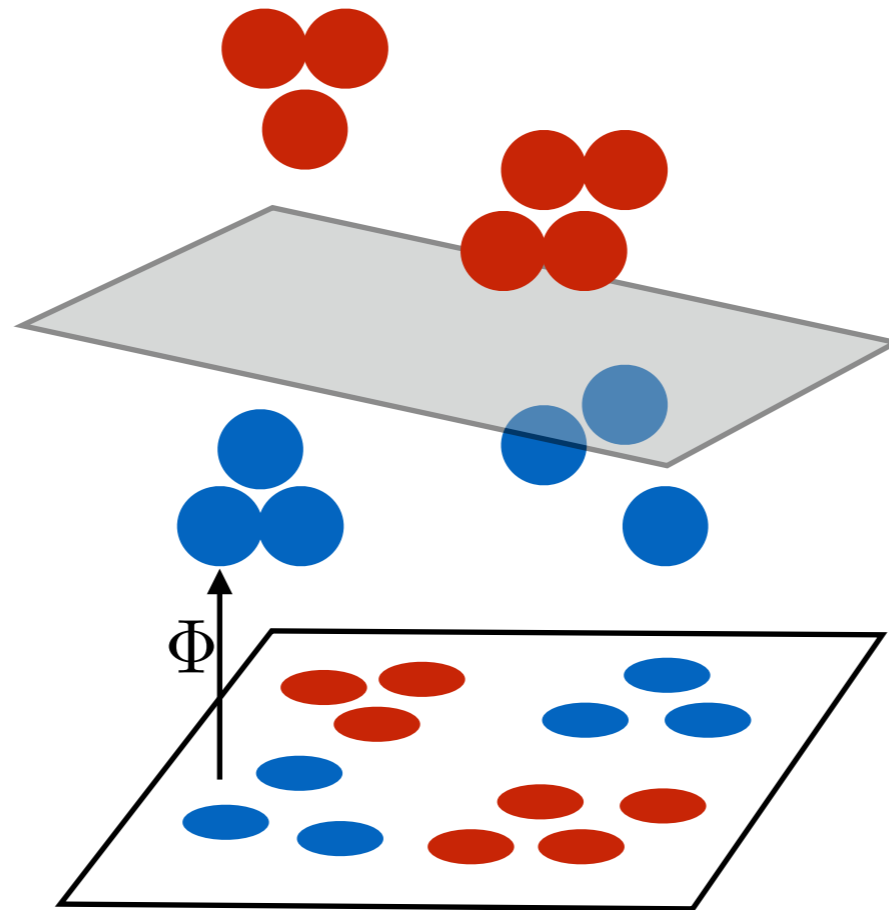


Introduction to Machine Learning



Machine learning galvanizing industry & science



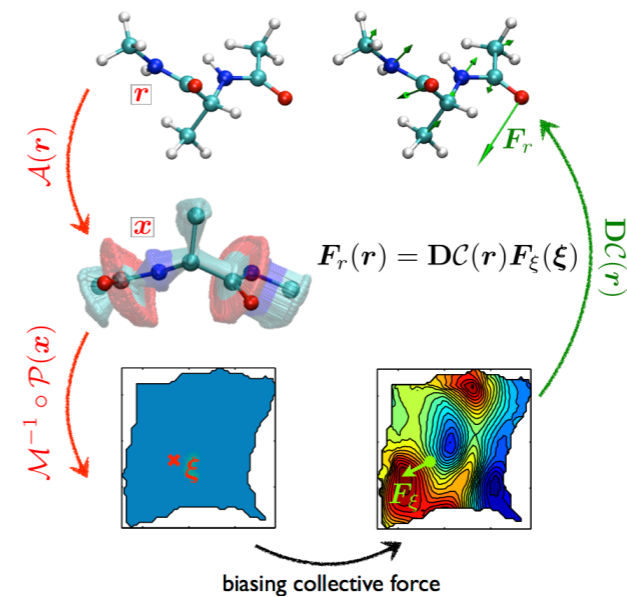
Language Processing



Self-driving cars

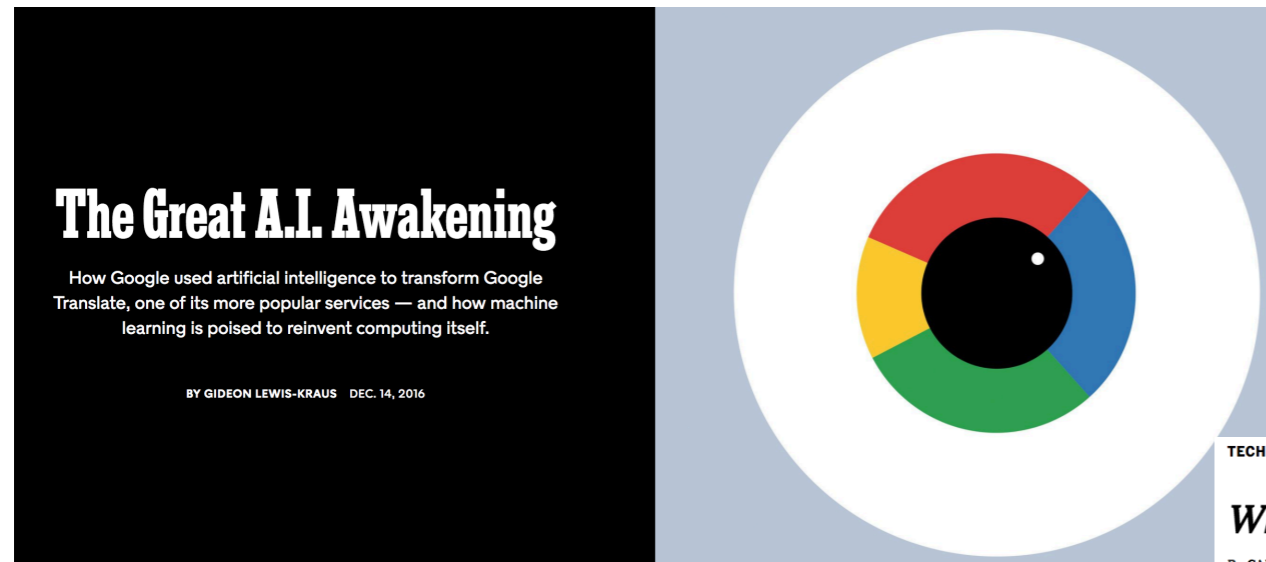


Medicine



Materials Science / Chemistry

Google rebranded a "machine learning first company"



TECHNOLOGY

Why A.I. Researchers at Google Got Desks Next to the Boss

By CADE METZ FEB. 19, 2018



Neural nets replace linguistic approach to Google Translate

arXiv.org > quant-ph > arXiv:1802.06002

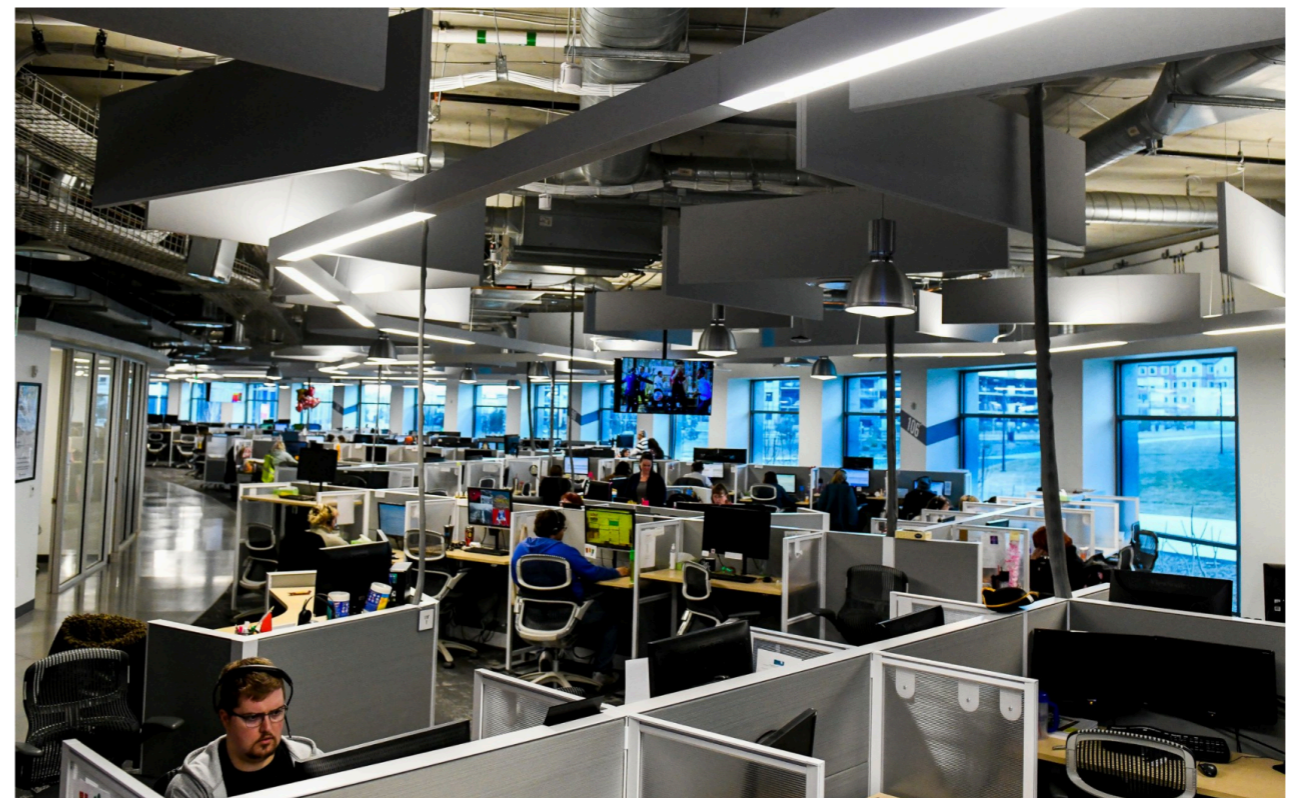
Quantum Physics

Classification with Quantum Neural Networks on Near Term Processors

Edward Farhi, Hartmut Neven

(Submitted on 16 Feb 2018)

Quantum machine learning



Examples of Machine Learning

Image recognition

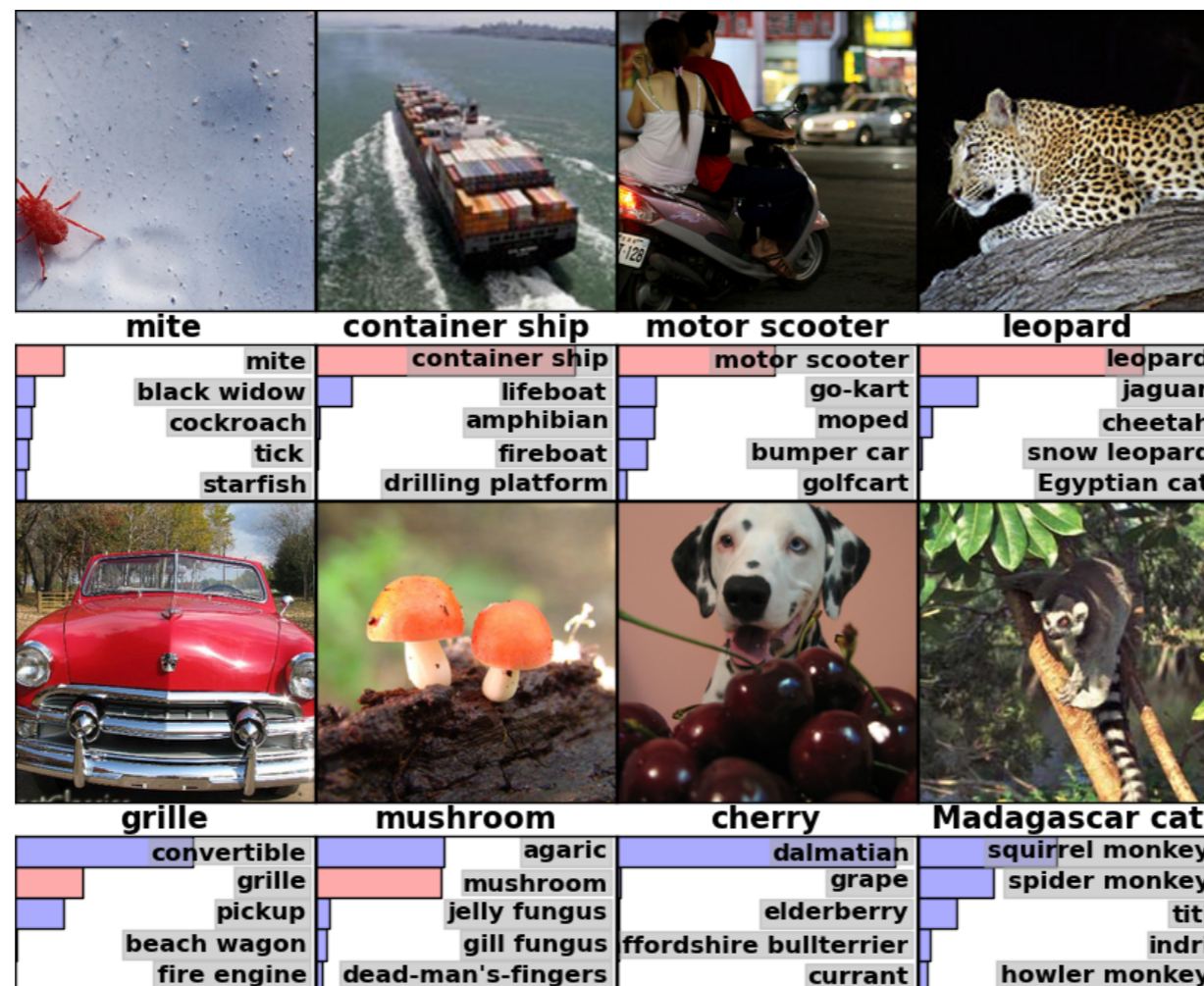
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

2012 paper that launched recent deep learning craze (20k citations)



ImageNet:

- 1.2 million training images (150k test)
- 1000 categories
- 15% neural net error
- 26% next best error

