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

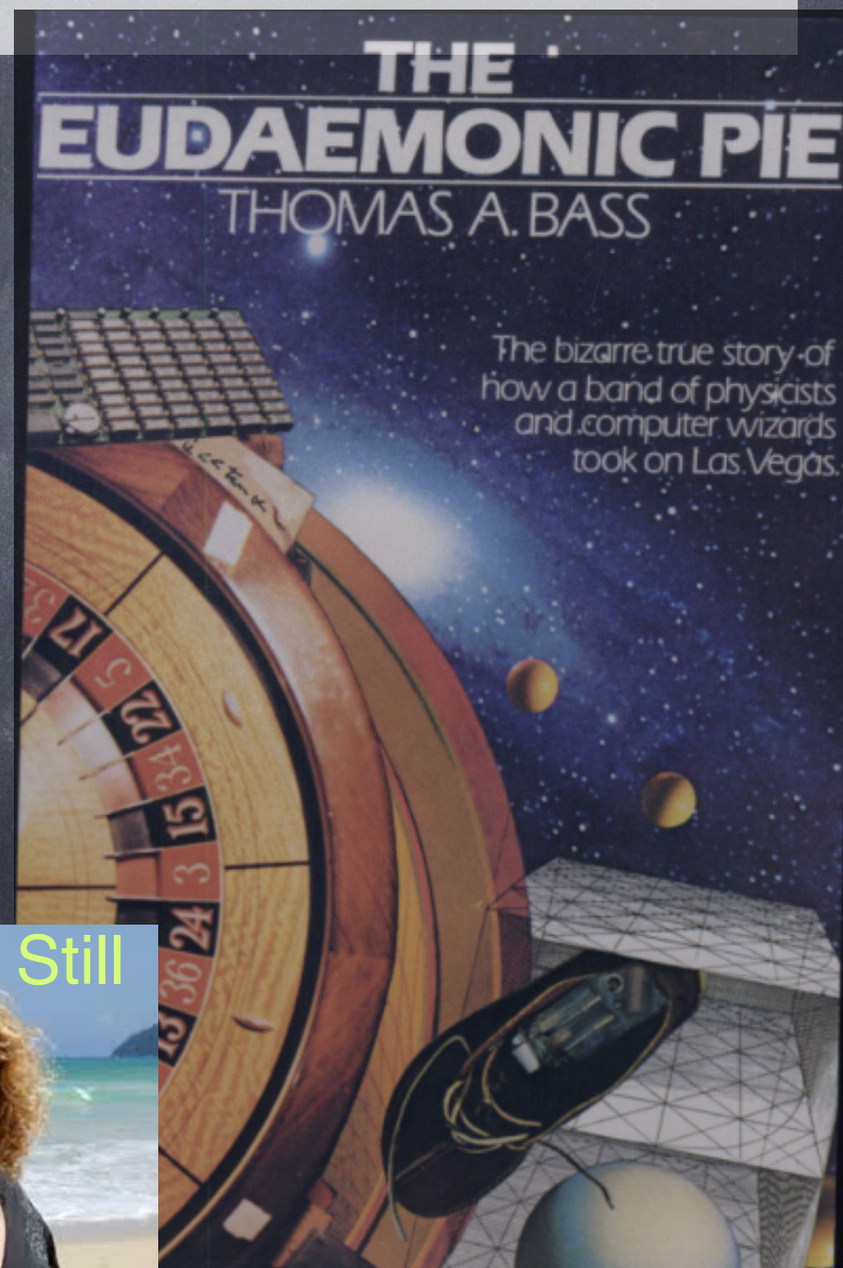
Final comment to mentioned book in week #3

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

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

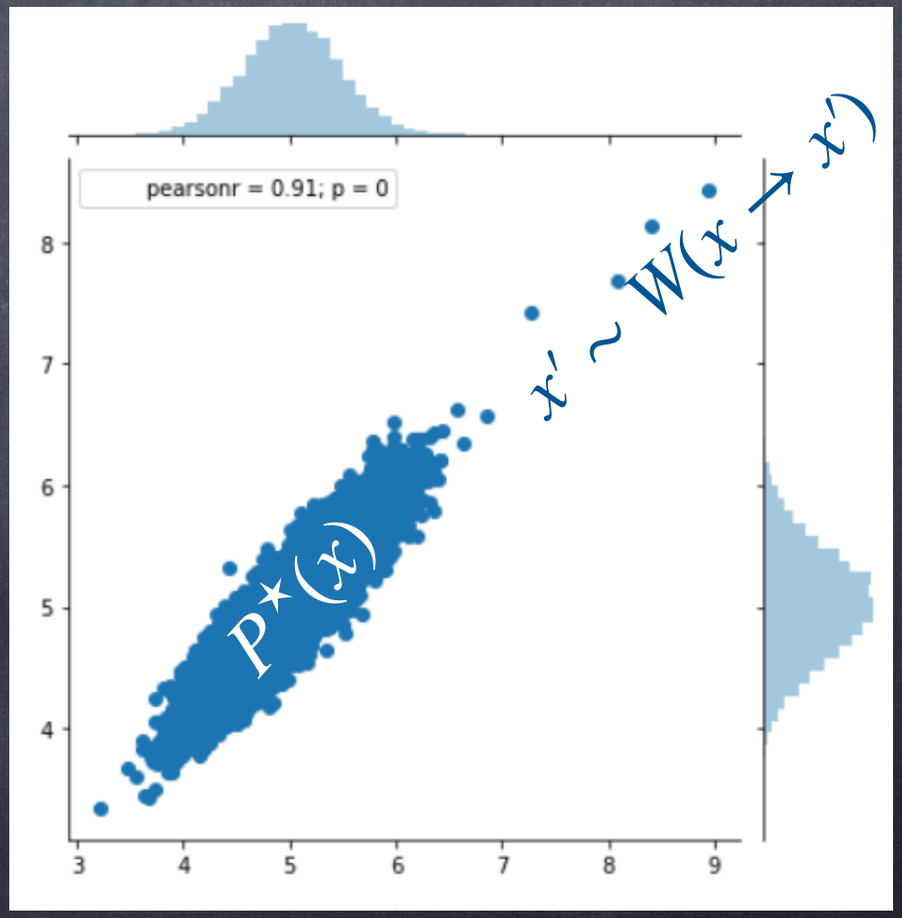
Probability (Wahrscheinlichkeit) of Markov Chain

