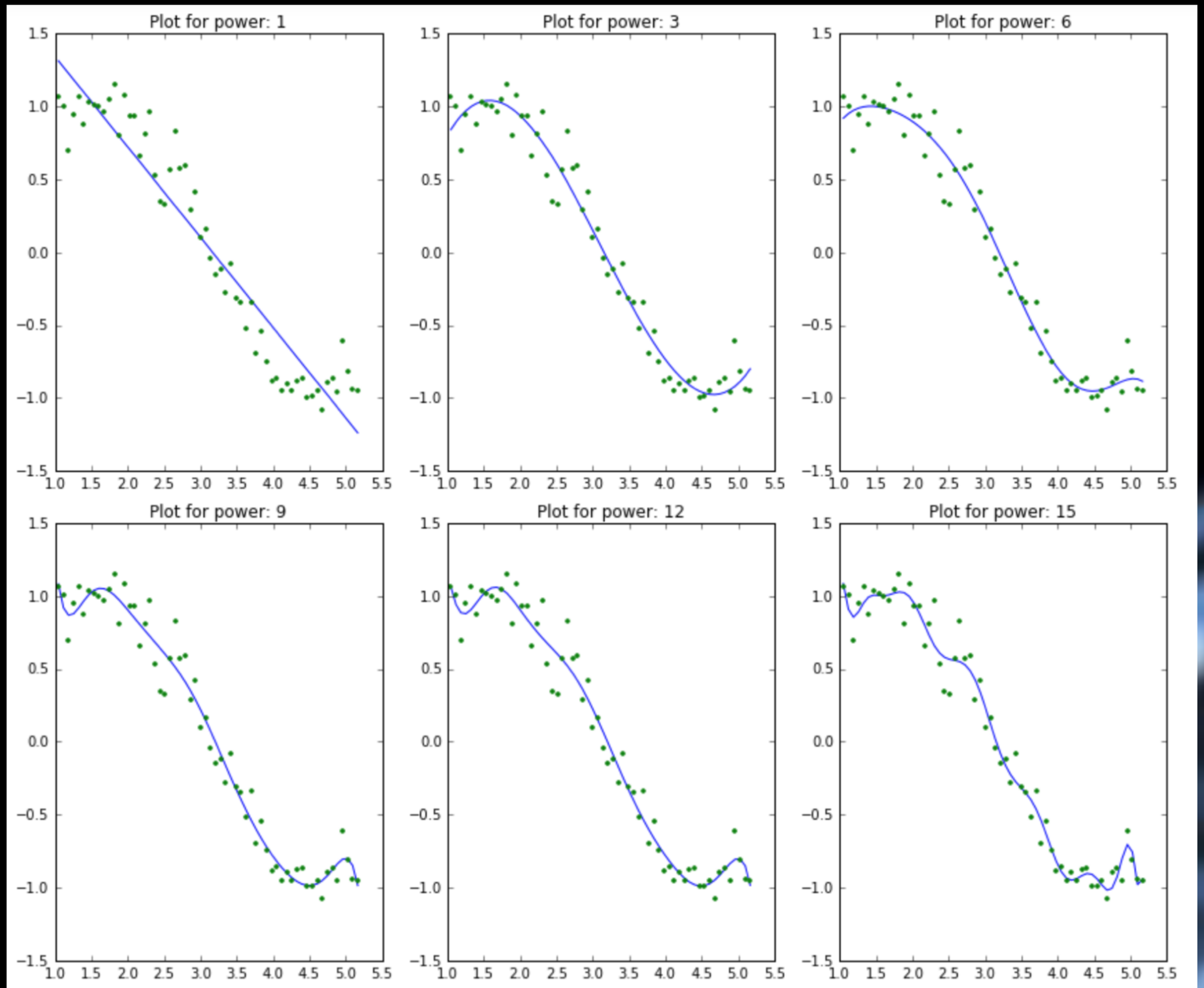
$$\sum_{i=1}^{N}\left[y_i-\sum_{j=0}^{M}w_j x_{ij}\right]^2+\lambda_1\sum_{j=0}^{M}|w_j|+\lambda_2\sum_{j=0}^{M}w_j^2$$

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)