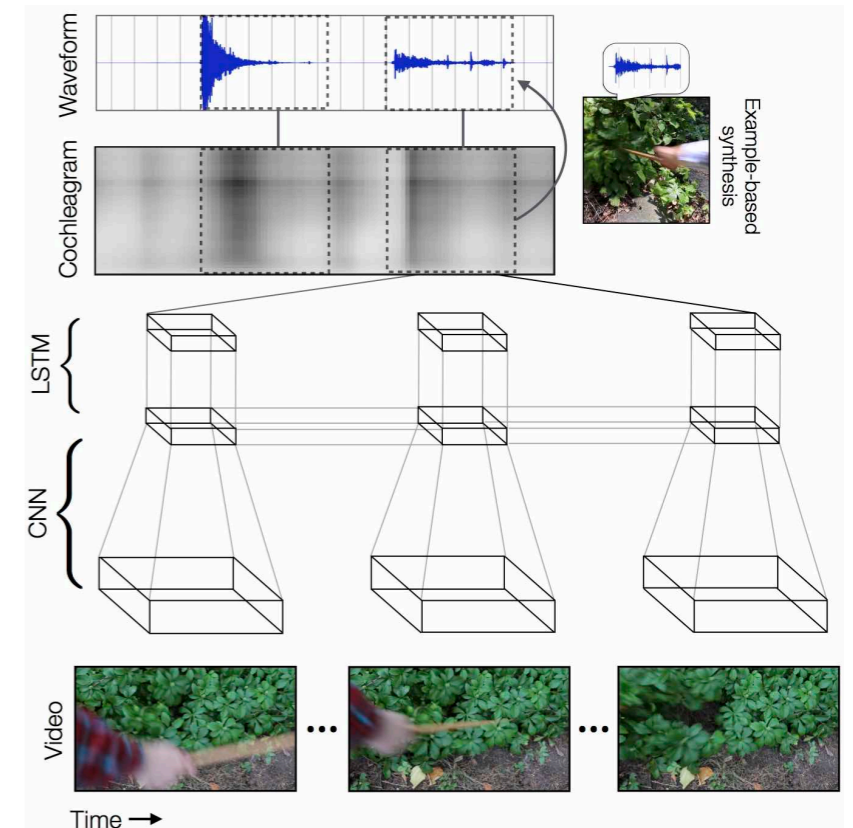
Sound prediction

Visually Indicated Sounds

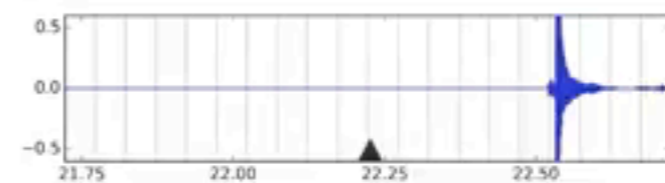
Andrew Owens¹
Antonio Torralba¹
¹MIT

Phillip Isola^{2,1}
Edward H. Adelson¹
²U.C. Berkeley

Josh McDermott¹
William T. Freeman^{1,3}
³Google Research

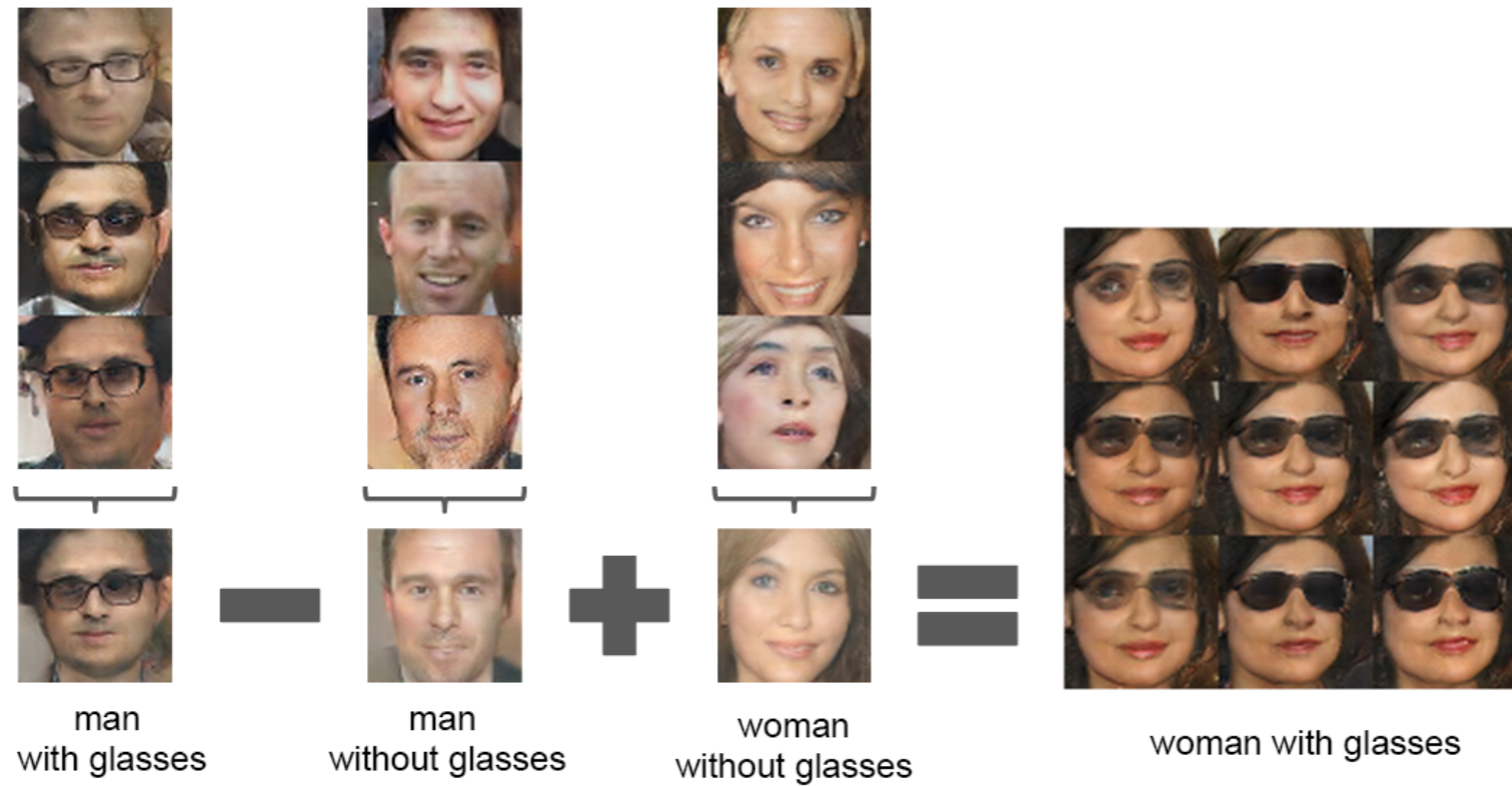


Silent video



Predicted soundtrack

Image Generation

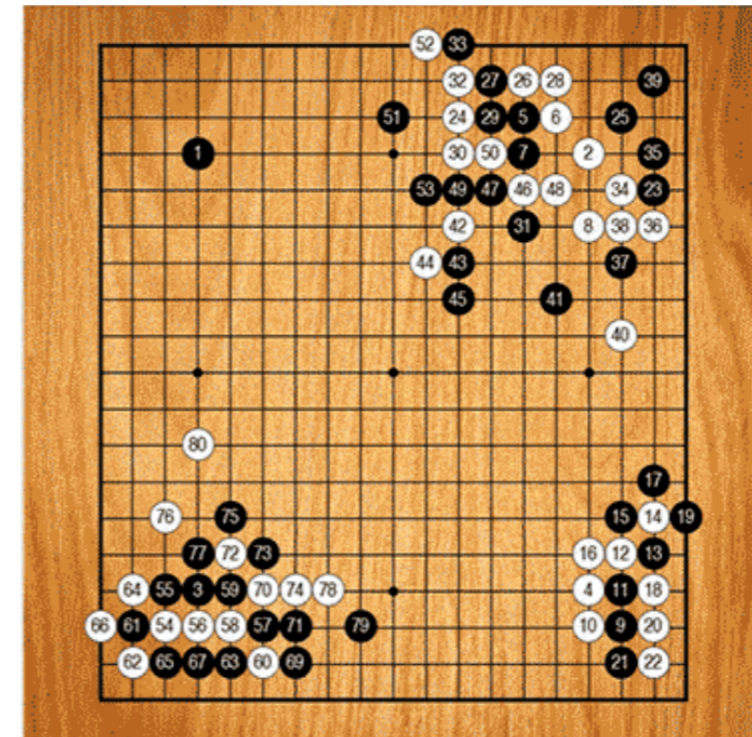
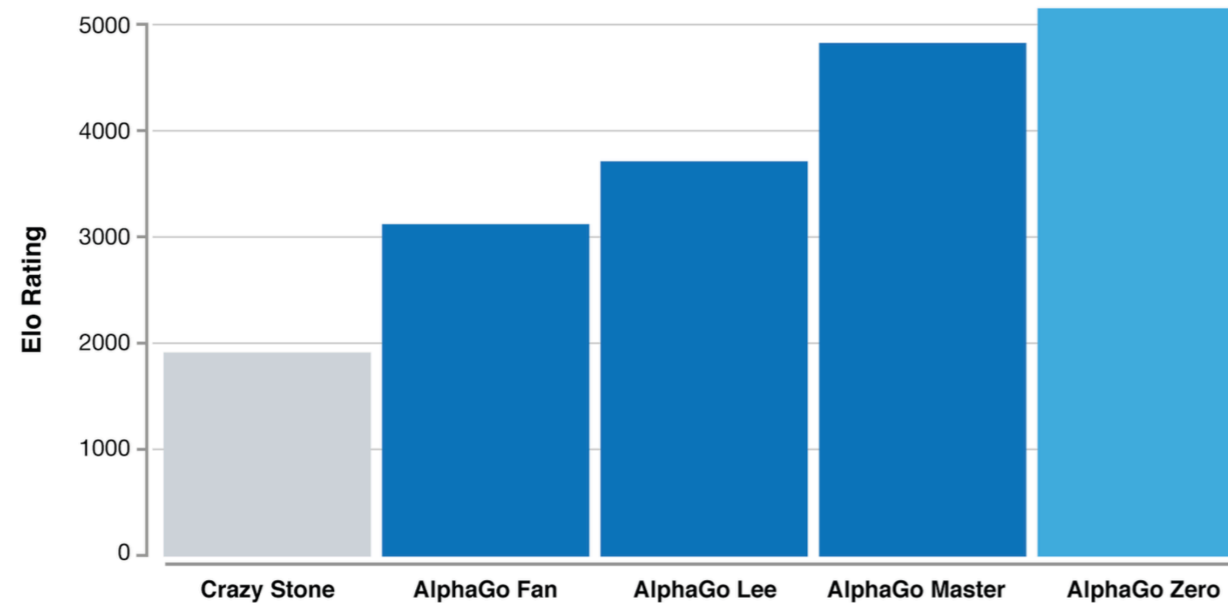


UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

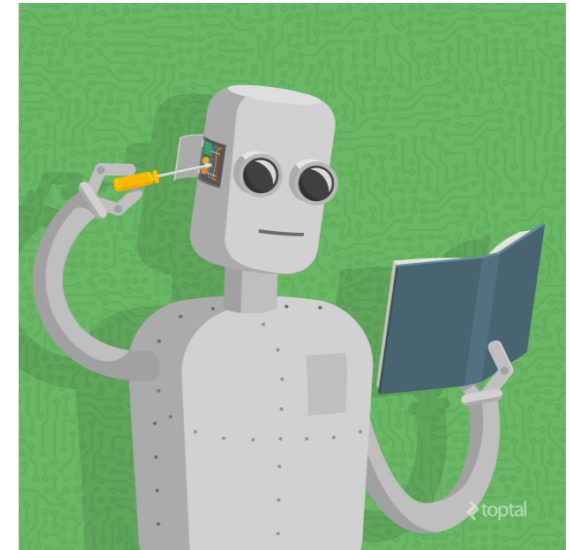
Alec Radford & Luke Metz
indico Research
Boston, MA
{alec, luke}@indico.io

Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

Success at tasks previously thought impossible



What is machine learning?



Data driven problem solving

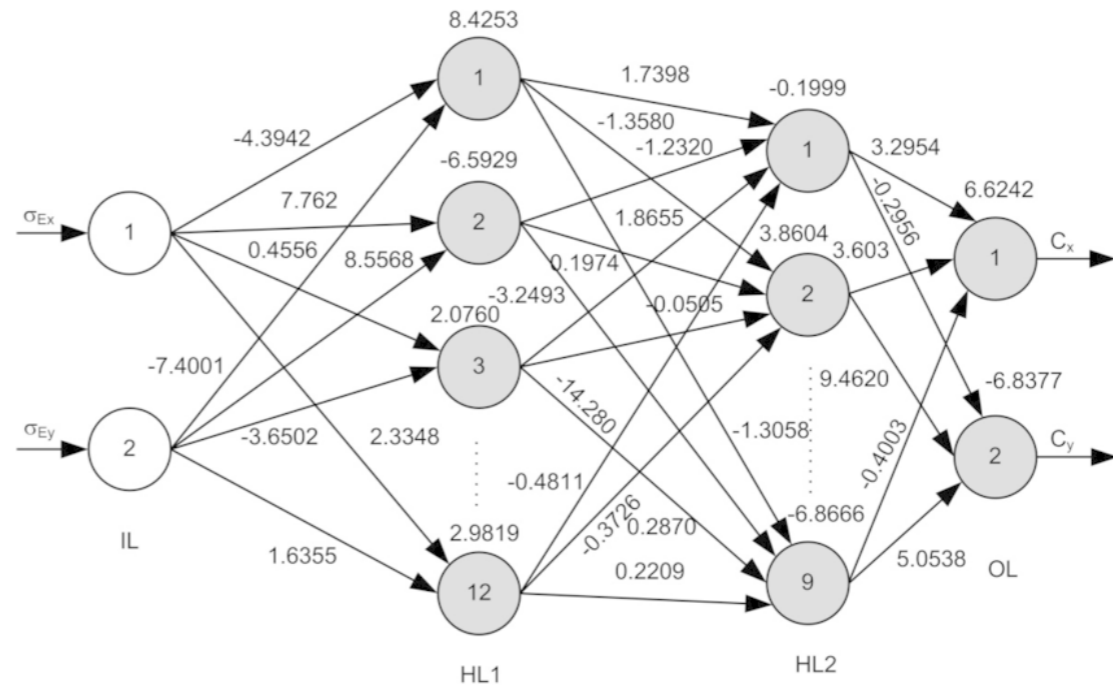
Any system that, given more data, performs increasingly better at some task

Framework / philosophy, not single method

Software 1.0



Software 2.0



Andrej Karpathy

[Follow](#)

Director of AI at Tesla. Previously Research Scientist at OpenAI and PhD student at Stanford. I like to train deep neural nets on large datasets.