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

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

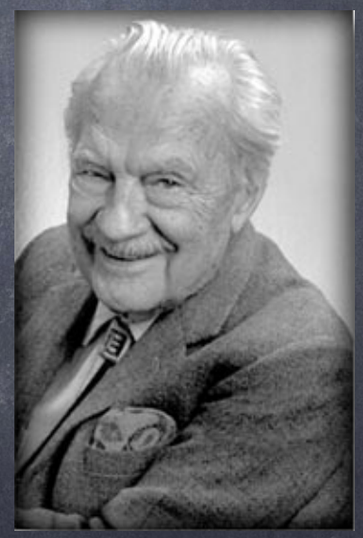
Metropolis and Metropolis-Hastings

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

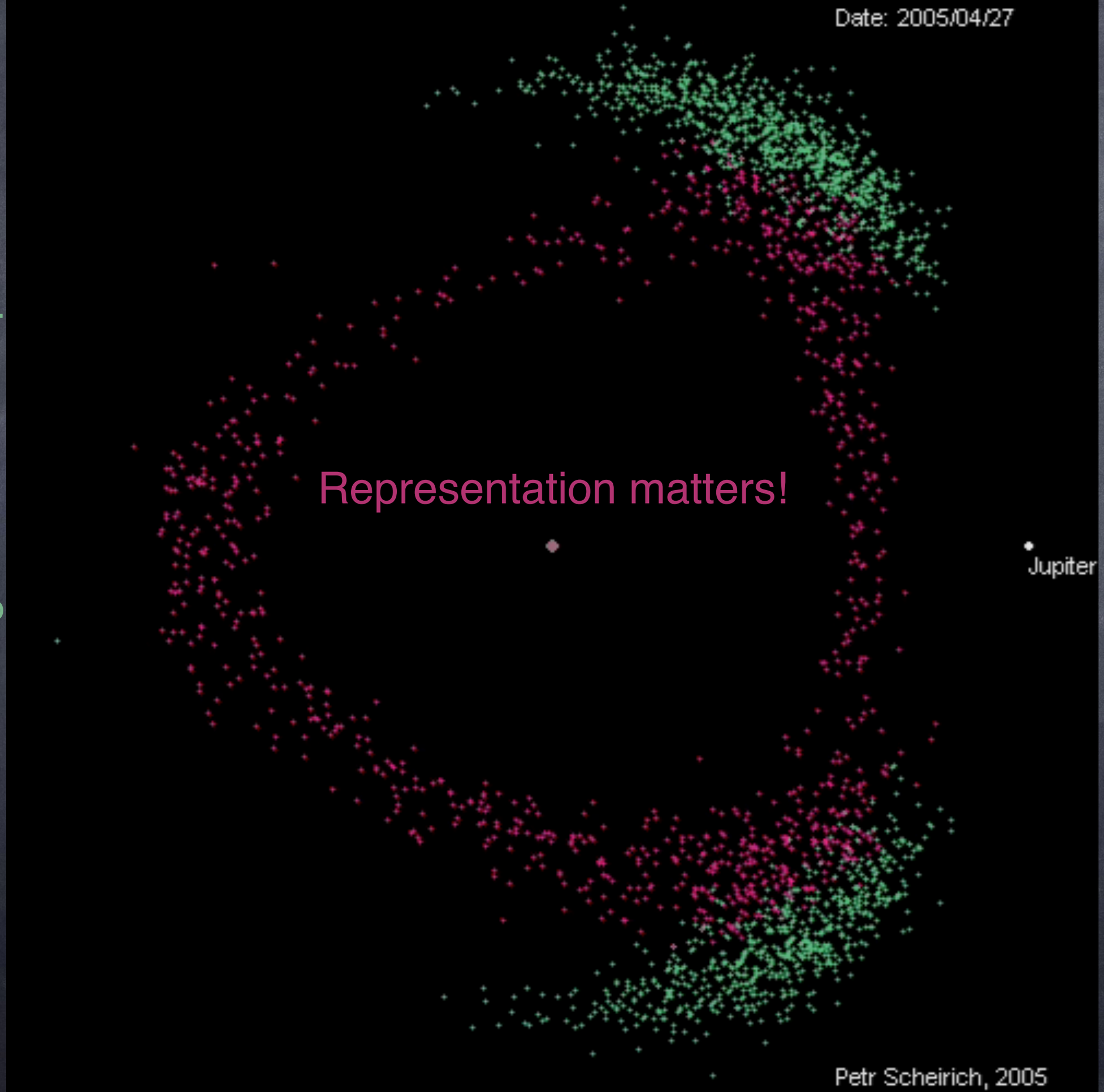
Kernel Methods

Representation matters

Heliocentrism

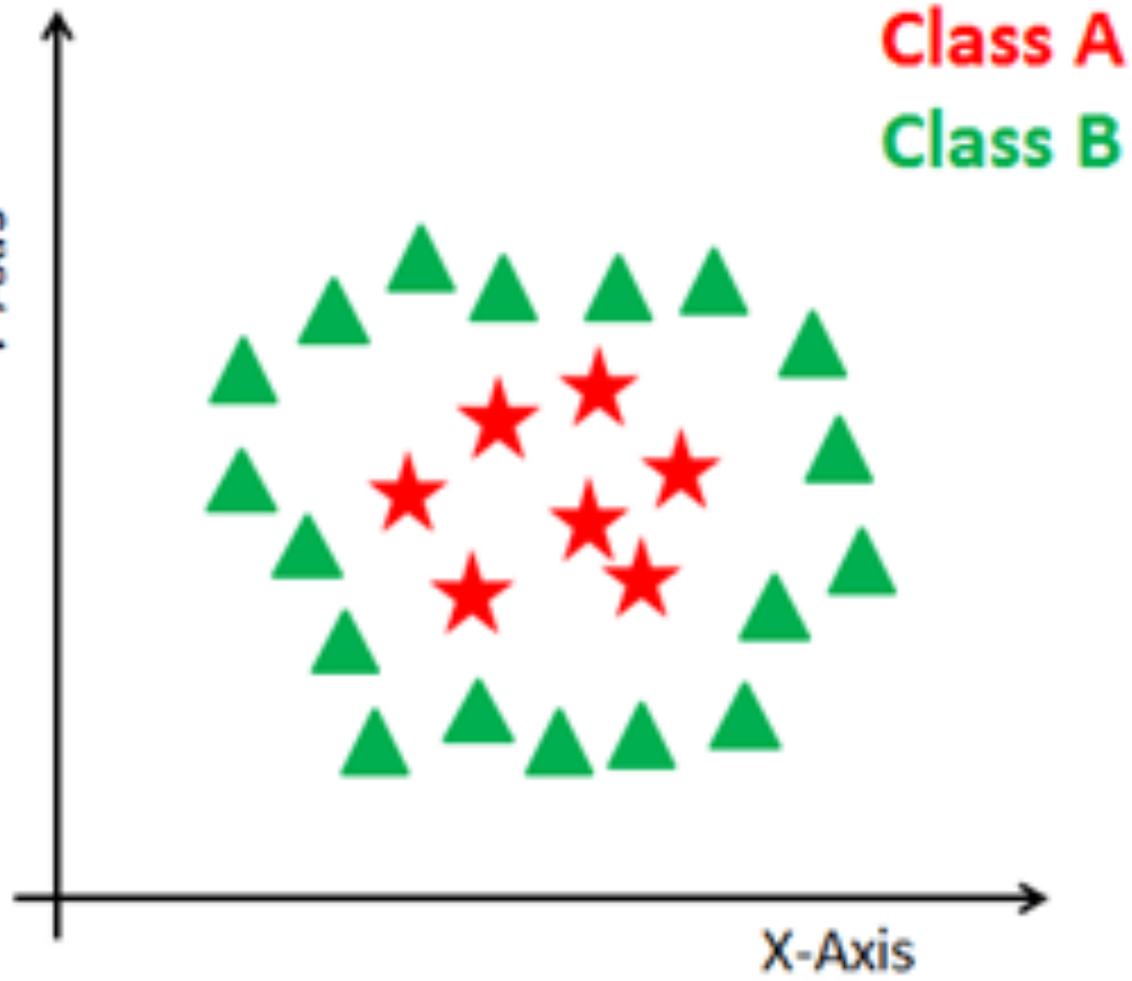
Geocentrism

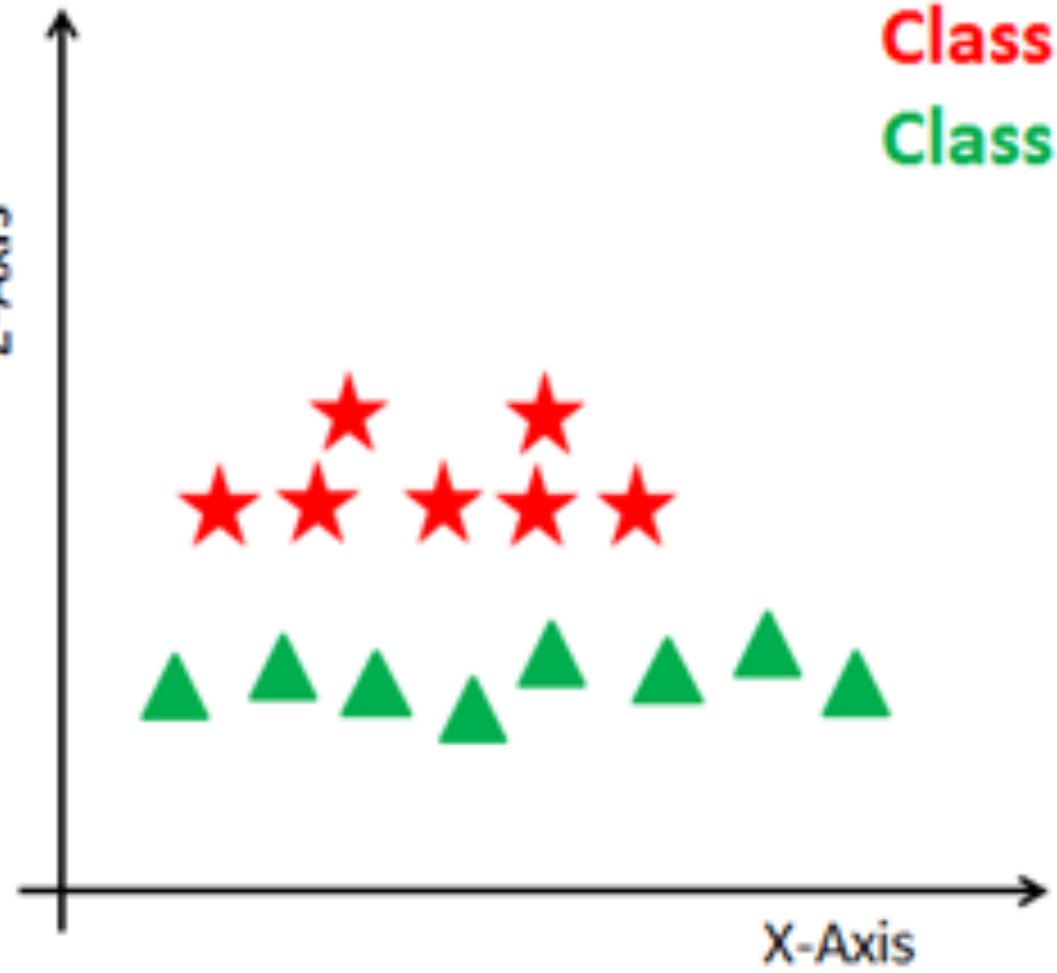
Corotating frame Sun-Jupiter



Representation matters!