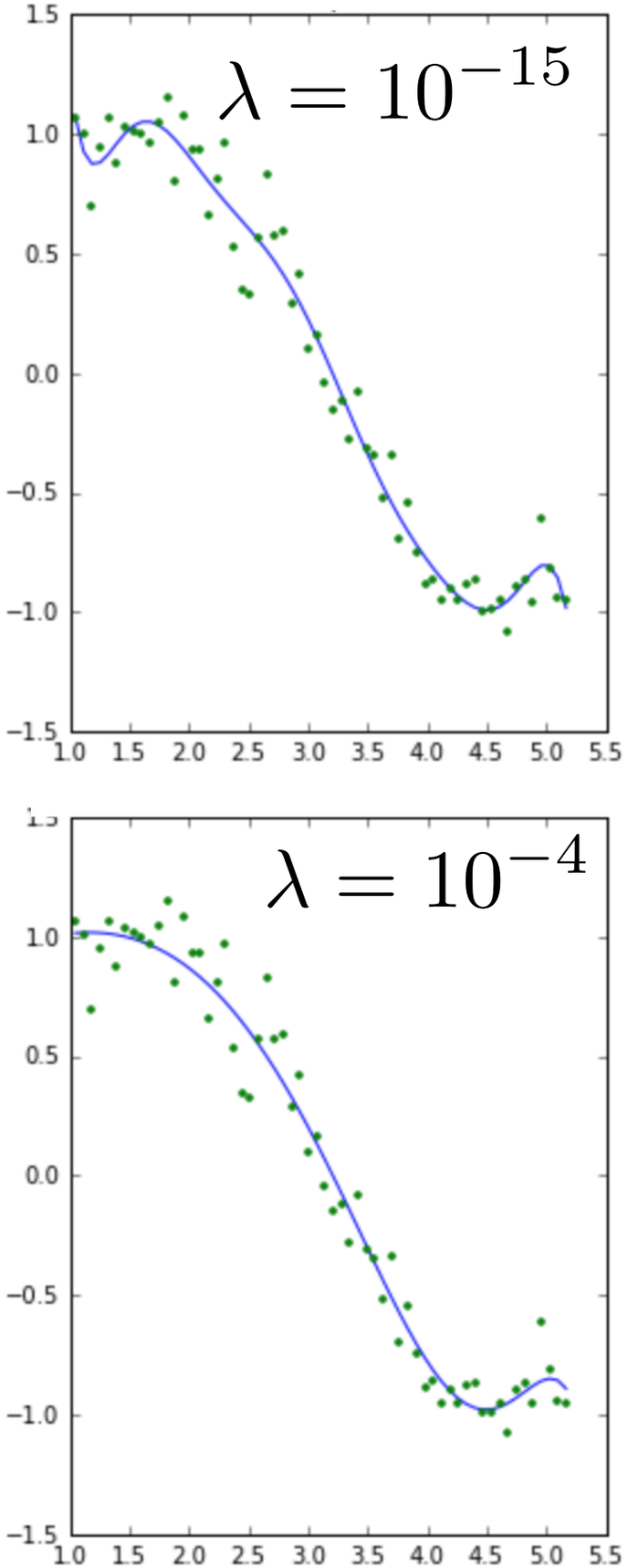
	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	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