Nov 11, 2017 · 7 min read

<https://medium.com/@karpathy/software-2-0-a64152b37c35>

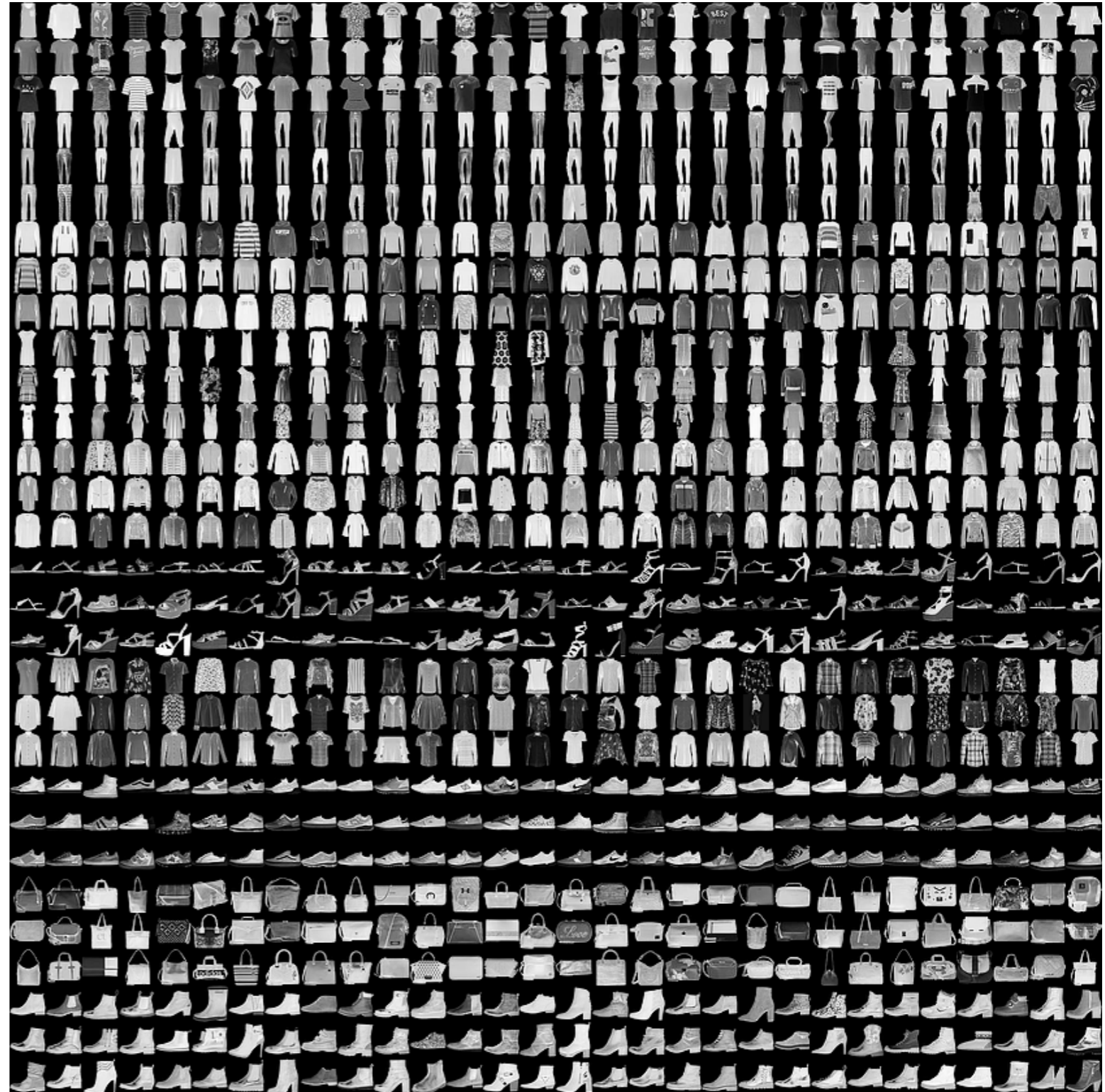
Basics of Machine Learning

Example of a Dataset – Fashion MNIST

10 categories (labels)

28x28 grayscale

70000 labeled images



Anatomy of a Dataset

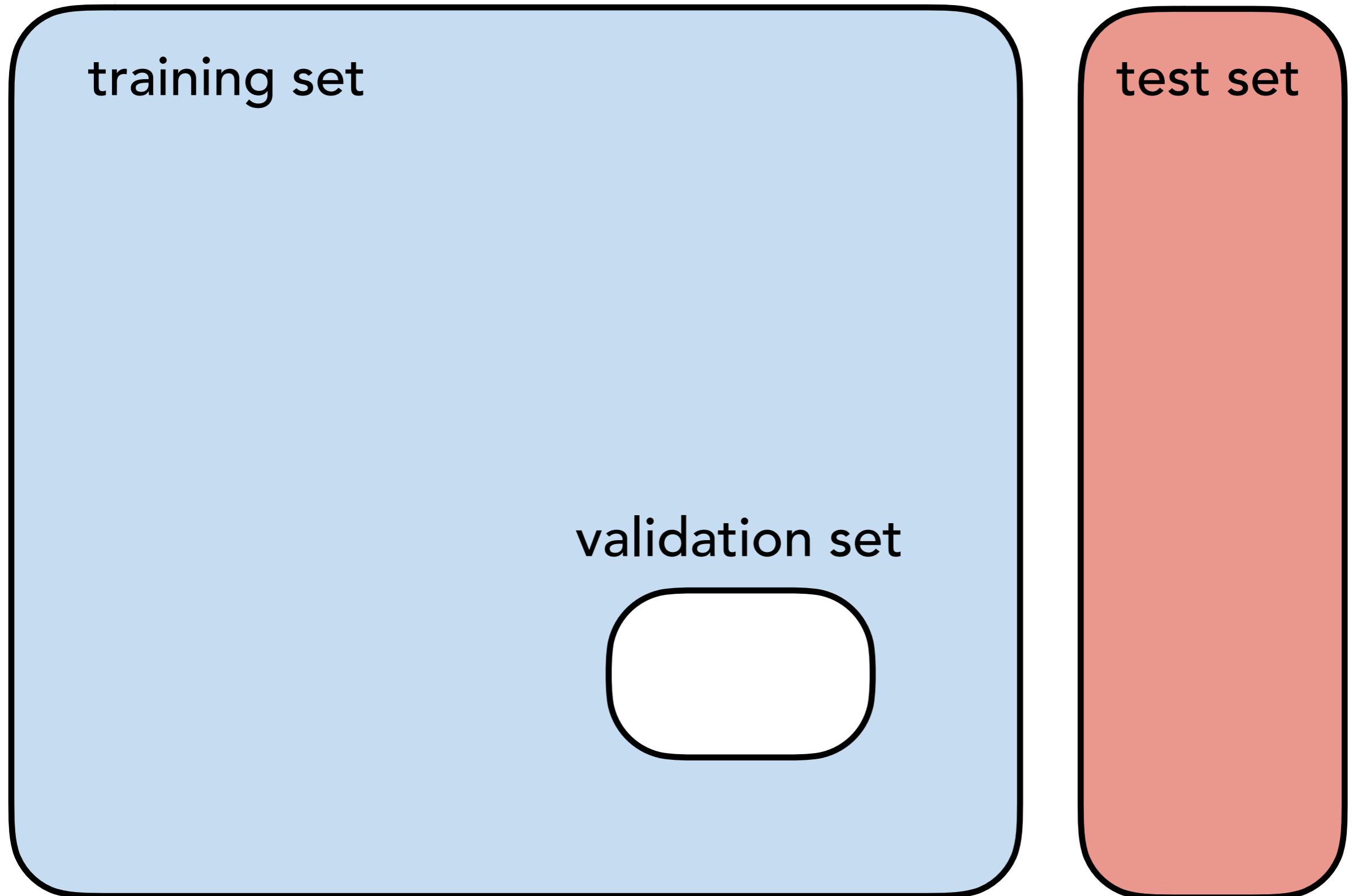
training set



The diagram illustrates the components of a dataset. It features two rounded rectangular boxes. The first box, on the left, is light blue and contains the text 'training set'. The second box, on the right, is light red and contains the text 'test set'. The training set box is significantly larger than the test set box, visually representing that the training set is a much larger portion of the dataset.

test set

Anatomy of a Dataset



Types of learning tasks:

- Supervised learning (labeled data)
- Unsupervised learning (unlabeled data)
- Reinforcement learning ('reward' data)

a priori knowledge

high

low

Supervised Learning

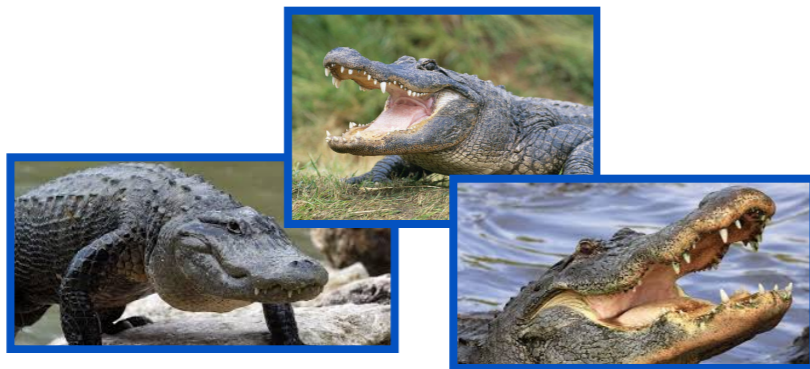
Given labeled training data (labels A and B)

Find *decision function* $f(\mathbf{x})$

$$f(\mathbf{x}) > 0 \quad \mathbf{x} \in A$$

$$f(\mathbf{x}) < 0 \quad \mathbf{x} \in B$$

Example: identify photos of **alligators** and **bears**



Supervised Learning

Typical strategy:

given training set $\{\mathbf{x}_j, y_j\}$, minimize cost function

$$C = \frac{1}{N_T} \sum_j (f(\mathbf{x}_j) - y_j)^2 \qquad y_j = \begin{cases} +1 & \mathbf{x}_j \in A \\ -1 & \mathbf{x}_j \in B \end{cases}$$

by varying adjustable params of f

Cost function measures distance of trial function $f(\mathbf{x}_j)$
from idealized "indicator" function y_j

Unsupervised Learning

Given unlabeled training data $\{\mathbf{x}_j\}$

- Find function $f(\mathbf{x})$ such that $f(\mathbf{x}_j) \simeq p(\mathbf{x}_j)$
- Find function $f(\mathbf{x})$ such that $|f(\mathbf{x}_j)|^2 \simeq p(\mathbf{x}_j)$
- Find data clusters and which data belongs to each cluster
- Discover reduced representations of data for other learning tasks (e.g. supervised)

Unsupervised Learning

Typical approach for inferring $p(\mathbf{x})$

Given data $\{\mathbf{x}_j\}$, maximize log likelihood

$$\mathcal{L} = \sum_j \log p(\mathbf{x}_j)$$

by varying p

Can view log likelihood as distance measure between

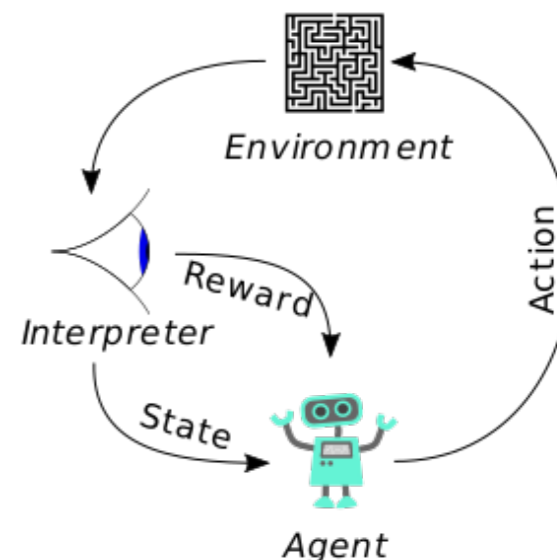
$$p(\mathbf{x}) \text{ and } p_{\text{data}}(\mathbf{x}) = \sum_j \delta(\mathbf{x} - \mathbf{x}_j)$$

("Kullback-Leibler divergence")

Reinforcement learning

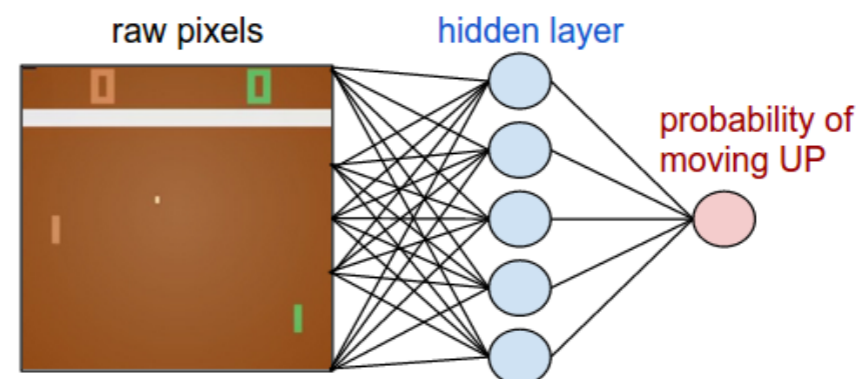
Many flavors, but have common features

- environment & agent with states s_n
- agent actions a_n
- reward $R(s_n)$ for being in state s_n



Goal: determine a policy $P(s_n) \longrightarrow a_n$,
best actions to maximize reward in fewest steps

Example: learning "Pong"
by observing screen state



General Philosophy of Machine Learning



General Philosophy of Machine Learning

- Solution to problem just some function $y(\mathbf{x})$



General Philosophy of Machine Learning

- Solution to problem just some function $y(\mathbf{x})$
- Parameterize very flexible functions $f(\mathbf{x})$
(prefer convenient over "correct")



General Philosophy of Machine Learning

- Solution to problem just some function $y(\mathbf{x})$
- Parameterize very flexible functions $f(\mathbf{x})$
(prefer convenient over "correct")
- Of all f that come closest to y for training data,
prefer the simplest f