Representation matters: Kernel Methods



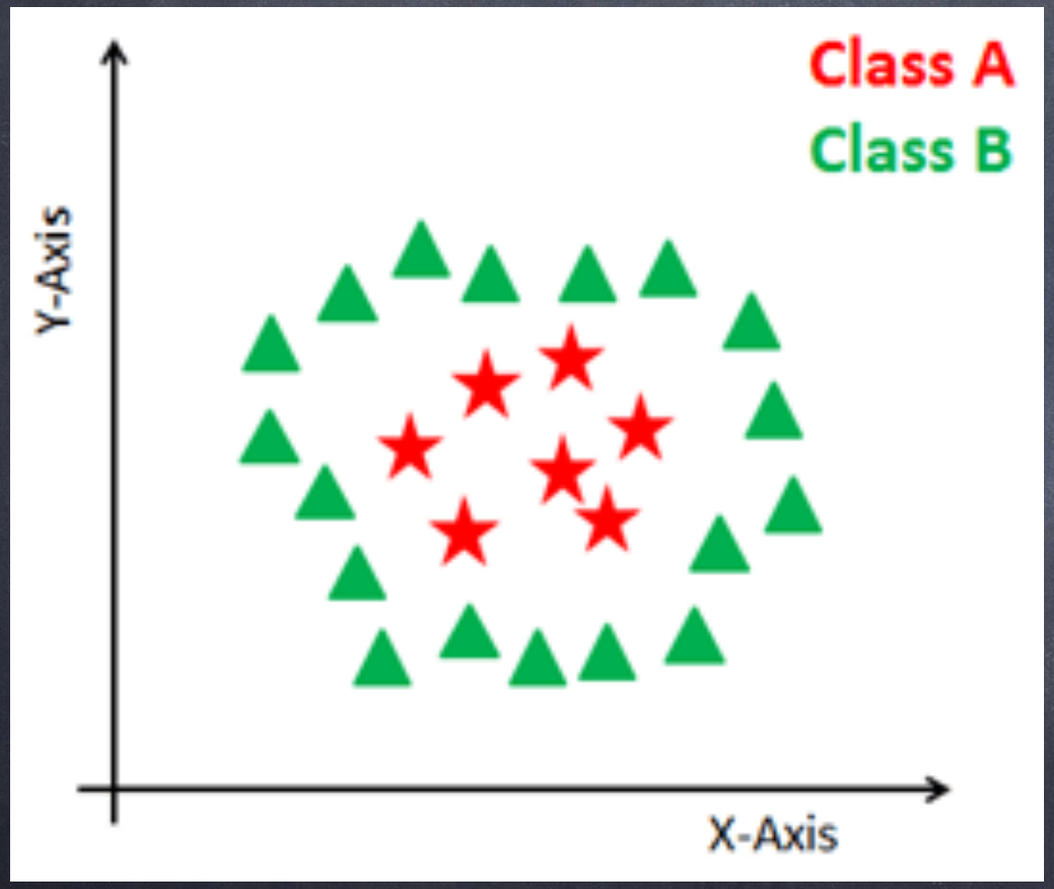


$$(x, y) \rightarrow (r, \varphi)$$

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

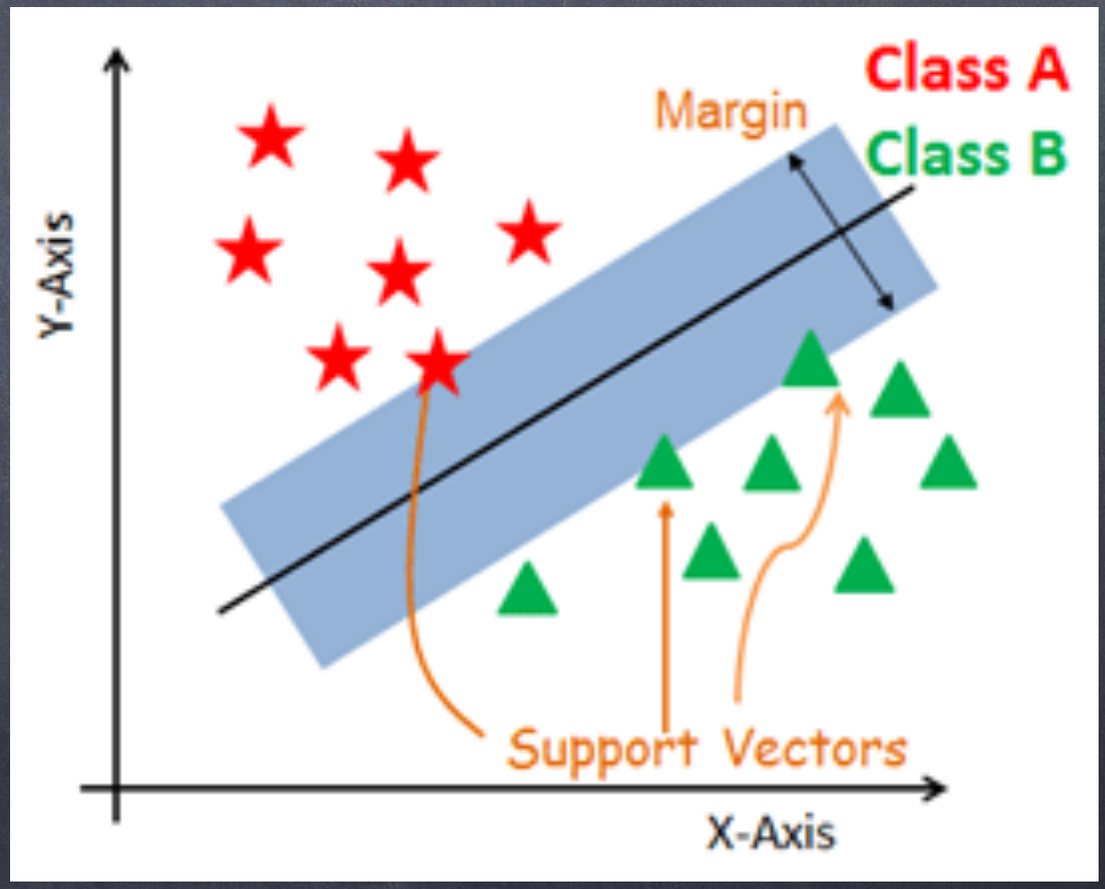
Nonlinear kernel Support Vector Machines