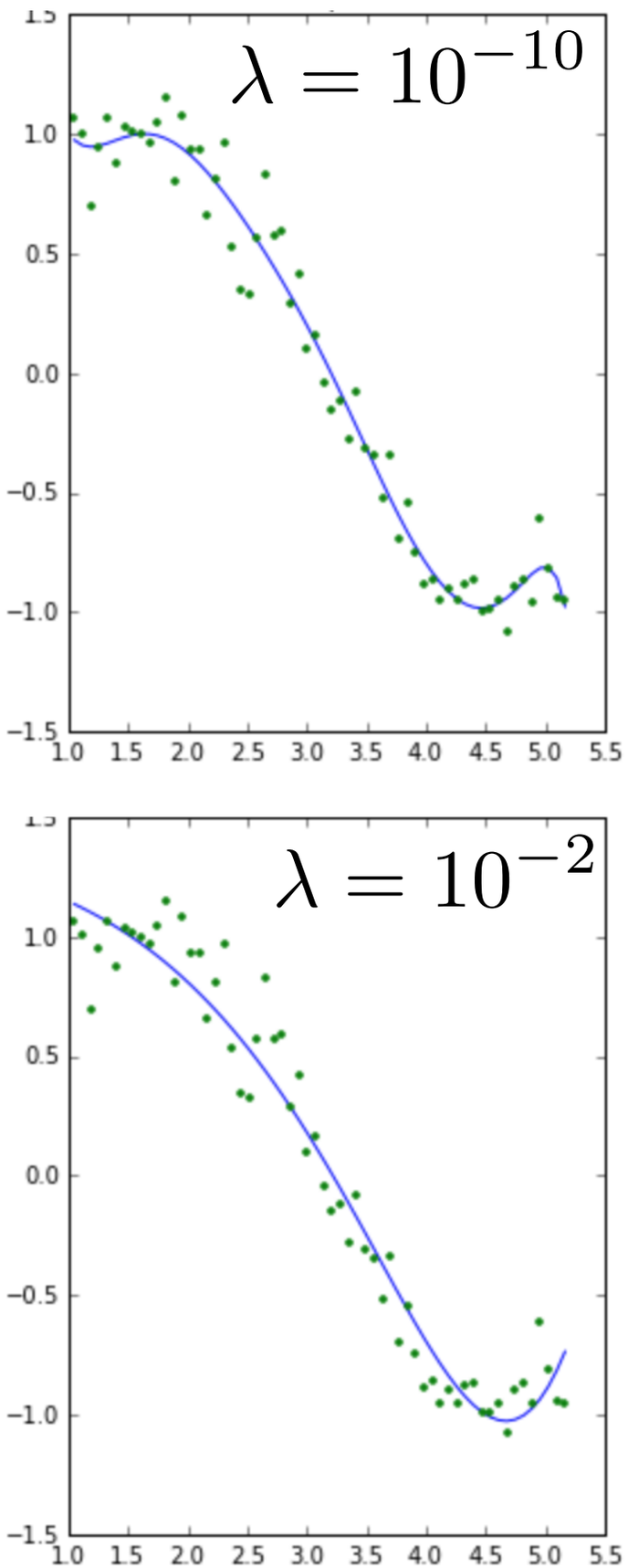
Coefficients increase exponentially with model complexity!

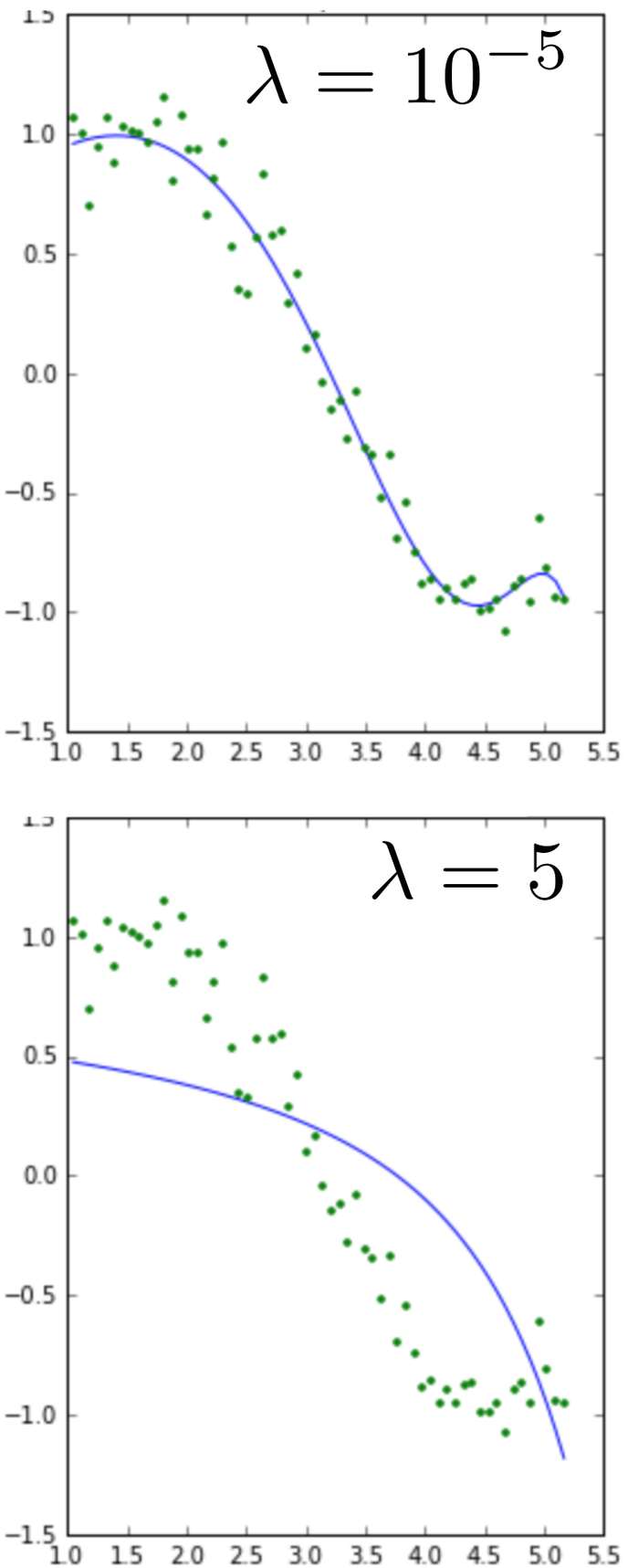
Regularization needed!