Bias-Variance Tradeoff

Bias-Variance Tradeoff

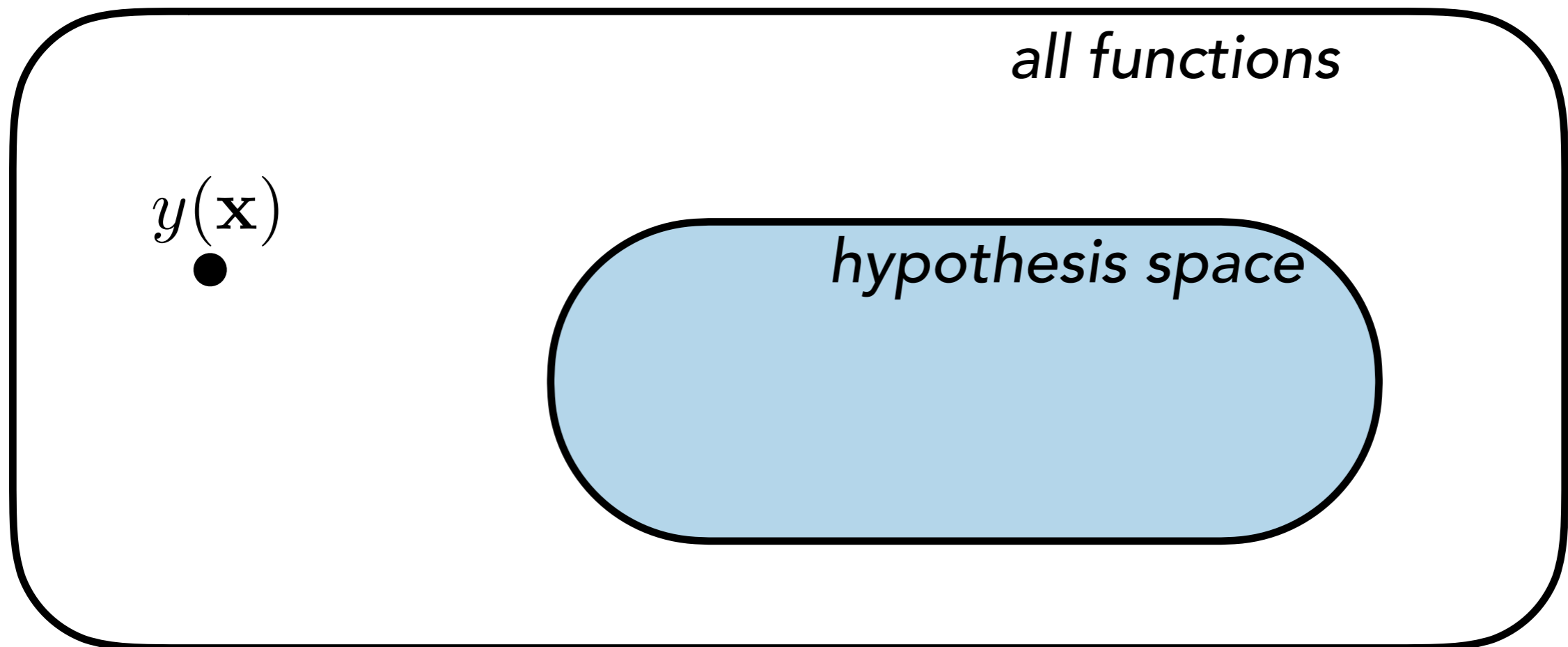
$y(\mathbf{x})$ – ideal solution function

all functions

$y(\mathbf{x})$
●

Bias-Variance Tradeoff

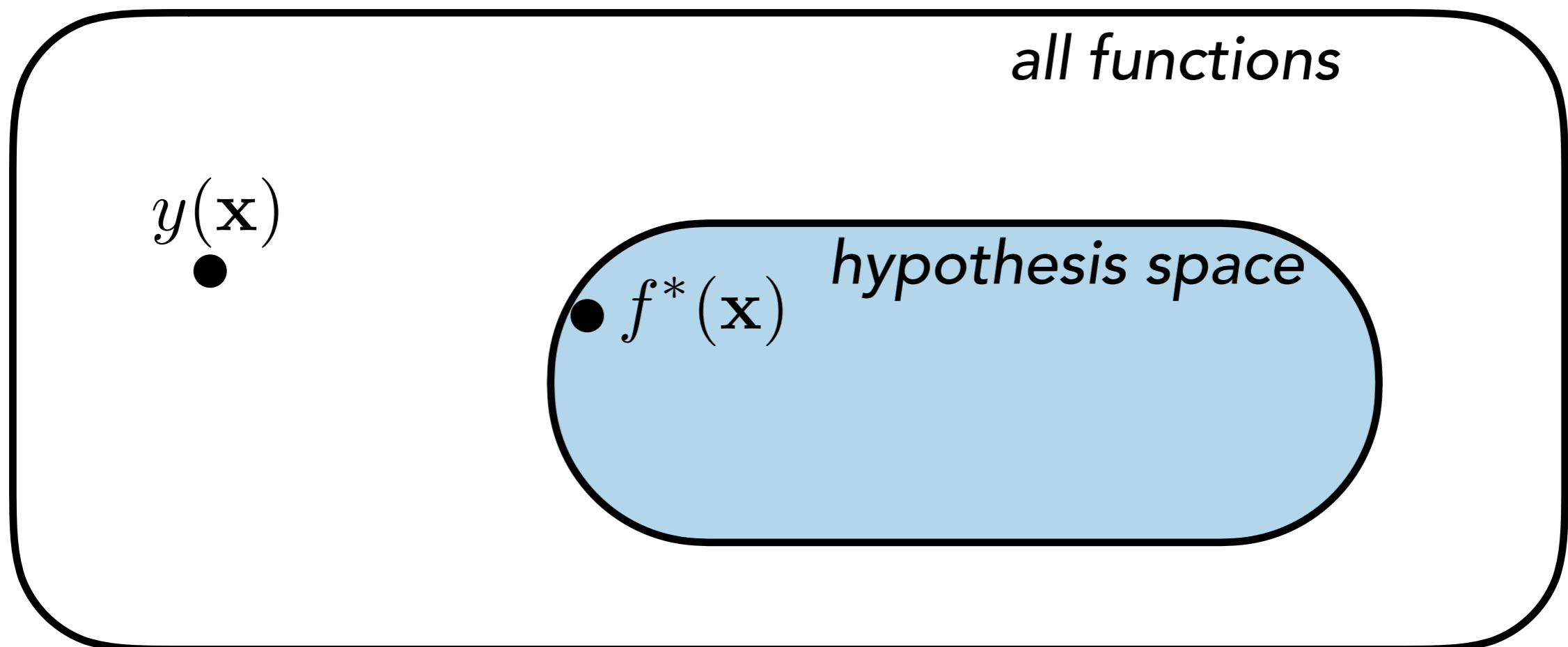
$y(\mathbf{x})$ – ideal solution function