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



Linearly inseparable

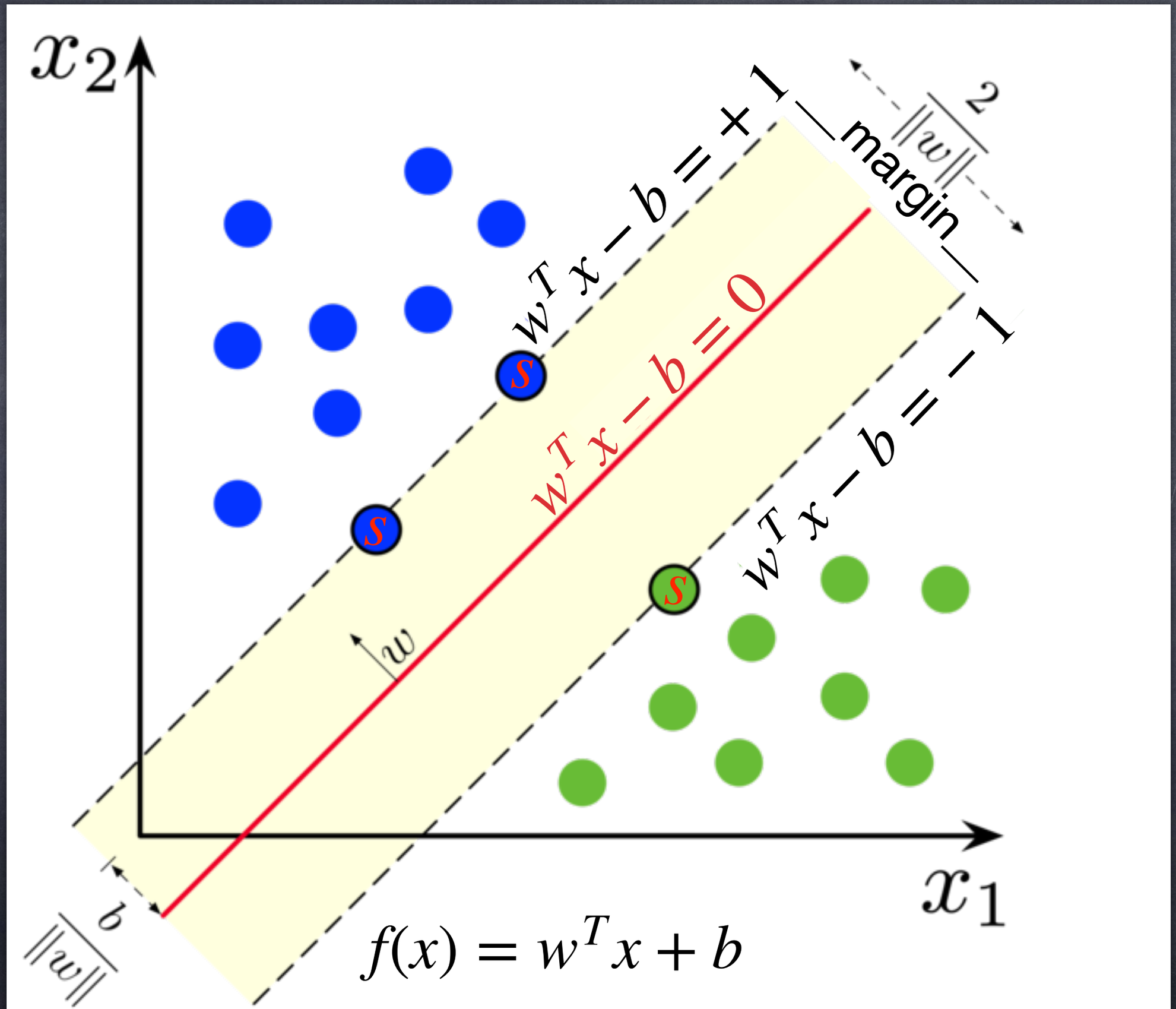
Map



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

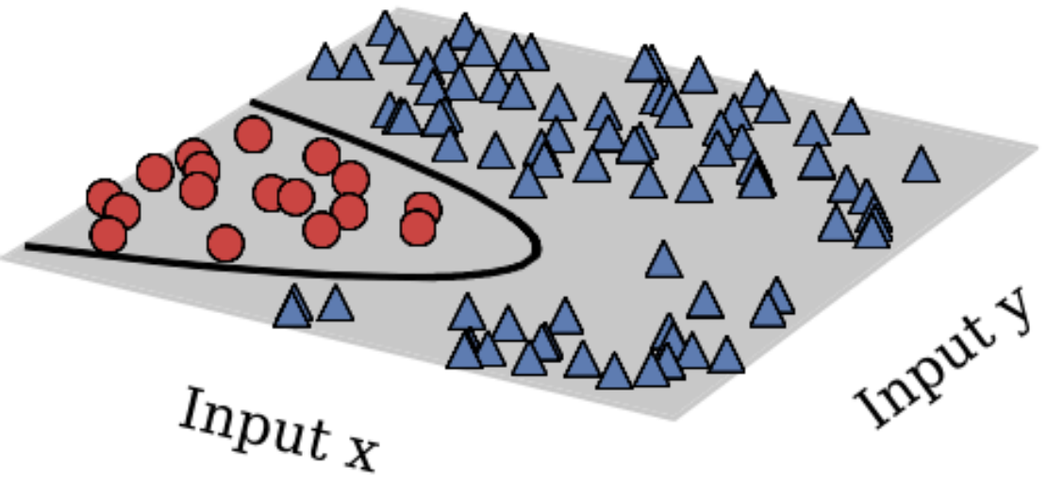
w, b : from support vectors

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

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

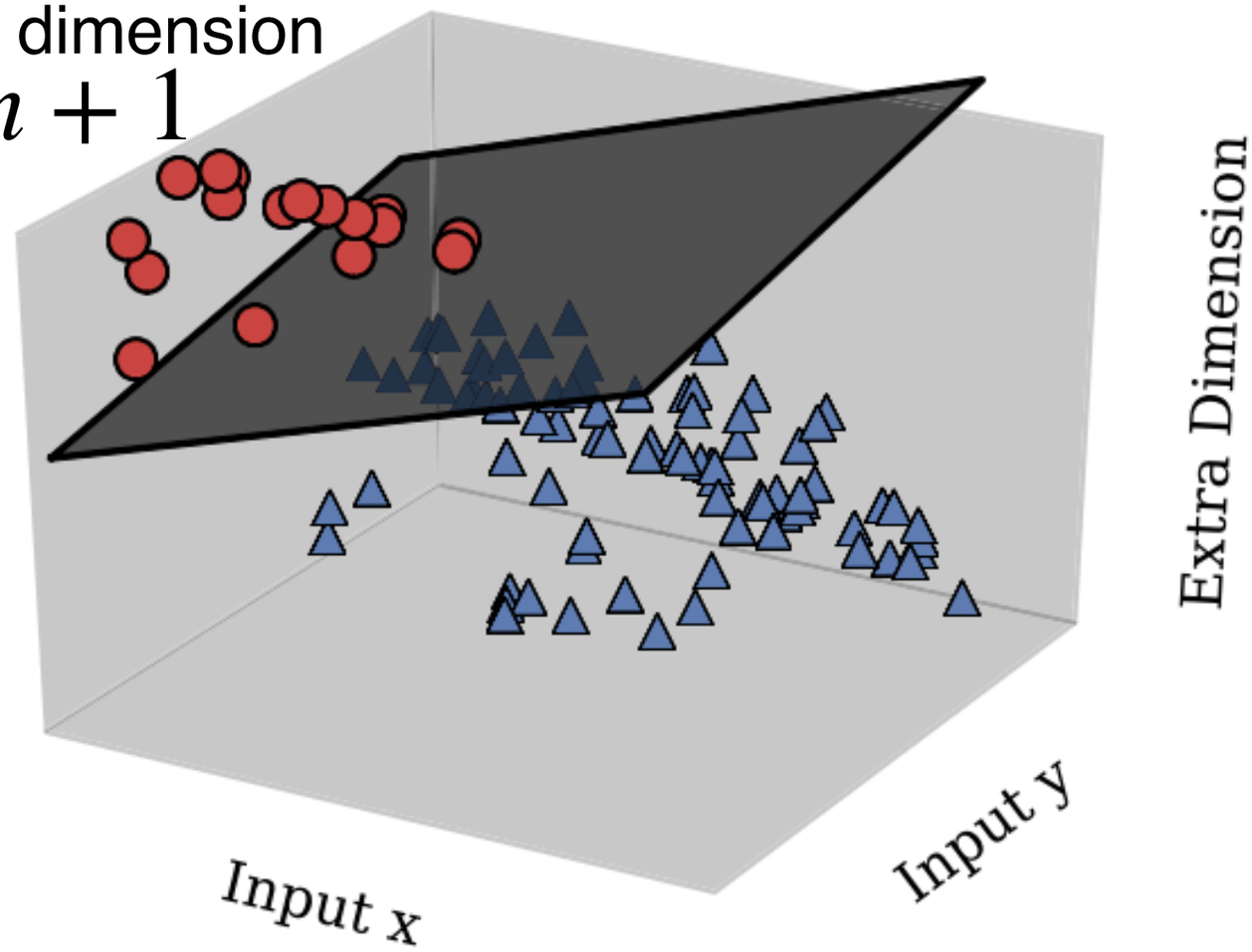
No extra dimension
but nonlinear feature map

$m = n$



With extra dimension

$m = n + 1$



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

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

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

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

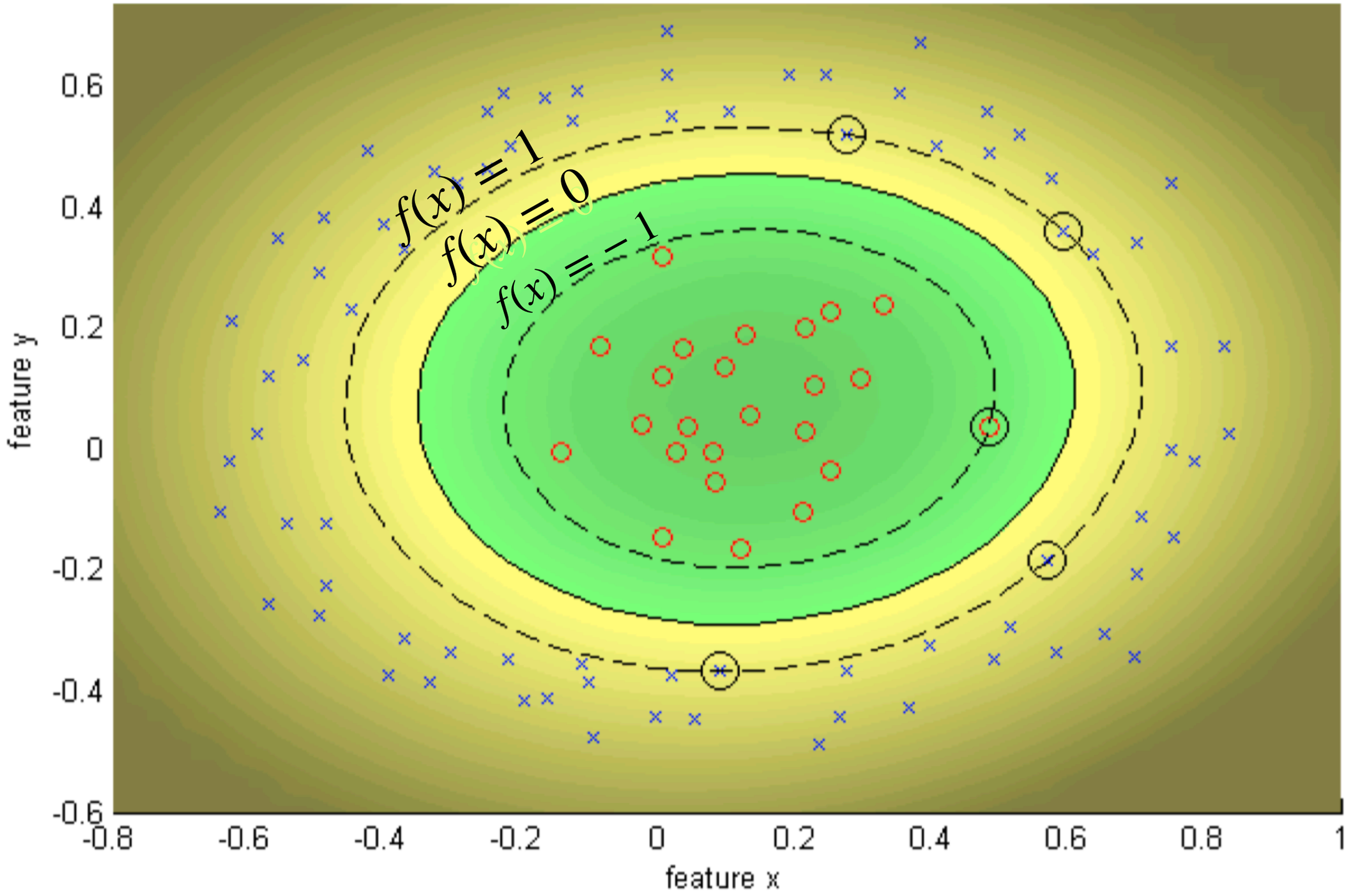
$$\sum_{i=1}^{n} \sum_{j=1}^{n} K(x_i, x_j) c_i c_j \geq 0$$