Courtesy: K. Jain

Ridge regression (regularization)







Courtesy: K. Jain

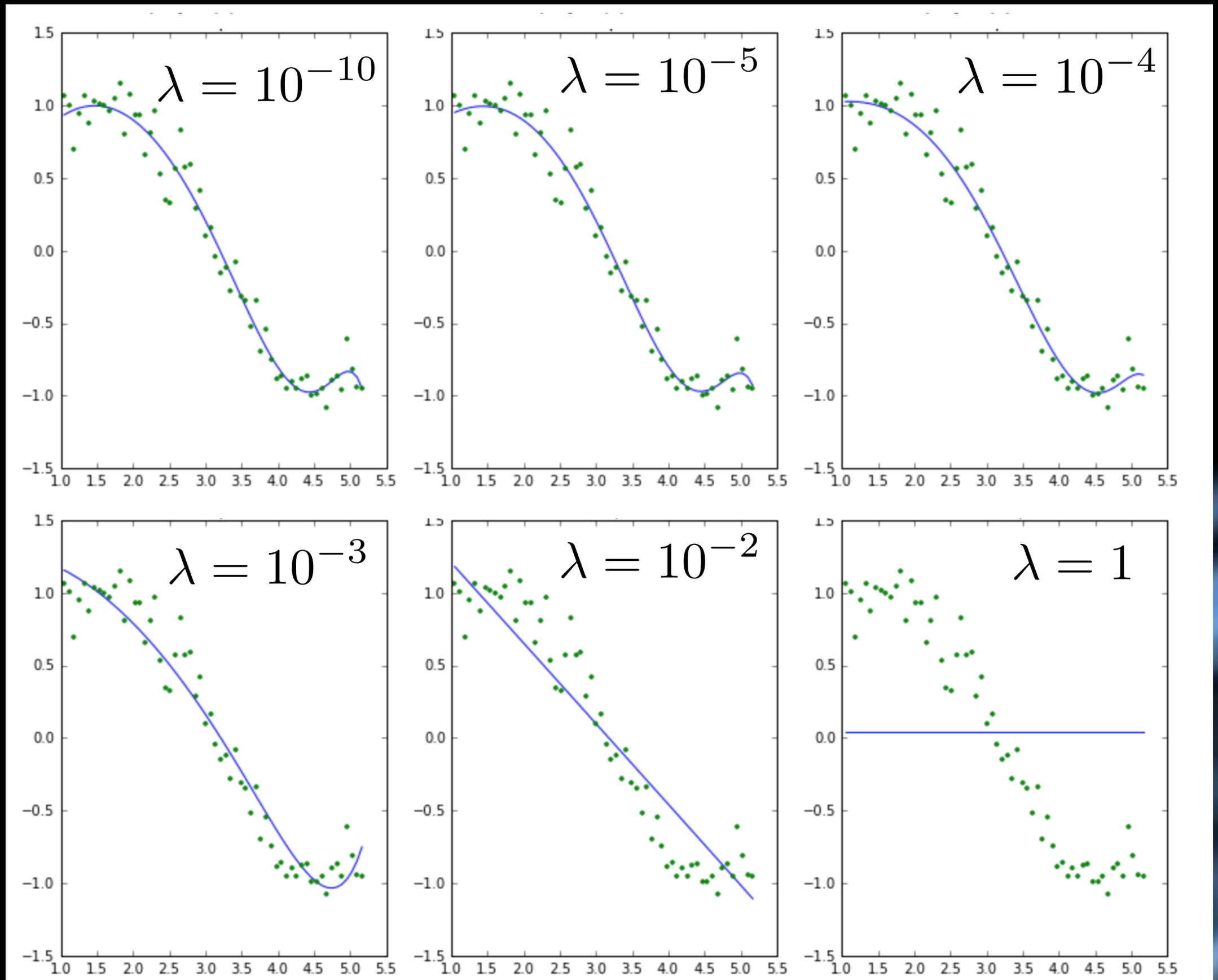
Fit coefficients & Residuals

(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

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

Markov Chain Monte Carlo (MCMC)