Bias-Variance Tradeoff

$y(\mathbf{x})$ – ideal solution function

$f^*(\mathbf{x})$ – best possible hypothesis

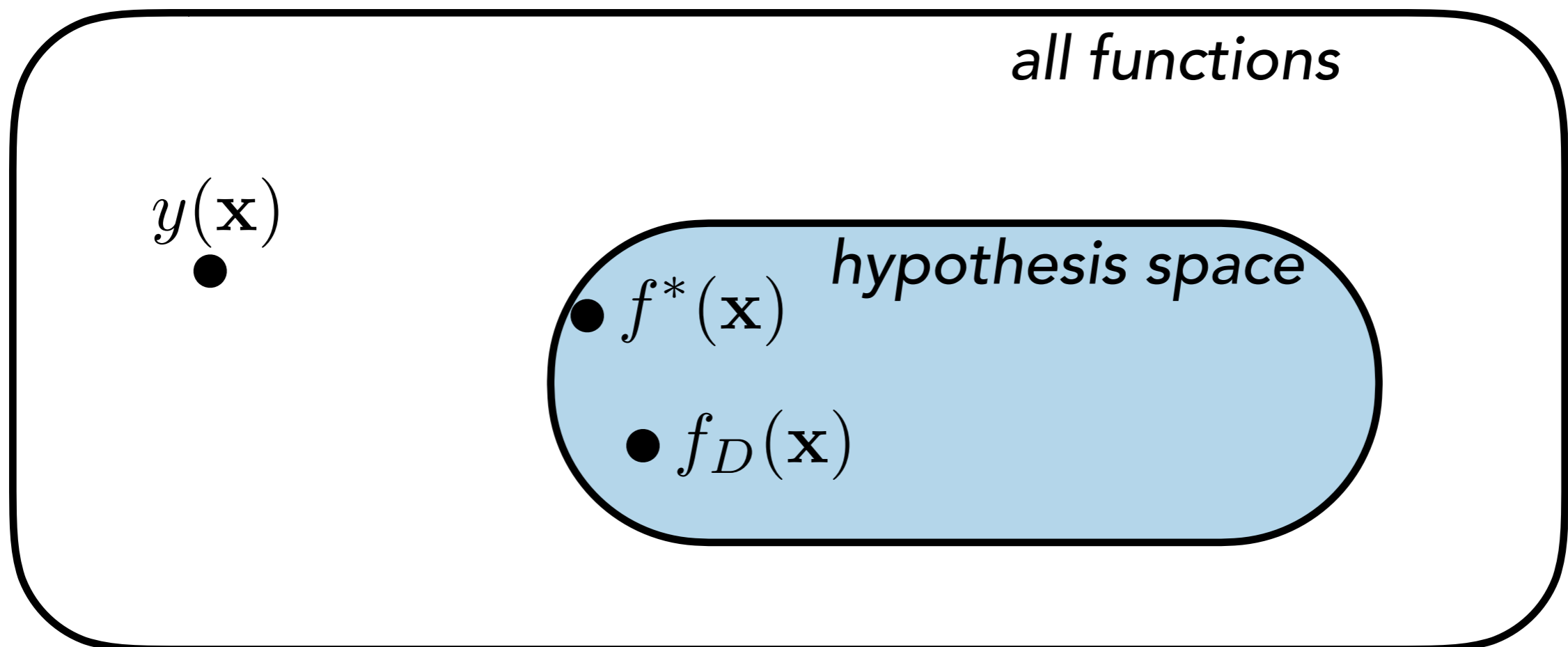


Bias-Variance Tradeoff

$y(\mathbf{x})$ – ideal solution function

$f^*(\mathbf{x})$ – best possible hypothesis

$f_D(\mathbf{x})$ – best hypothesis given training data

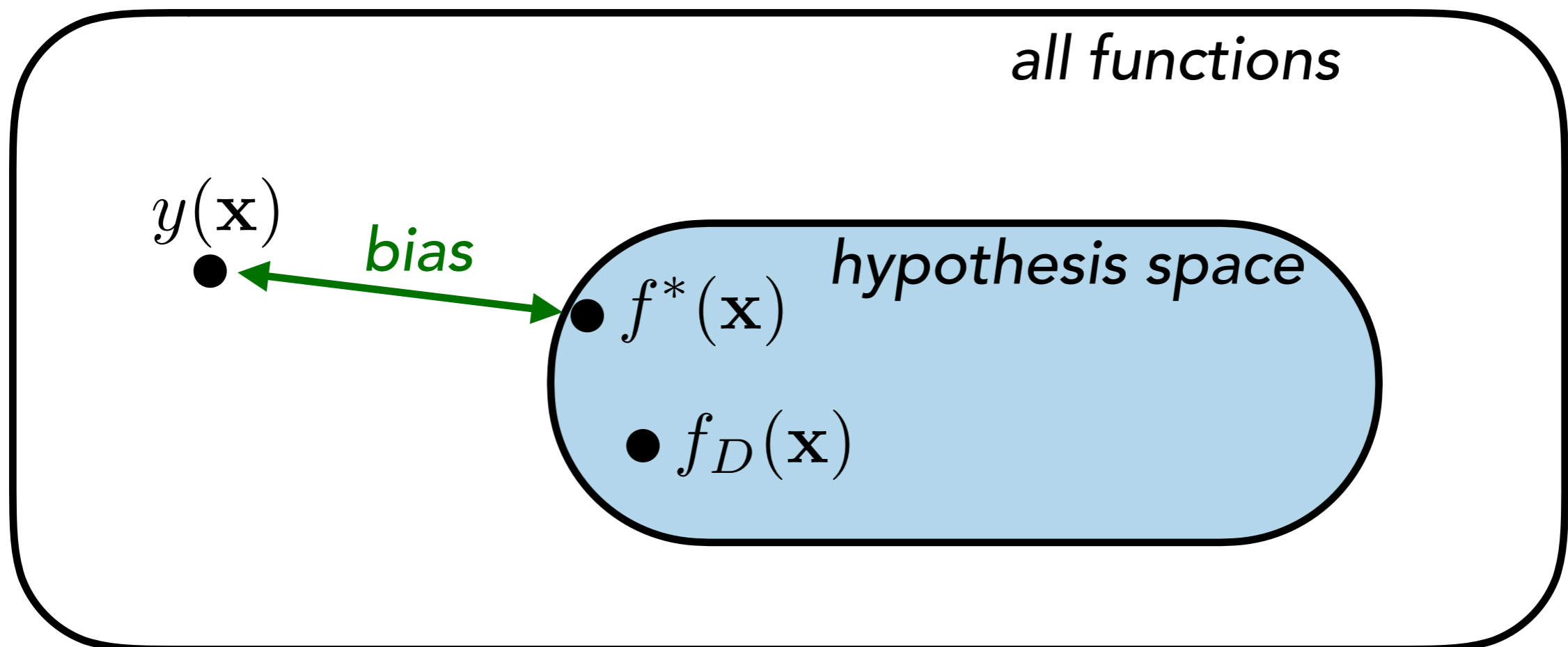


Bias-Variance Tradeoff

$y(\mathbf{x})$ – ideal solution function

$f^*(\mathbf{x})$ – best possible hypothesis

$f_D(\mathbf{x})$ – best hypothesis given training data

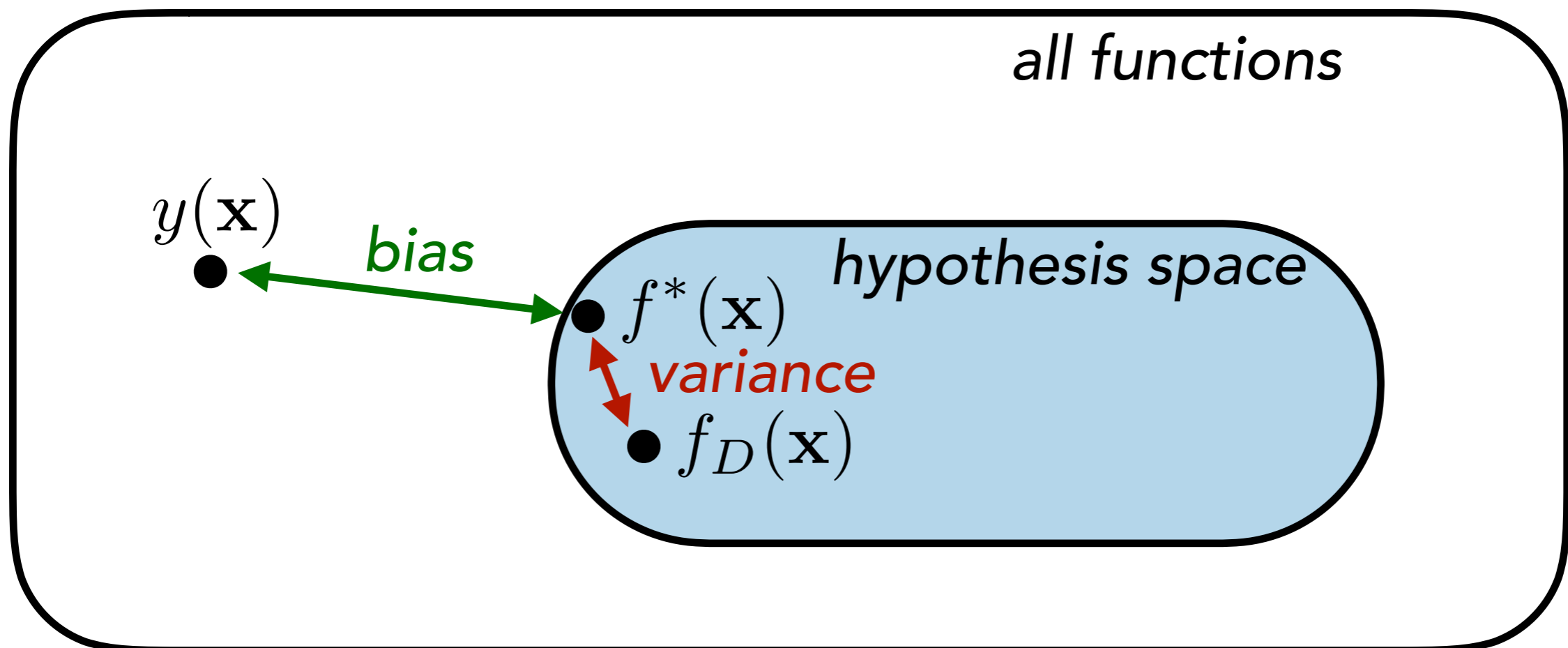


Bias-Variance Tradeoff

$y(\mathbf{x})$ – ideal solution function

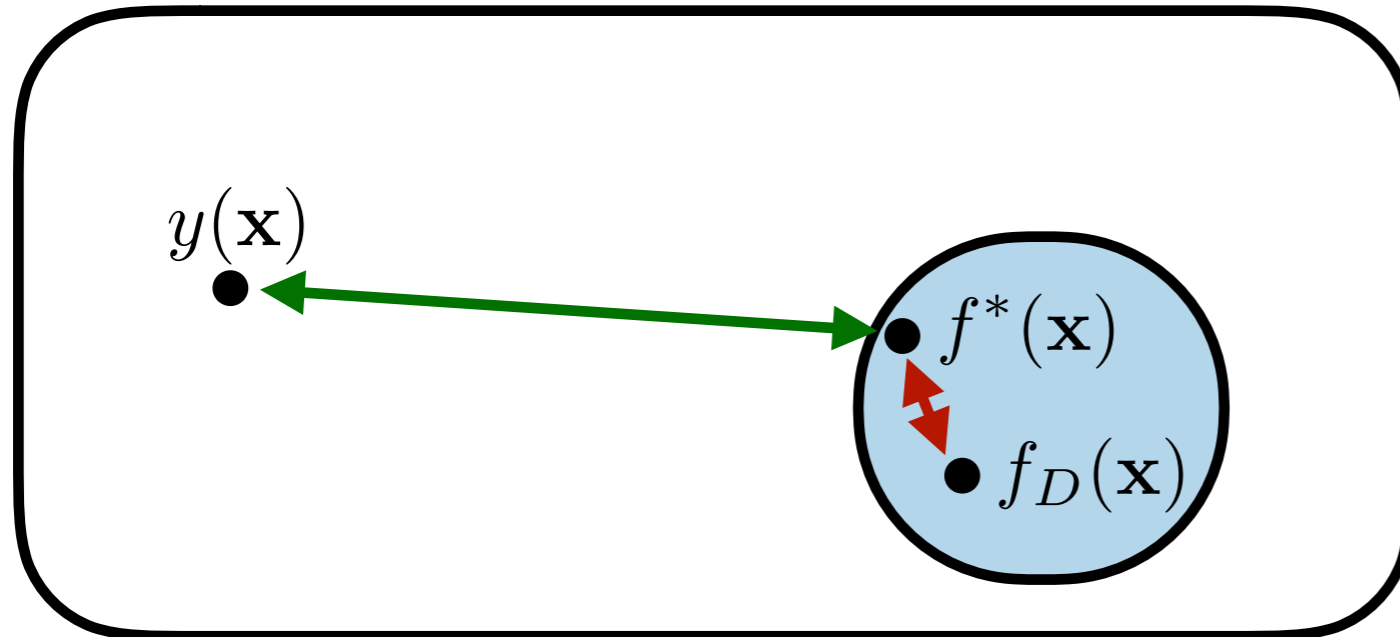
$f^*(\mathbf{x})$ – best possible hypothesis

$f_D(\mathbf{x})$ – best hypothesis given training data



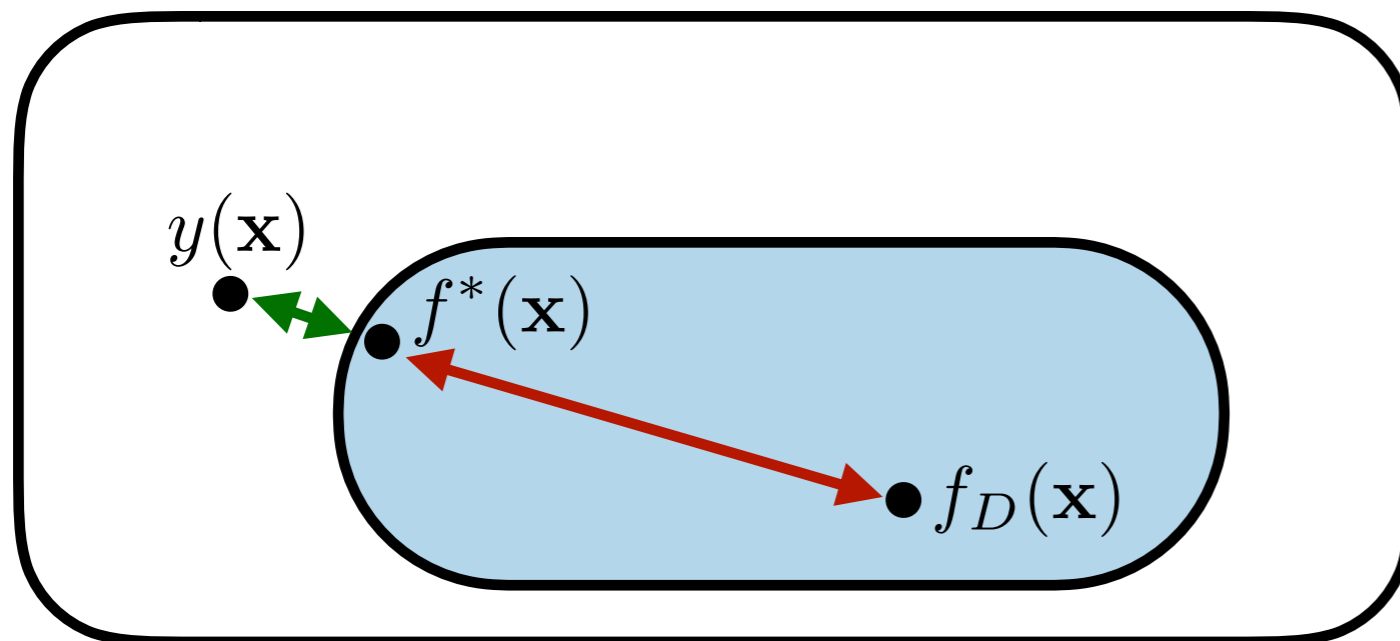
Bias-Variance Tradeoff

Two extreme situations



low variance: will generalize!

high bias: poor results



low bias: good result possible

low variance: might overfit

Model Architectures

Let's discuss the 3 most used types of models
(increasing complexity)

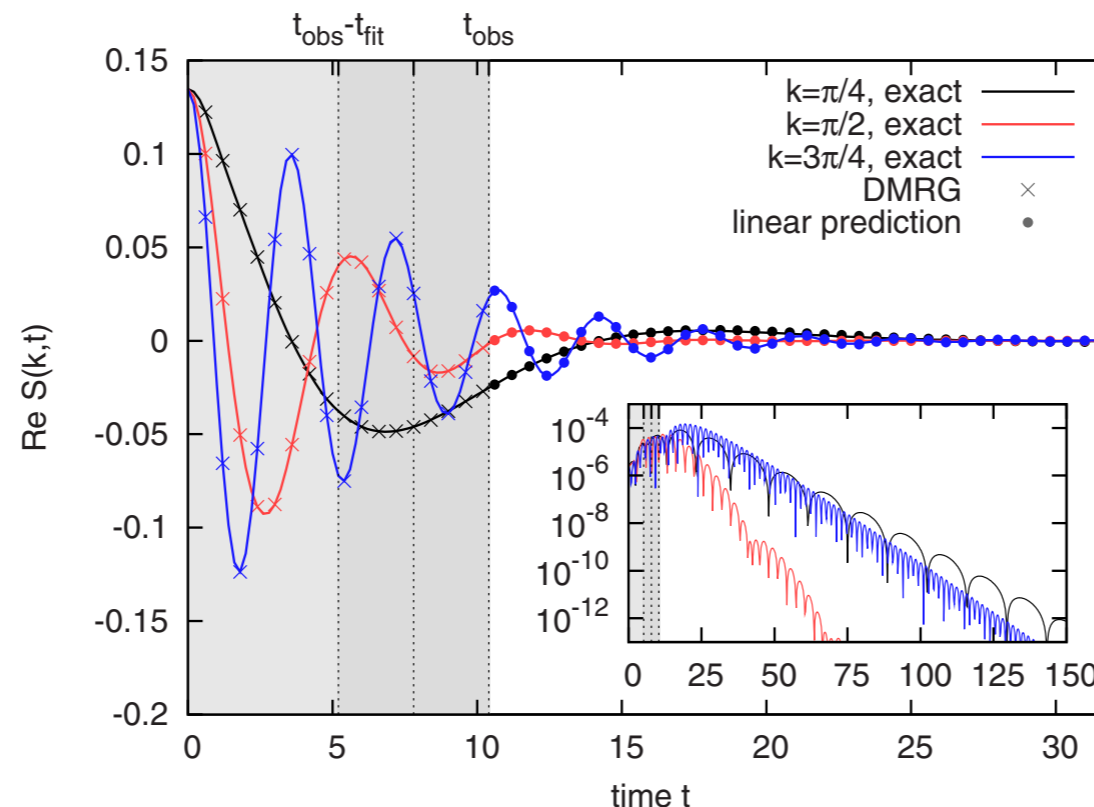
- The linear model
- Kernel learning / support vector machines
- Neural networks

The linear model

$$f(\mathbf{x}) = W \cdot \mathbf{x} + W_0$$

Where W and W_0 are the weights to be learned

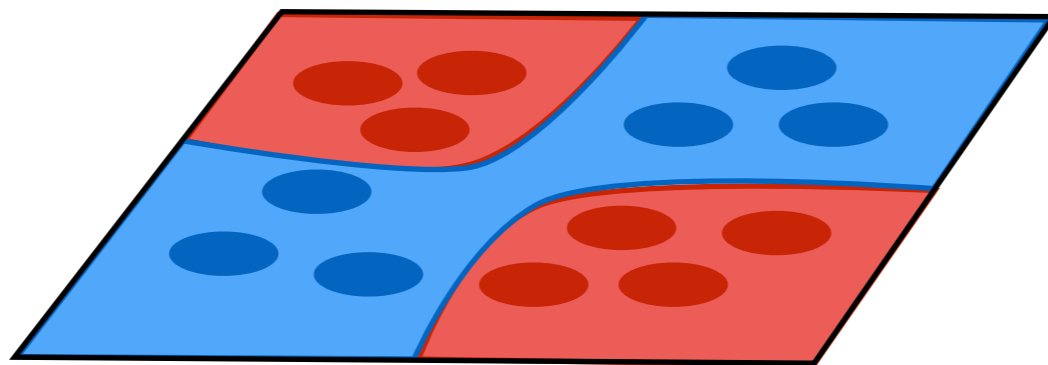
Can be surprisingly powerful, and a useful starting point