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

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

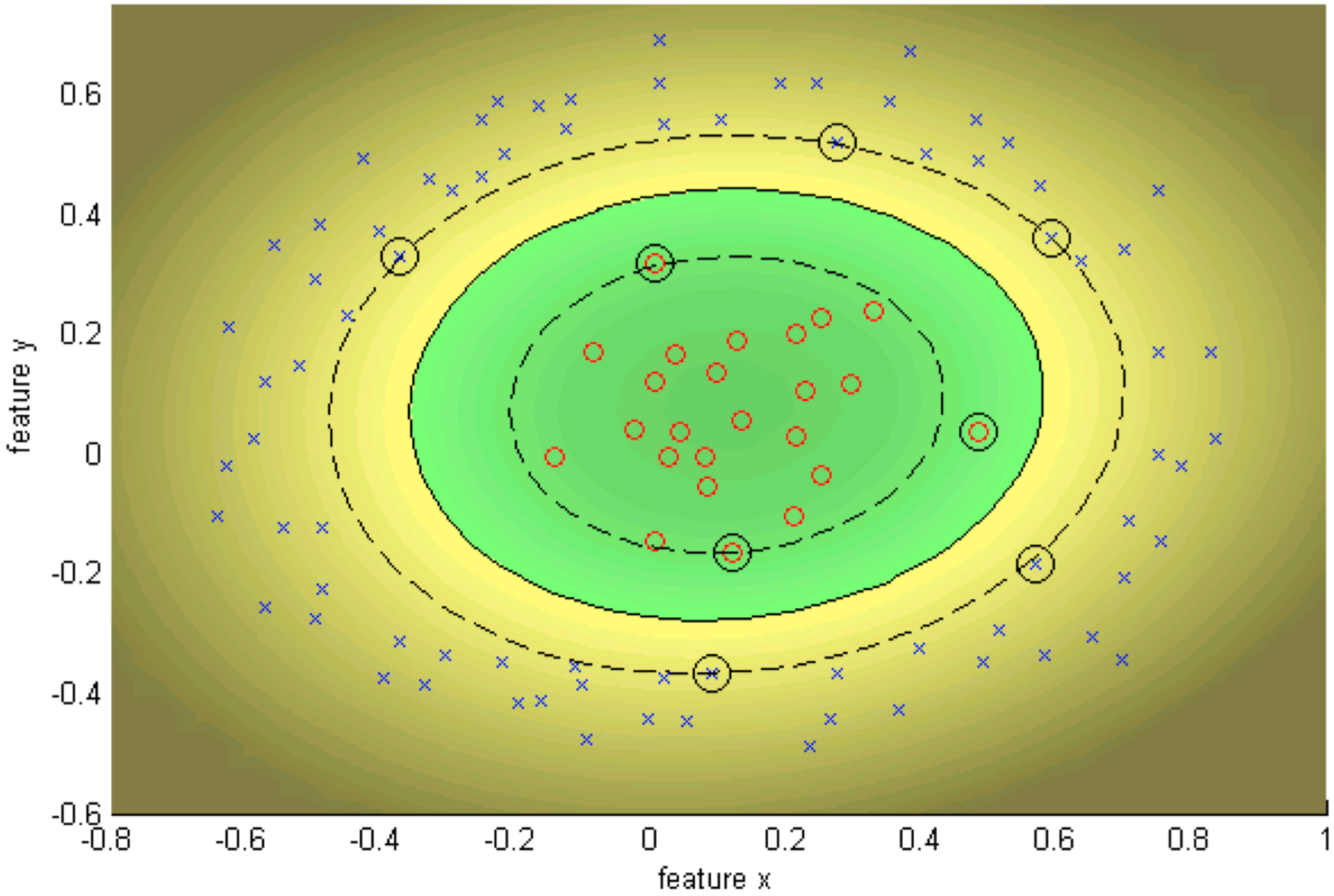
nonlinear decision boundaries



$\sigma = 1.0$ $C = 100$

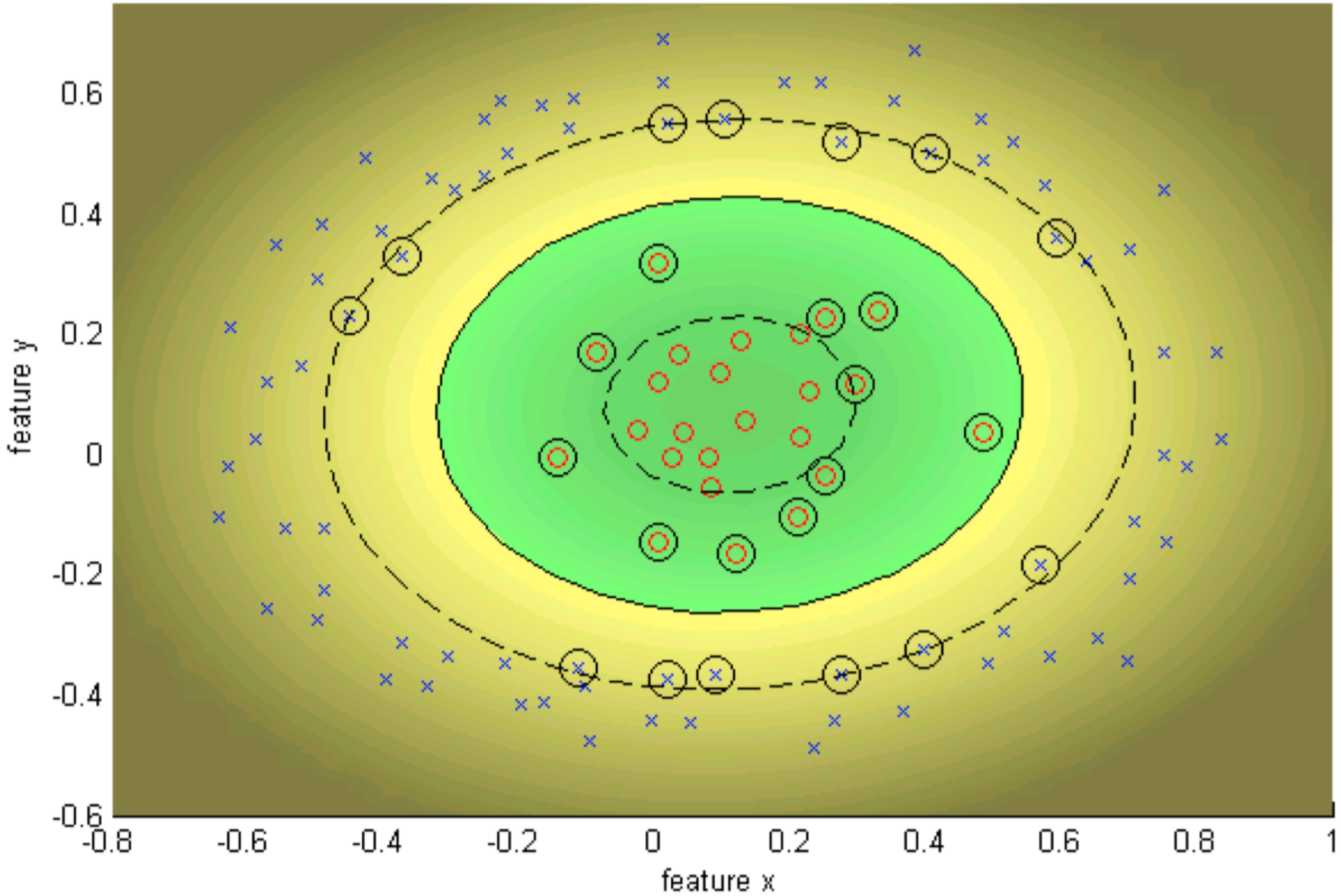
C: Soft margin parameter

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

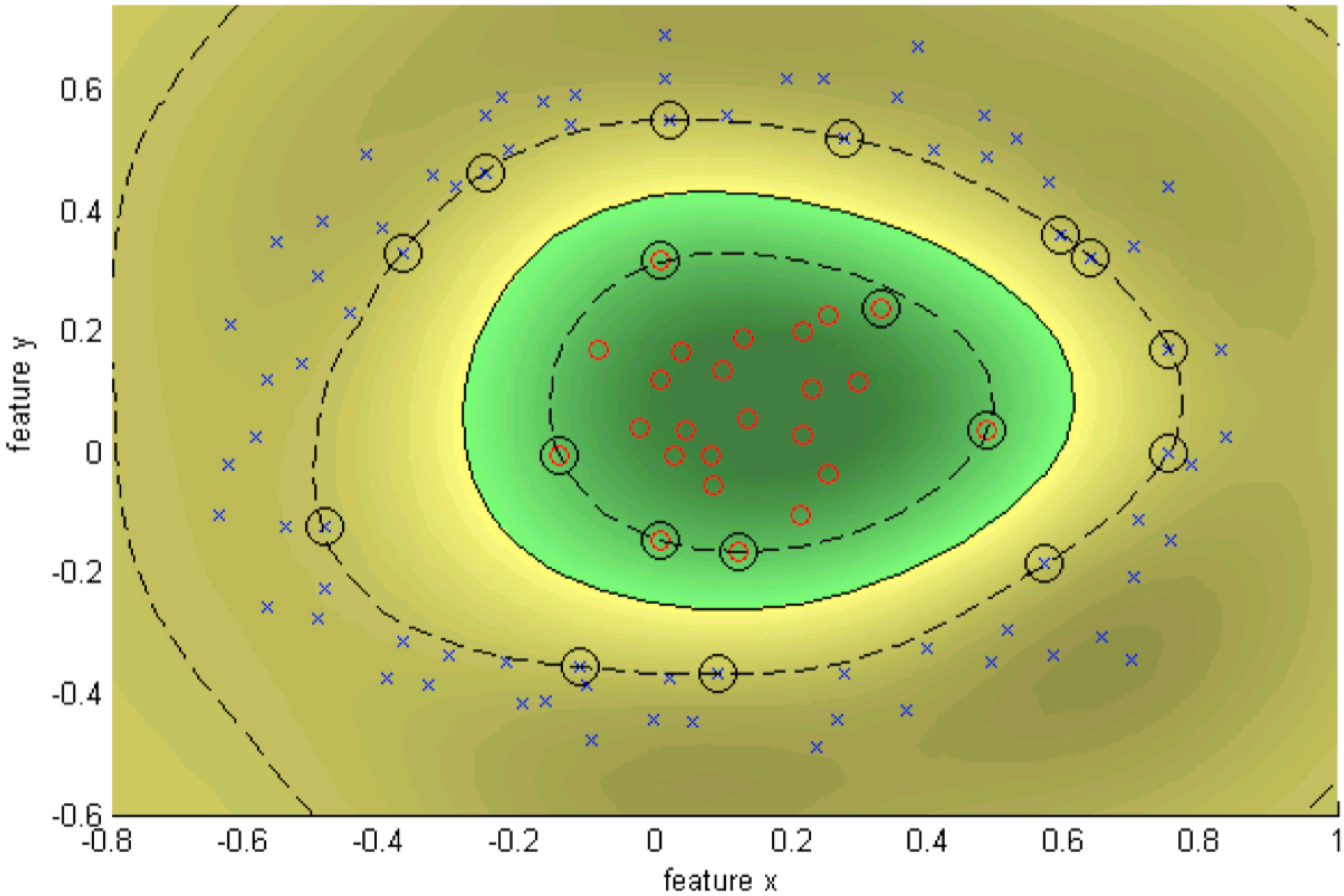
$$\gamma = 1/(2\sigma^2)$$

Omega: Vice versa