Barthel, Schollwöck, White, PRB 79, 245101

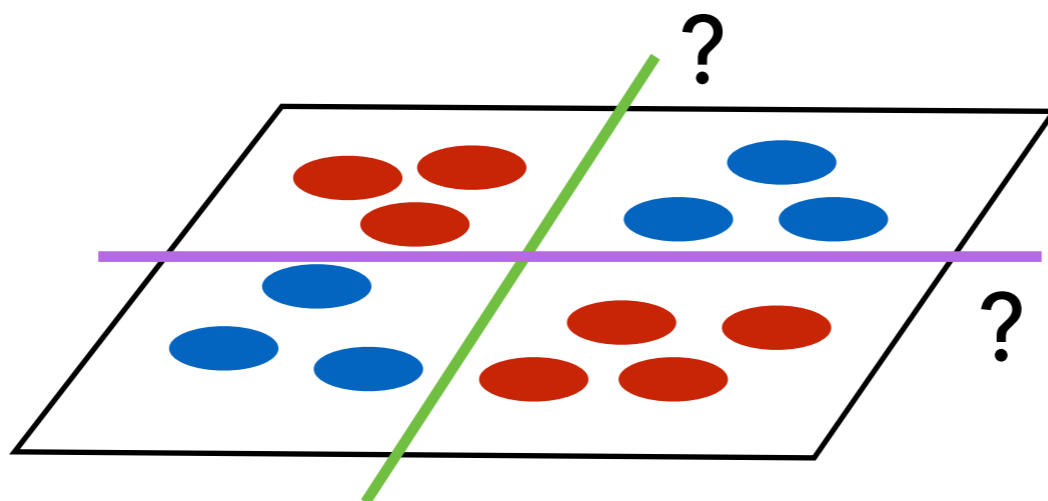
Kernel learning

Want $f(\mathbf{x})$ to separate classes, say



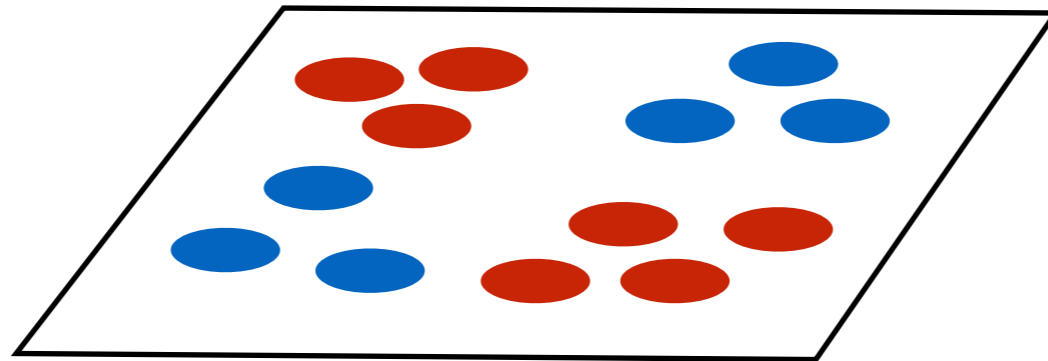
Linear classifier
may be insufficient

$$f(\mathbf{x}) = W \cdot \mathbf{x}$$



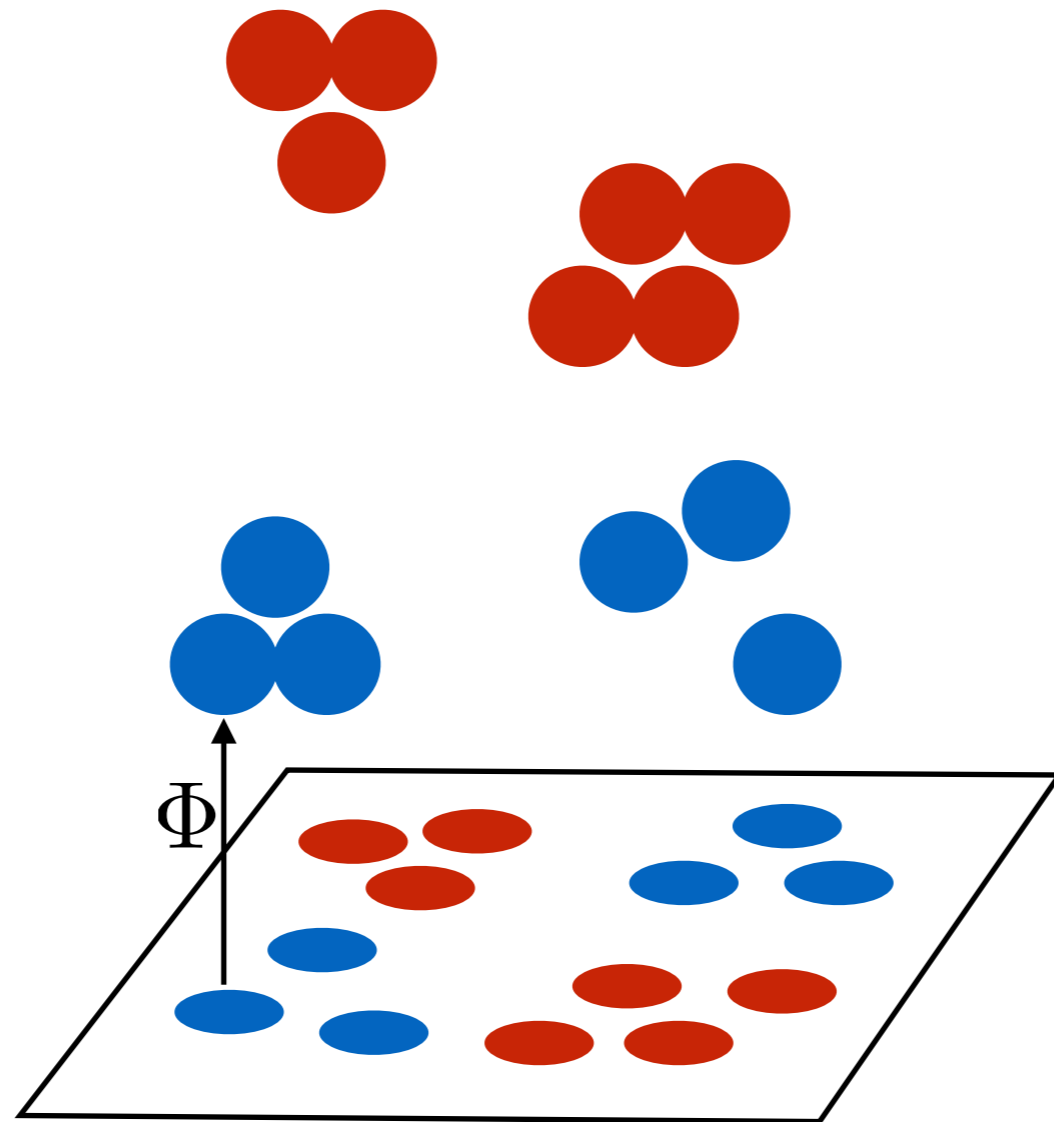
Kernel learning

Apply non-linear "feature map" $\mathbf{x} \rightarrow \Phi(\mathbf{x})$



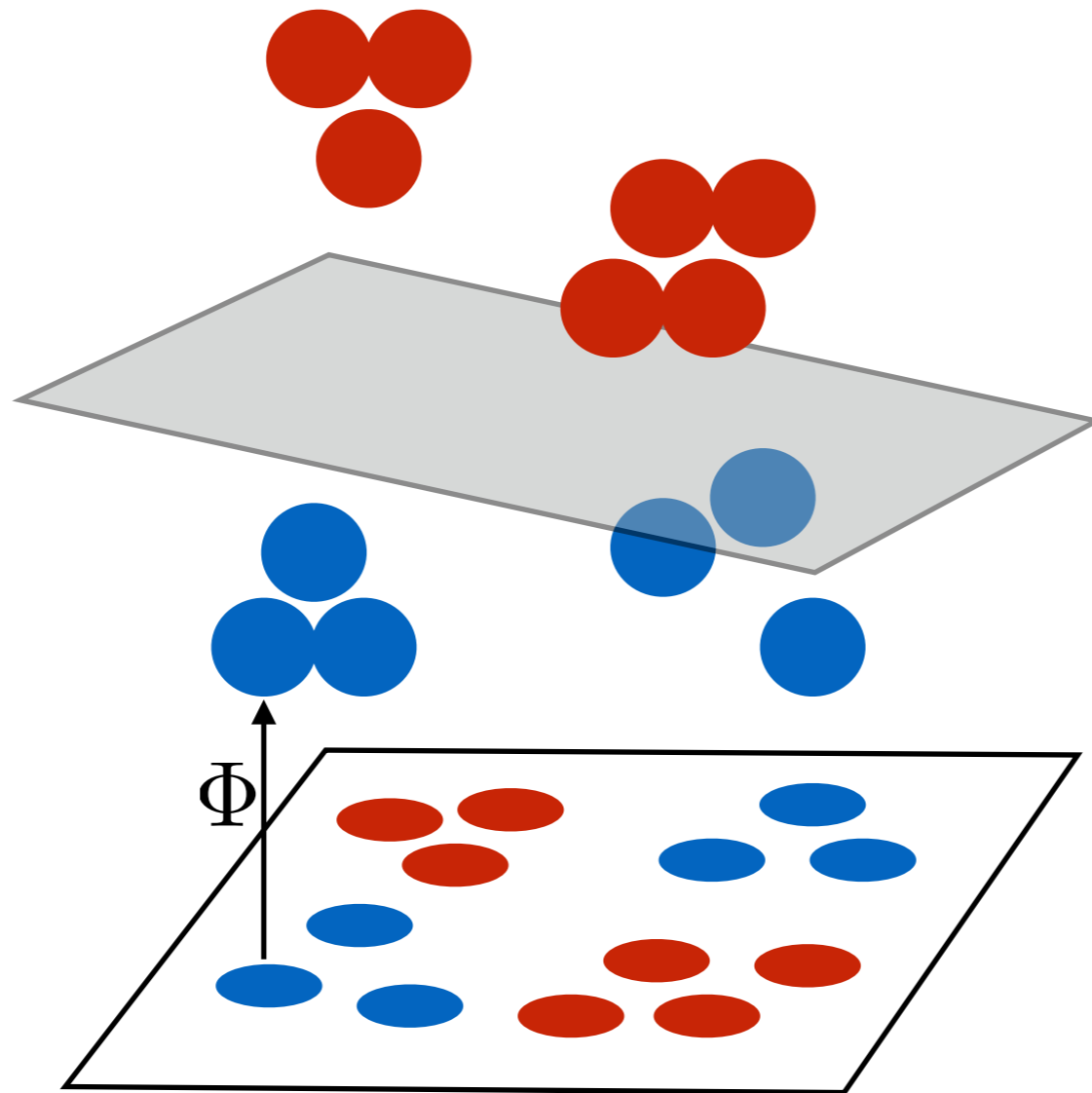
Kernel learning

Apply non-linear "feature map" $\mathbf{x} \rightarrow \Phi(\mathbf{x})$



Kernel learning

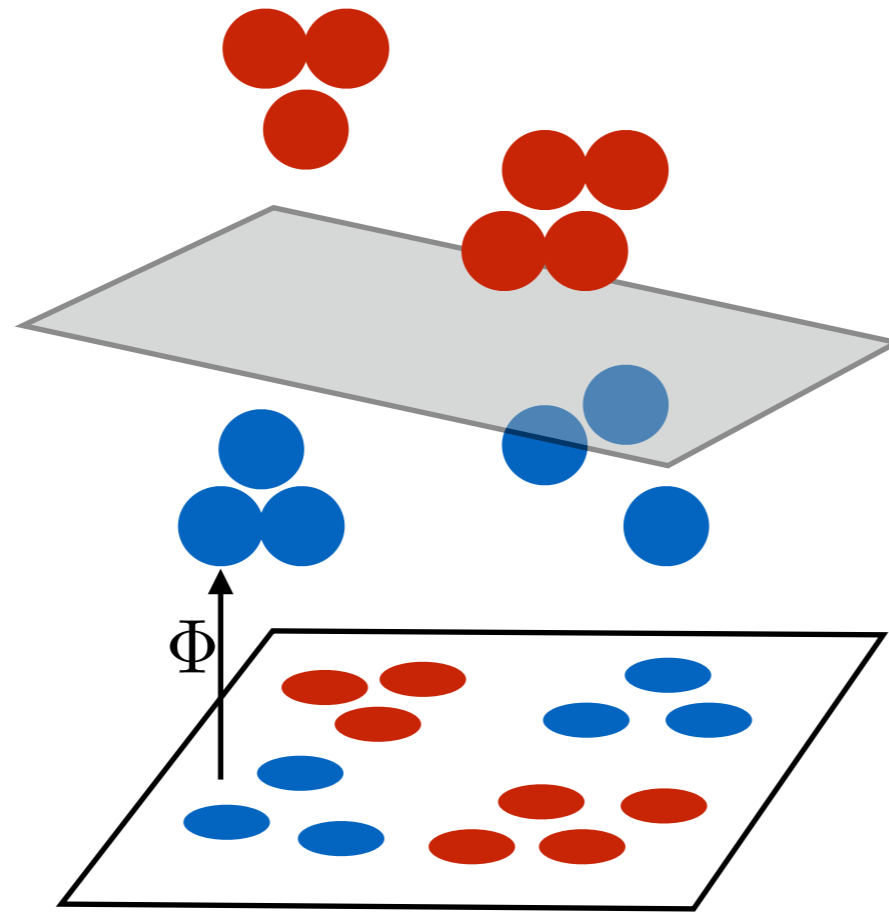
Apply non-linear "feature map" $\mathbf{x} \rightarrow \Phi(\mathbf{x})$



Decision function

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$$

Kernel learning



Decision function

$$f(\mathbf{x}) = W \cdot \Phi(\mathbf{x})$$

Linear classifier in *feature space*

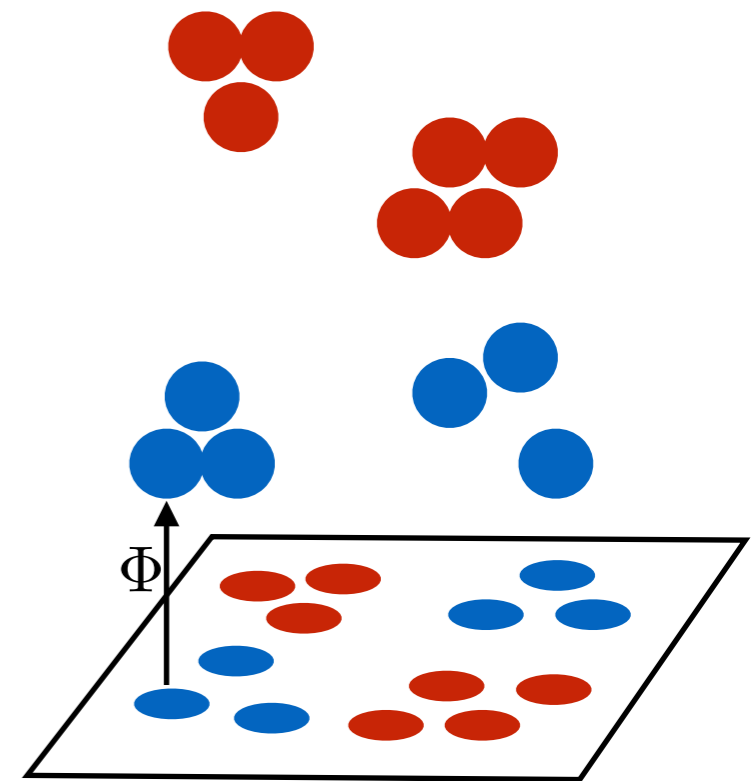
Kernel learning

Example of *feature map*

$$\mathbf{x} = (x_1, x_2, x_3)$$

$$\Phi(\mathbf{x}) = (1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3)$$

\mathbf{x} is "lifted" to feature space



Kernel learning

Technical notes:

Kernel learning

Technical notes:

- Also called "support vector machine" when using a particular choice of cost function
- Name "kernel learning" comes from idea that $\Phi(\mathbf{x})$ may be too high dimensional, yet $K_{ij} = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$ may be efficiently computable, enough to optimize
- Very generally, optimal weights have the form

$$W = \sum_j \alpha_j \Phi(\mathbf{x}_j)$$

a result known as the "representer theorem"

Kernel learning

Kernel learning still popular among academics & for certain applications (e.g. life sciences)