C: Soft margin parameter (as shown)

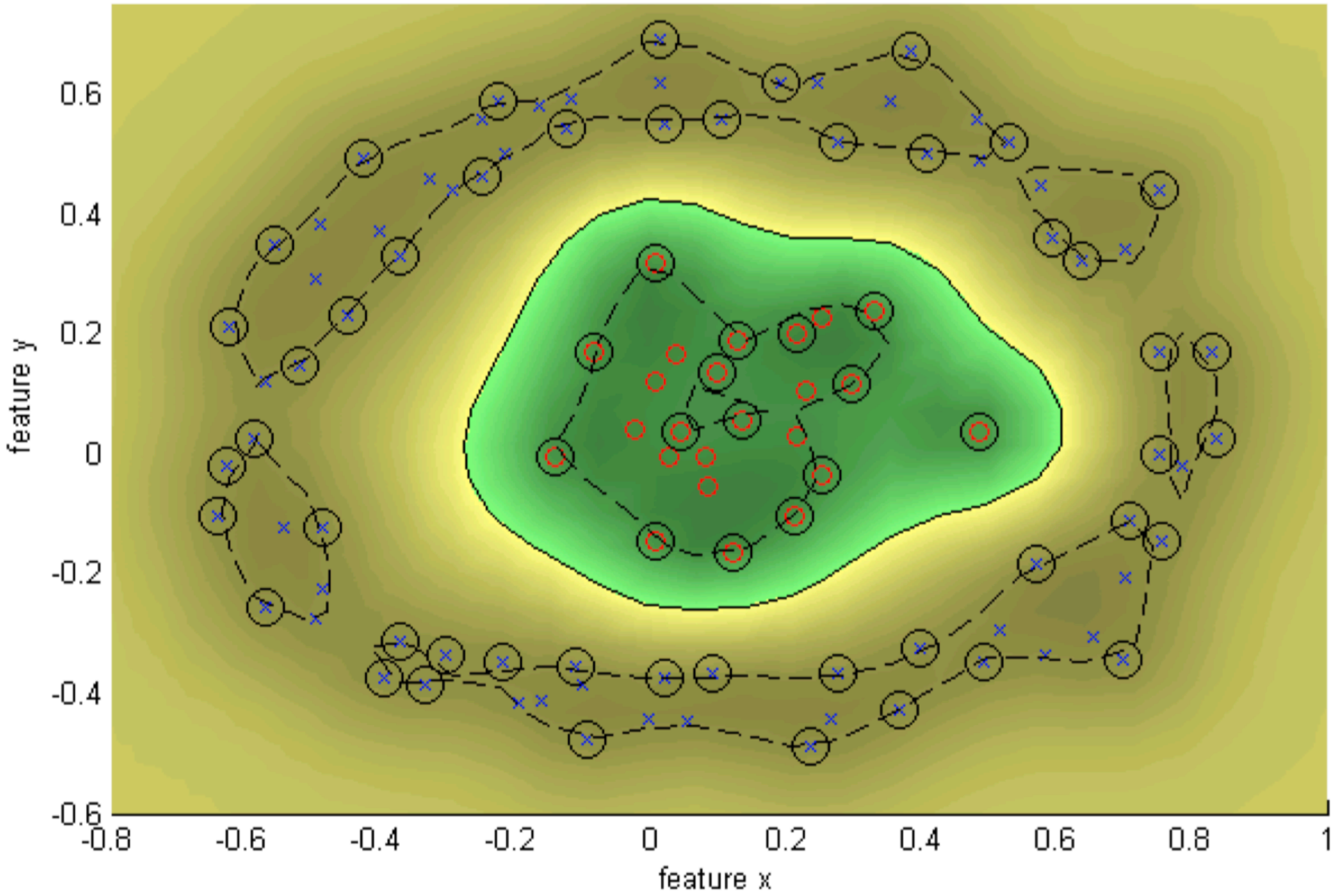
$\sigma = 0.25 \quad C = \infty$

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

Source: LyricFind
Songwriters: H. P. Baxxter / Rick J. Jordan / Jens Thele / Ferris Bueller

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`

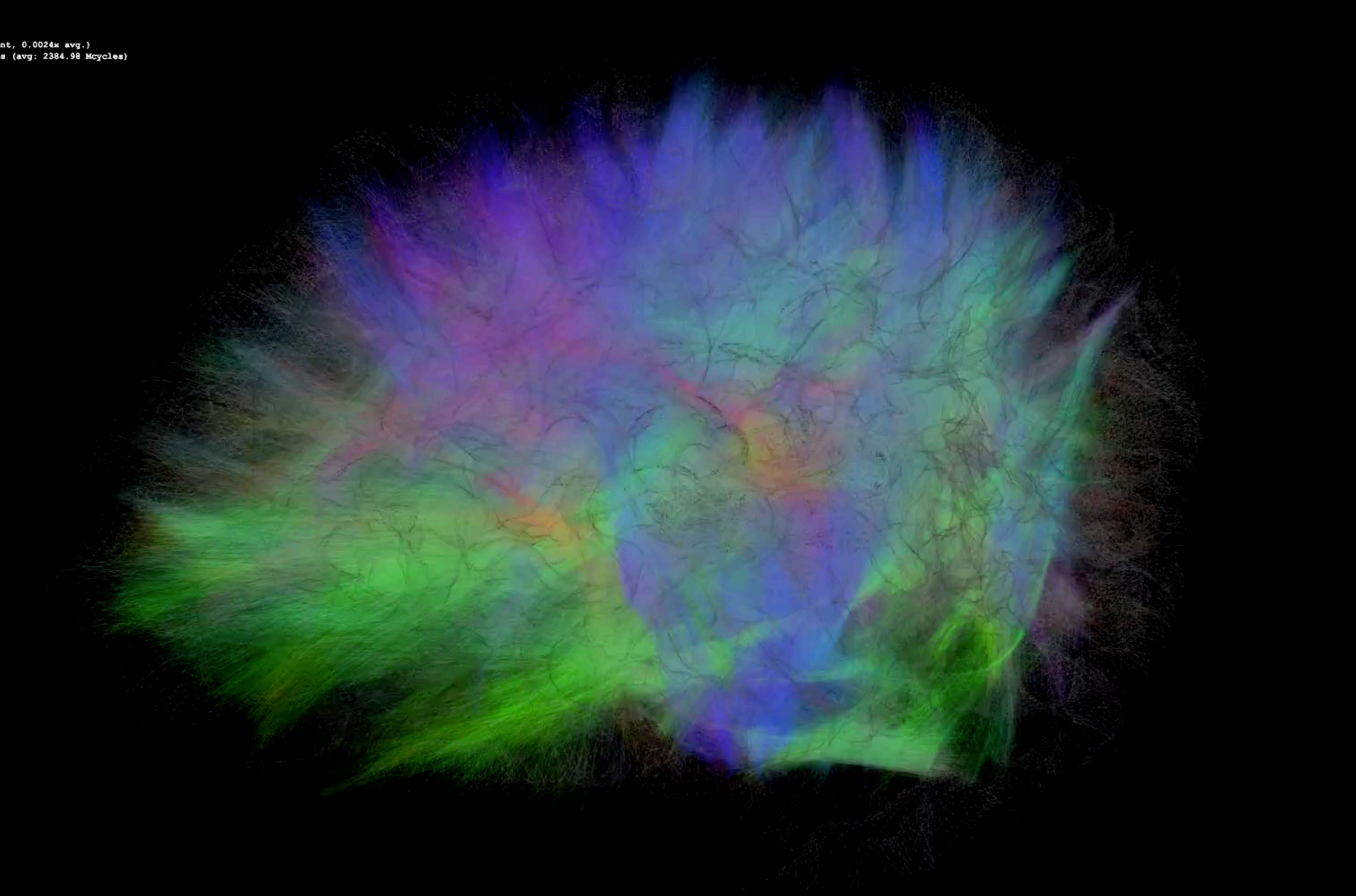
Final part:
News, Views & Threats in Machine Learning