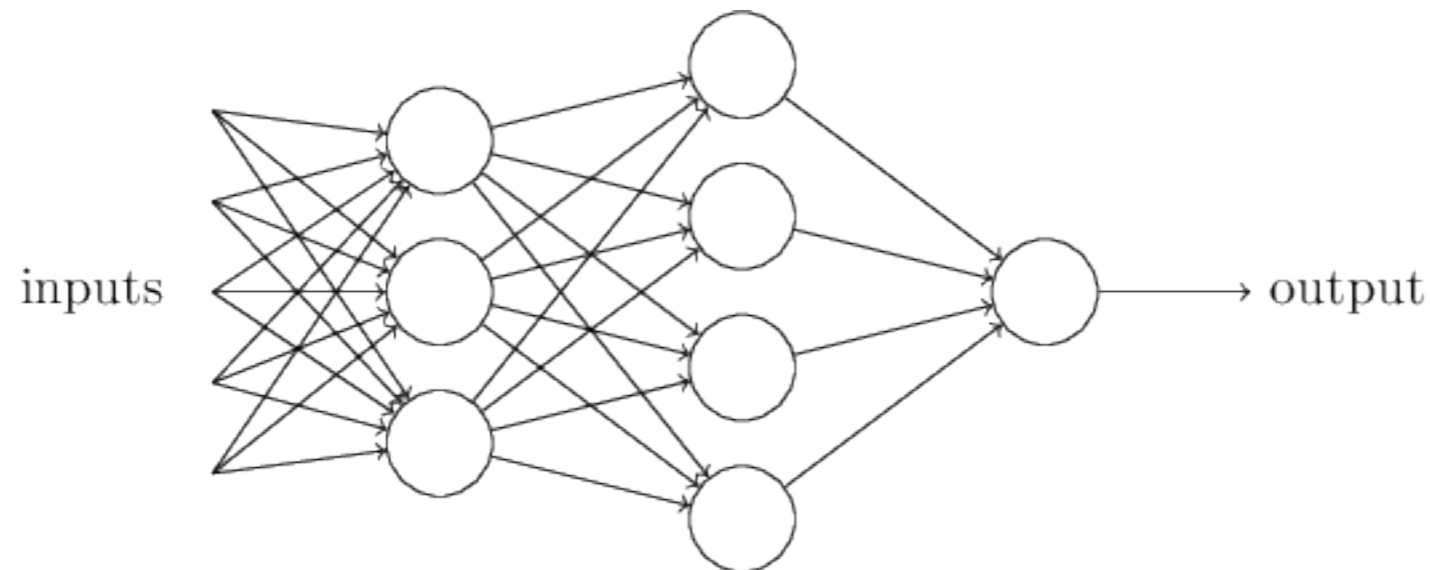
But "kernelization" approach scales as N^3 where N is size of training set – very costly!

Thus kernel methods not popular with engineers

Tomorrow: learning kernel models with tensor network weights

Neural networks

Current favorite of M.L. engineers



Often notated diagrammatically
(not a tensor diagram!)

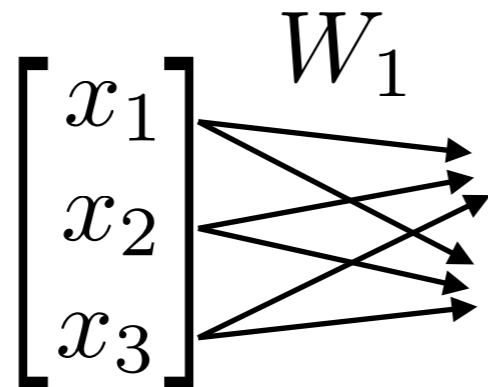
Neural networks

Actually very simple: compute a function $f(\mathbf{x})$ as

Neural networks

Actually very simple: compute a function $f(\mathbf{x})$ as

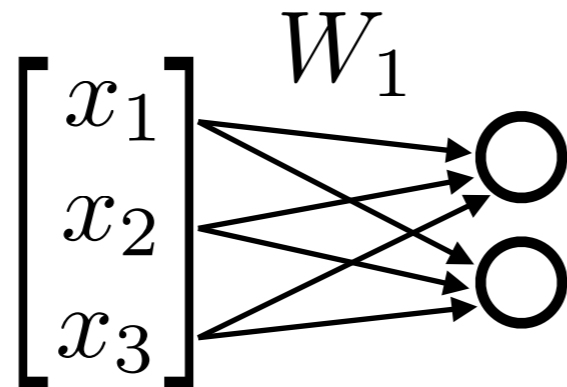
- Multiply input \mathbf{x} by rectangular "weight" matrix W_1



Neural networks

Actually very simple: compute a function $f(\mathbf{x})$ as

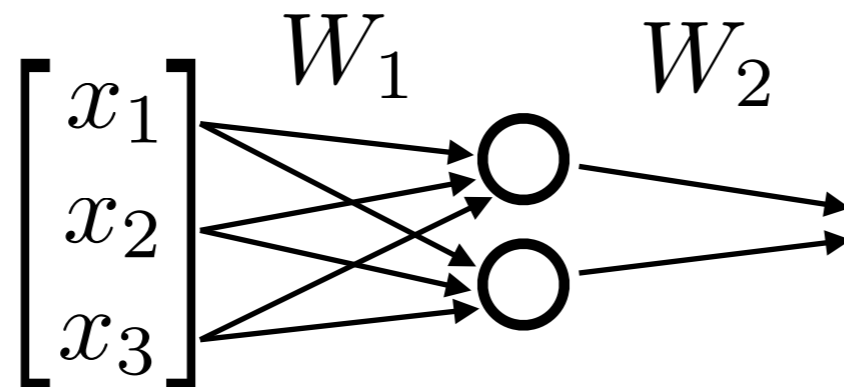
- Multiply input \mathbf{x} by rectangular "weight" matrix W_1
- Point-wise evaluate components of $\mathbf{x}' = W_1\mathbf{x}$ by some non-linear function [e.g. $\sigma(x'_j) = 1/(1 - e^{x'_j - b})$]



Neural networks

Actually very simple: compute a function $f(\mathbf{x})$ as

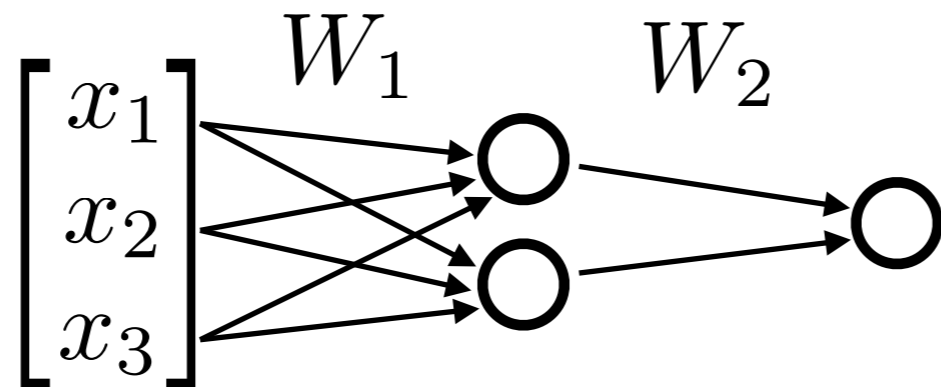
- Multiply input \mathbf{x} by rectangular "weight" matrix W_1
- Point-wise evaluate components of $\mathbf{x}' = W_1\mathbf{x}$ by some non-linear function [e.g. $\sigma(x'_j) = 1/(1 + e^{-x'_j})$]
- Multiply result by second weight matrix W_2



Neural networks


Actually very simple: compute a function $f(\mathbf{x})$ as

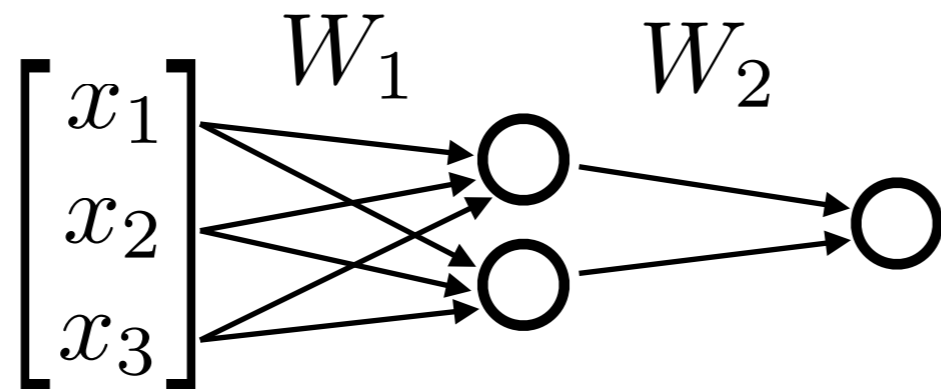
- Multiply input \mathbf{x} by rectangular "weight" matrix W_1
- Point-wise evaluate components of $\mathbf{x}' = W_1\mathbf{x}$ by some non-linear function [e.g. $\sigma(x'_j) = 1/(1 + e^{-x'_j})$]
- Multiply result by second weight matrix W_2
- Plug new components into non-linearities, etc.



Neural networks

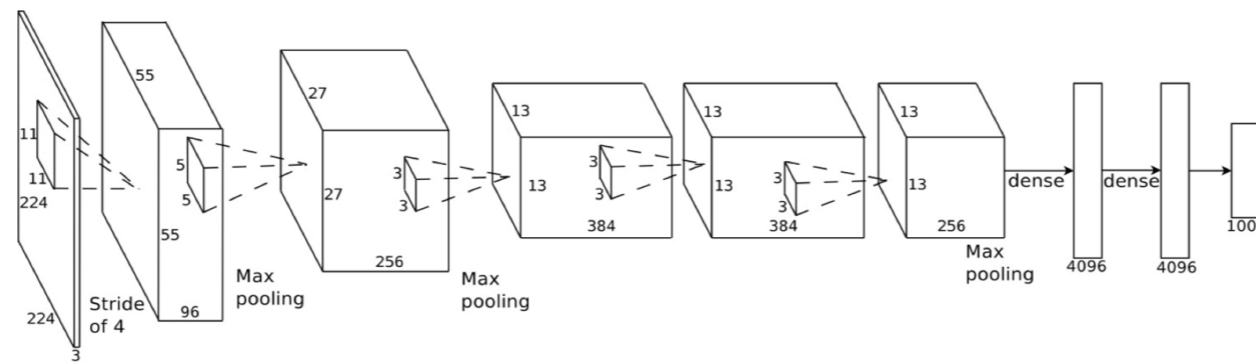
Additional facts:

- Non-linearities $\sigma(x)$ called "neurons"
- Other neurons include tanh and ReLU 
- Neural net with more than one weight matrix is "deep"
- Number of neurons is arbitrary, but with enough can represent any function



Neural networks

Many successful neural nets include "convolutional layers"
These have sparser weight layers with few parameters.



Recent upsurge of neural nets since 2012 (ImageNet paper)

"Deep learning" often associated with 3 researchers:



Yann LeCun (Facebook)



Geoff Hinton (Vector/Google)



Yoshua Bengio (Montreal)

Other model types

Graphical models

very similar to tensor networks, except

- always interpreted as probability
- non-negative parameters only

Other model types

Graphical models

very similar to tensor networks, except

- always interpreted as probability
- non-negative parameters only

Boltzmann machines

identical to random-bond classical Ising ($T=1$)

J_{ij} values learnable parameters

generate data by sampling subset of spins

Other model types

Graphical models

very similar to tensor networks, except

- always interpreted as probability
- non-negative parameters only

Boltzmann machines

identical to random-bond classical Ising ($T=1$)

J_{ij} values learnable parameters

generate data by sampling subset of spins

Decision trees

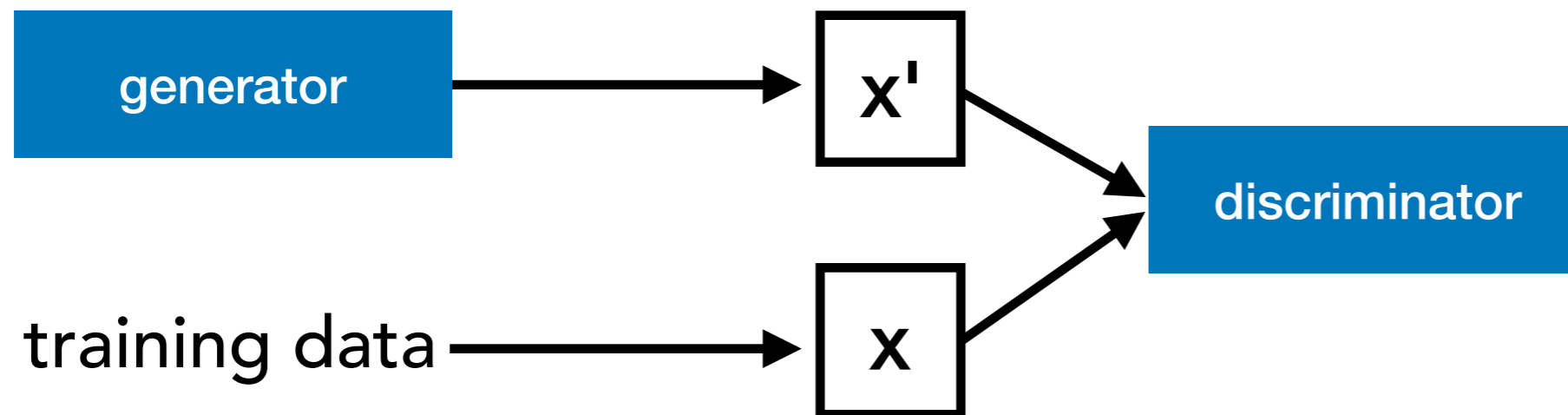
make decisions about input by taking
forking paths

Recent Developments

GANs

Generative Adversarial Networks
(Goodfellow et al. arxiv:1406.2661)

Train two models simultaneously



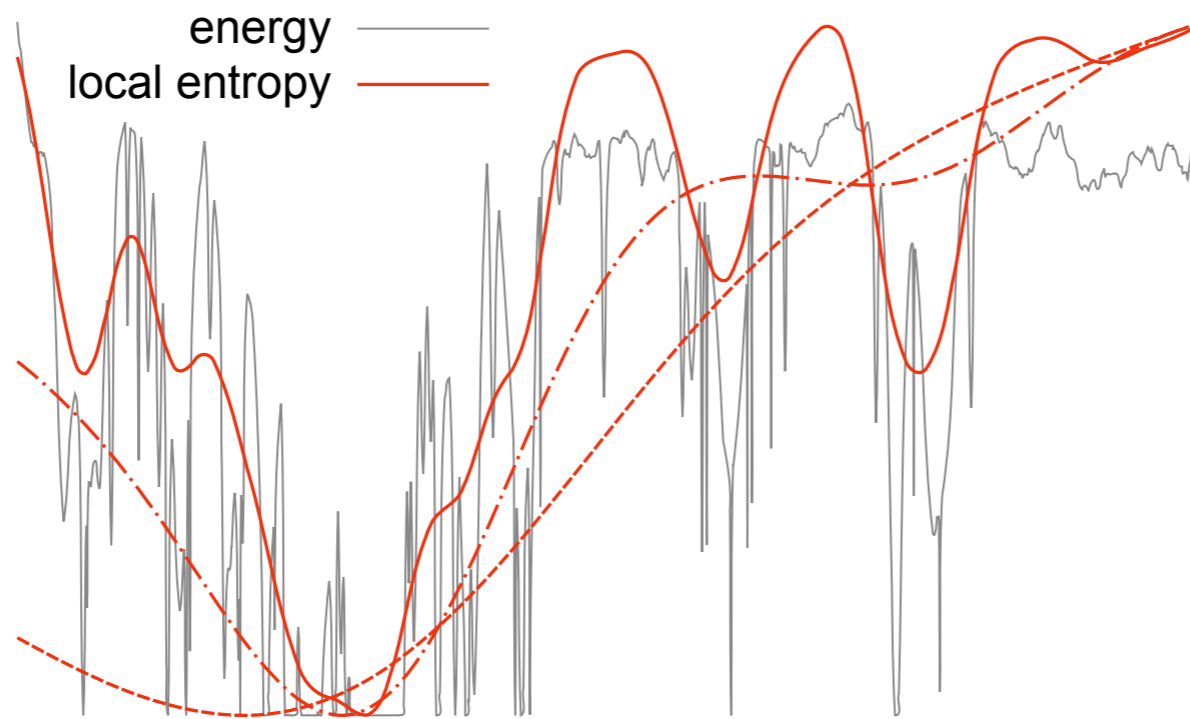
Generator produces superior results compared to other methods



Radford et al., arxiv:1511.06434

Progress in Understanding Generalization

Neural nets empirically generalize even when very expressive – why?