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)

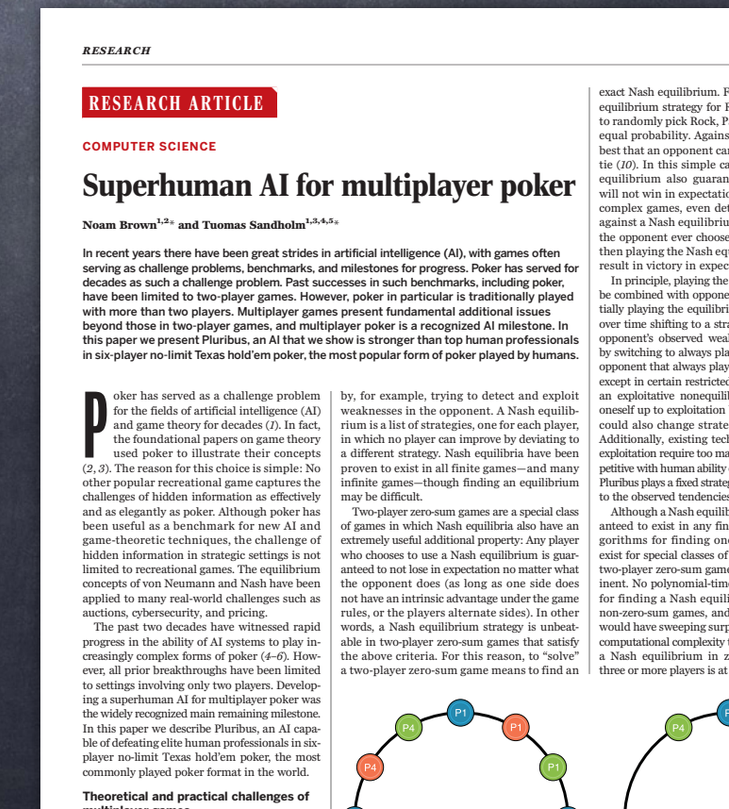


Simulated Thalamocortical Brain Network, 3 Million Neurons, 476M Synapses, Ivan Dimkovic

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 R
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 det
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 stra
opponent’s observed weal
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

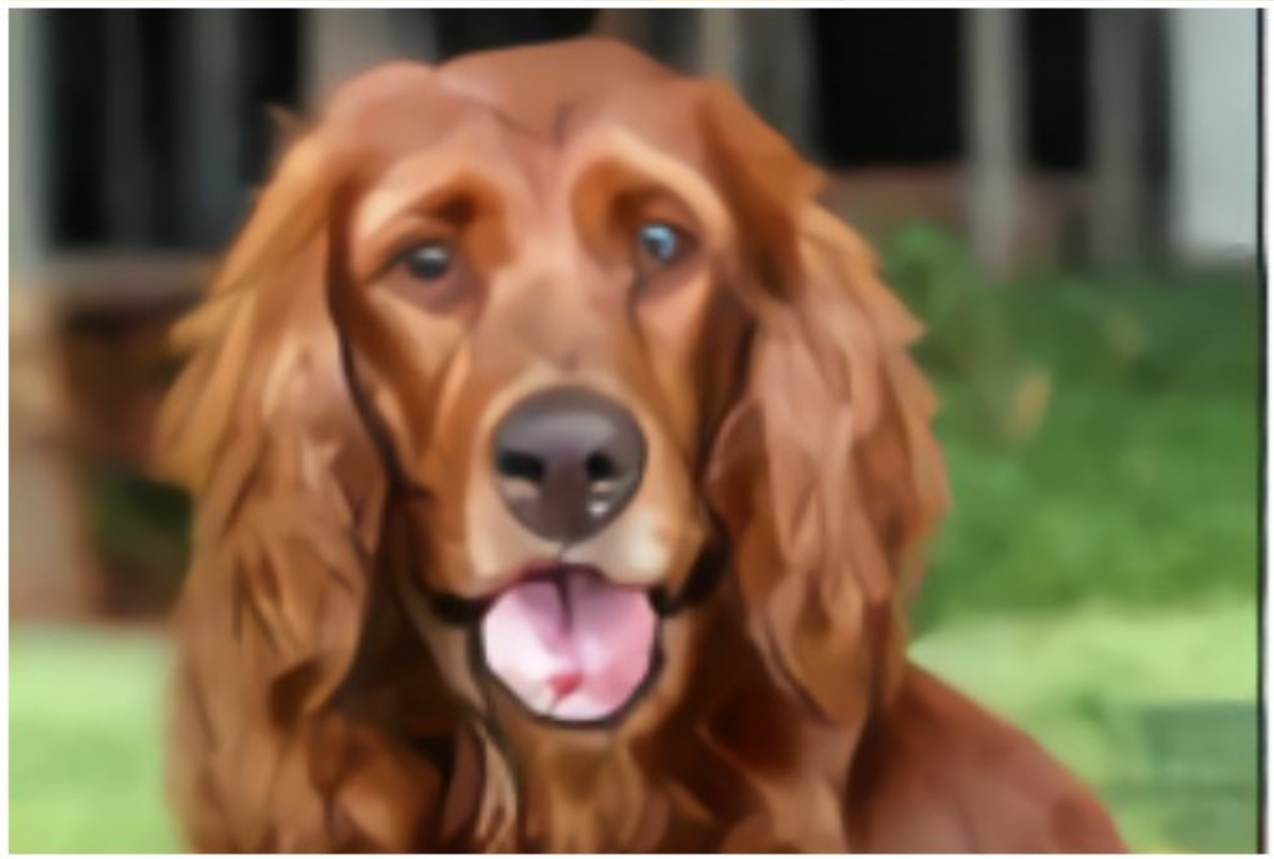
Introducing CoastRunner…

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

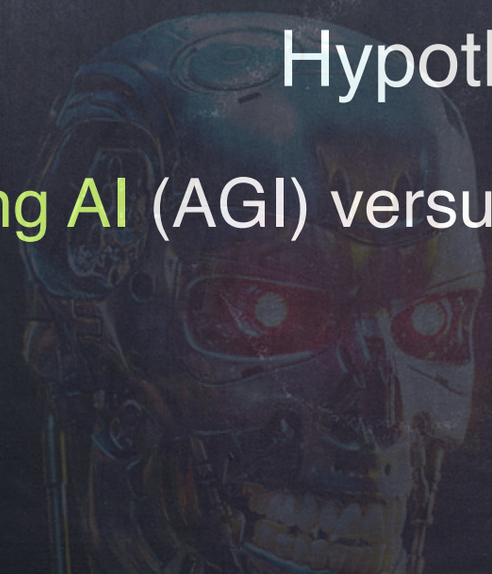
Drones (with and w/o face recognition)

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

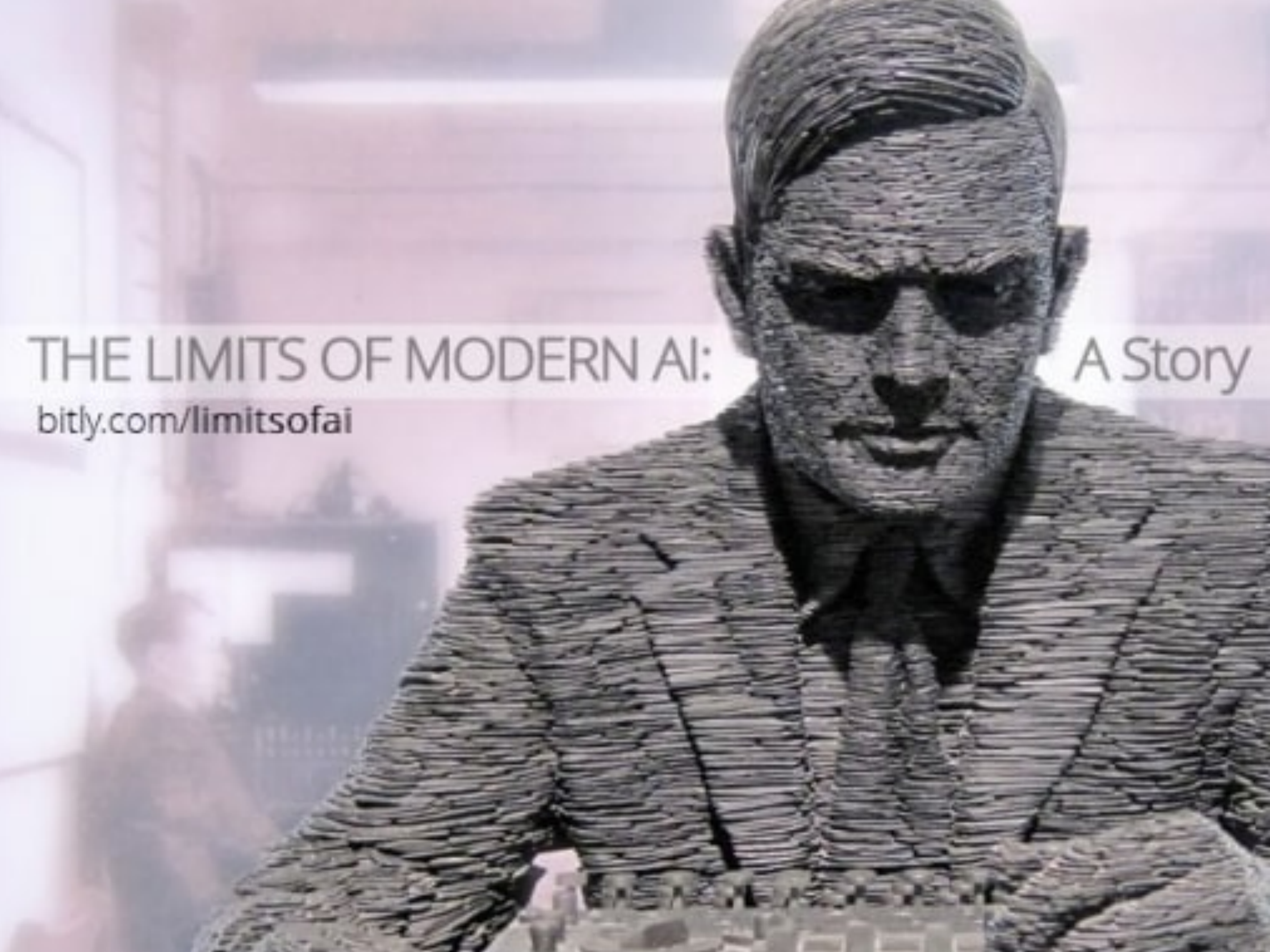
By David Hambling

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

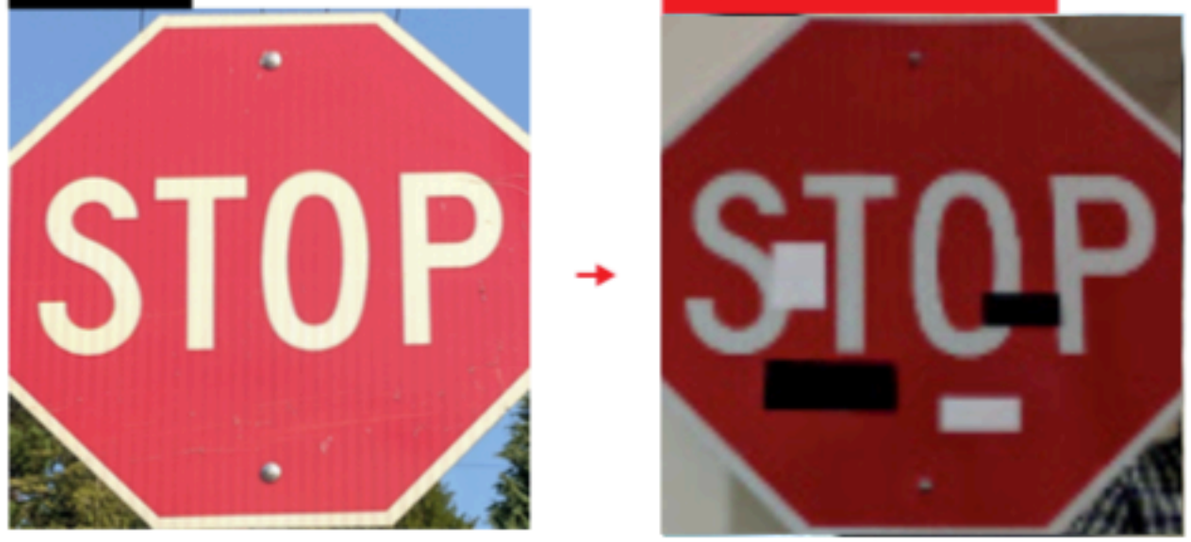
The current interest in big data, machine learning and data analytics has generated the widespread impression that such methods are capable of solving most problems without the need for conventional scientific methods of inquiry. Interest in these methods is intensifying, accelerated by the ease with 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

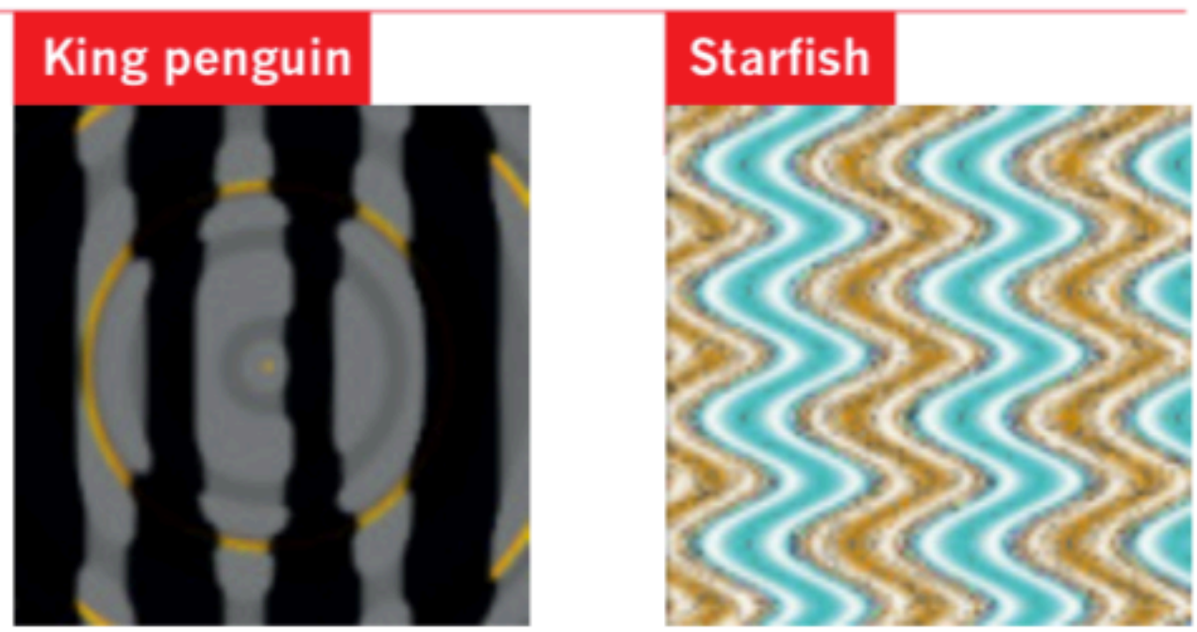
FOOLING THE AI

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

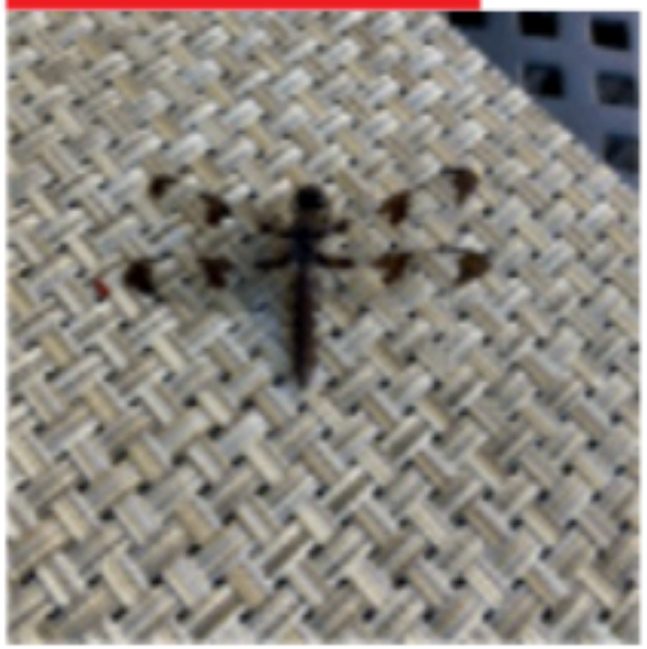
Adding carefully crafted noise to a picture can create a new image that people would see as identical, but which a DNN sees as utterly different.

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