Baldassi et al. arxiv:1605.06444

Reason may be training finds broad local minima
versus narrow global minimum

Intuitively broader minima (w/ more neighbors) are more "robust"
to changes in data

FALKON Algorithm

Kernel methods have strong theoretical guarantees,
but scale worse than neural nets (N^3 training time, N^2 memory)

Algorithm 1 MATLAB code for FALKON. It requires $O(nMt + M^3)$ in time and $O(M^2)$ in memory. See Sect. A and Alg. 2 in the appendixes for the complete algorithm.

Input: Dataset $X = (x_i)_{i=1}^n \in \mathbb{R}^{n \times D}$, $\hat{y} = (y_i)_{i=1}^n \in \mathbb{R}^n$, centers $C = (\tilde{x}_j)_{j=1}^M \in \mathbb{R}^{M \times D}$, KernelMatrix computing the kernel matrix given two sets of points, regularization parameter λ , number of iterations t .

Output: Nyström coefficients α .

```
function alpha = FALKON(X, C, Y, KernelMatrix, lambda, t)
    n = size(X,1); M = size(C,1); KMM = KernelMatrix(C,C);
    T = chol(KMM + eps*M*eye(M));
    A = chol(T*T'/M + lambda*eye(M));

    function w = KnM_times_vector(u, v)
        w = zeros(M,1); ms = ceil(linspace(0, n, ceil(n/M)+1));
        for i=1:ceil(n/M)
            Kr = KernelMatrix( X(ms(i)+1:ms(i+1),:), C );
            w = w + Kr'*(Kr*u + v(ms(i)+1:ms(i+1),:));
        end
    end

    BHB = @(u) A'\'(T'\'(KnM_times_vector(T\'(A\'u), zeros(n,1))/n) + lambda*(A\'u));
    r = A'\'(T'\'KnM_times_vector(zeros(M,1), Y/n));
    alpha = T\'(A\'conjgrad(BHB, r, t));
end
```

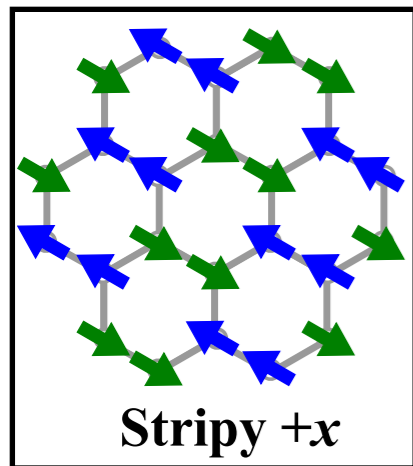
Rudi, Carratino, Rosasco, arxiv:1705.10958

Through combination of subsampling data & preconditioned
conj gradient, reduce to $N\sqrt{N}$ time, N memory

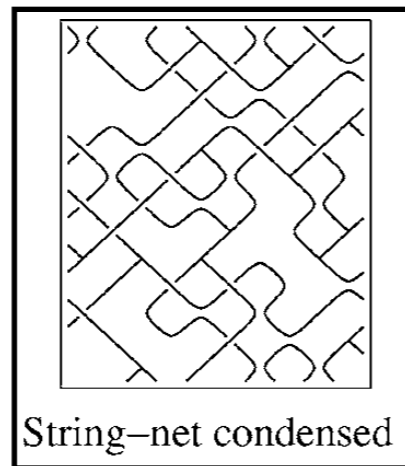
Results competitive with best neural net, can do ImageNet!

Selected Physics Applications

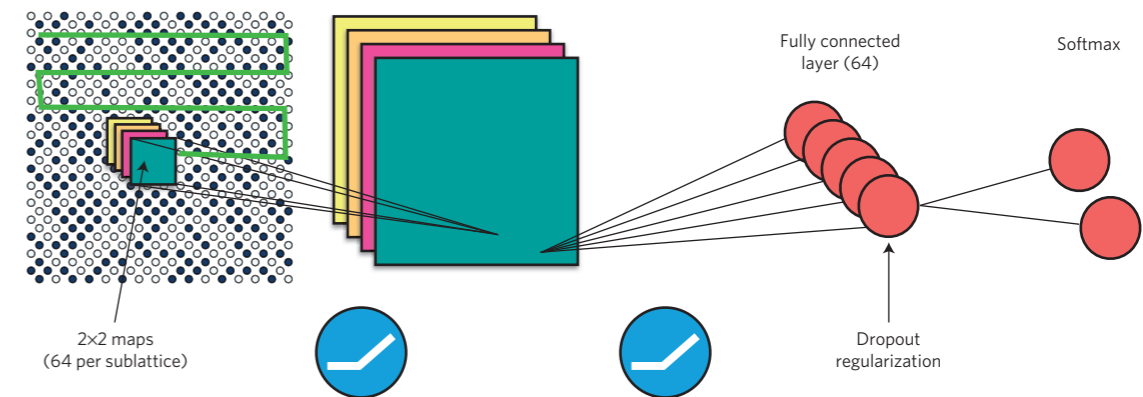
Phase recognition



Friends:
Lev Landau
Werner Heisenberg



Friends:
Michael Levin
Xiao-Gang Wen



View Monte Carlo configurations as input data,
train model (supervised or unsupervised) to distinguish phases

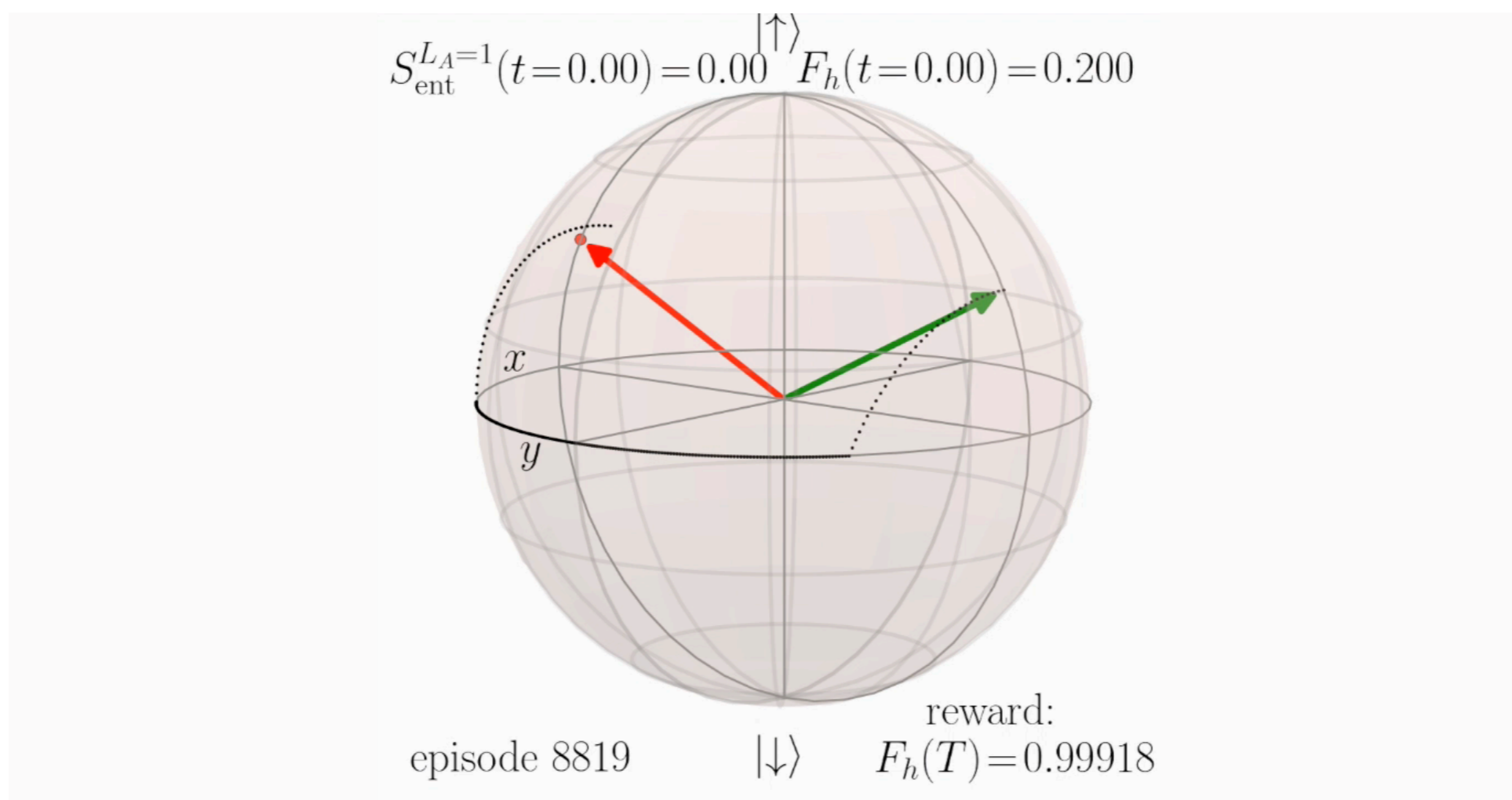
Some relevant papers:

- Carrasquilla, Melko, Nature Phys. (2017) [supervised]
- Wang, PRB 94, 195105 [unsupervised]
- Broecker, Carrasquilla, Melko, Trebst Scientific Reports 7, 8823 (2017) [from aux. field QMC]
- Broecker, Assaad, Trebst arxiv:1707.00663 [unsupervised]
- ... and quite a few others ...

Learning to Control Quantum Systems

How to apply time-dependent field to quantum system and reach some target state?

Treat fidelity as "reward" and train reinforcement learning agent to work out best protocol



Many Other Creative Ideas

Learning quantum Monte Carlo updates

J. Liu, Y. Qi, et al. arxiv:1610.03137

L. Huang, L. Wang, arxiv:1610.02746

L. Wang, arxiv:1702.08586

H. Shen, J. Liu, L. Fu, arxiv:1801.01127

Many Other Creative Ideas

Learning quantum Monte Carlo updates

J. Liu, Y. Qi, et al. arxiv:1610.03137

L. Huang, L. Wang, arxiv:1610.02746

L. Wang, arxiv:1702.08586

H. Shen, J. Liu, L. Fu, arxiv:1801.01127

Neural Net Representations of Wavefunctions

G. Carleo, M. Troyer, arxiv:1606.02318

D. Deng, X. Li, S. Das Sarma, arxiv:1609.09060, arxiv: 1701.04844

S. Clark, arxiv:1710.03545

Many Other Creative Ideas

Learning quantum Monte Carlo updates

J. Liu, Y. Qi, et al. arxiv:1610.03137

L. Huang, L. Wang, arxiv:1610.02746

L. Wang, arxiv:1702.08586

H. Shen, J. Liu, L. Fu, arxiv:1801.01127

Neural Net Representations of Wavefunctions

G. Carleo, M. Troyer, arxiv:1606.02318

D. Deng, X. Li, S. Das Sarma, arxiv:1609.09060, arxiv: 1701.04844

S. Clark, arxiv:1710.03545

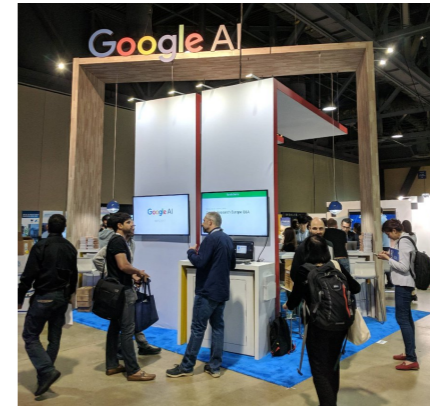
Learning Density Functionals

J. Snyder, et al., arxiv:1112.5441

F. Brockherde, et al., arxiv:1609.02815

L. Li, et al., arxiv:1609.03705

Machine Learning Research Culture



One sub-community is academic: papers often involve theorems

Another community is engineering-oriented: papers focus on results, developments are intuitive/faddish

Conference talks/posters valued above journal articles

Strong industry ties: Google, Microsoft, etc. have booths at conferences, grad students poached often

Recommended Resources

- Online book by Michael Nielsen (quant. computing author)
<http://neuralnetworksanddeeplearning.com>
- Caltech Lectures by Yaser Abu-Mostafa CS 156
Available on YouTube. Companion book "Learning from Data"
- Upcoming M.L. review article by Pankaj Mehta, David Schwab
aimed at physicists
- TensorFlow examples (MNIST demo)
- Blogs of Chris Olah and Andrej Karpathy