Machine Learning Beyond Physics

from a Physicist's Perspective

Jonathan Walsh

How do we do physics?

model building:

Parameters → Model → Data

(kinda) natural — *(well) motivated* — *extant/future*

calculations:

Observable → Model → Result

defined — *known* — *unknown*

simulations:

Phase Space → Model → Data

tractable — *approximated* — *realistic*

A core tenet of particle physics

We can *understand* the model

My own experience in physics (QCD-side)

grad school LHC Olympics — jet algorithms, jet substructure — SCET for jets

phenomenological QCD → theoretical foundations

postdoc non-global logarithms — high-precision Monte Carlo — resummation — fixed order calculations —

increasingly technical →

Evolution of my research program largely governed by the process:

1. finding interesting problems in QCD
2. **understanding the underlying theory**
3. applying it to more interesting problems
4. rinse and repeat

Modeling beyond physics

Beyond physics, the model may be too complex or just unknown.

A first-principles approach *does not apply* in these cases.

We need tools capable of model inference that can learn and utilize relevant information in the data

2 common approaches: *statistical* and *deterministic* modeling

- statistical : derive the form of the model
- deterministic : input the form of the model

Reframing the core tenet

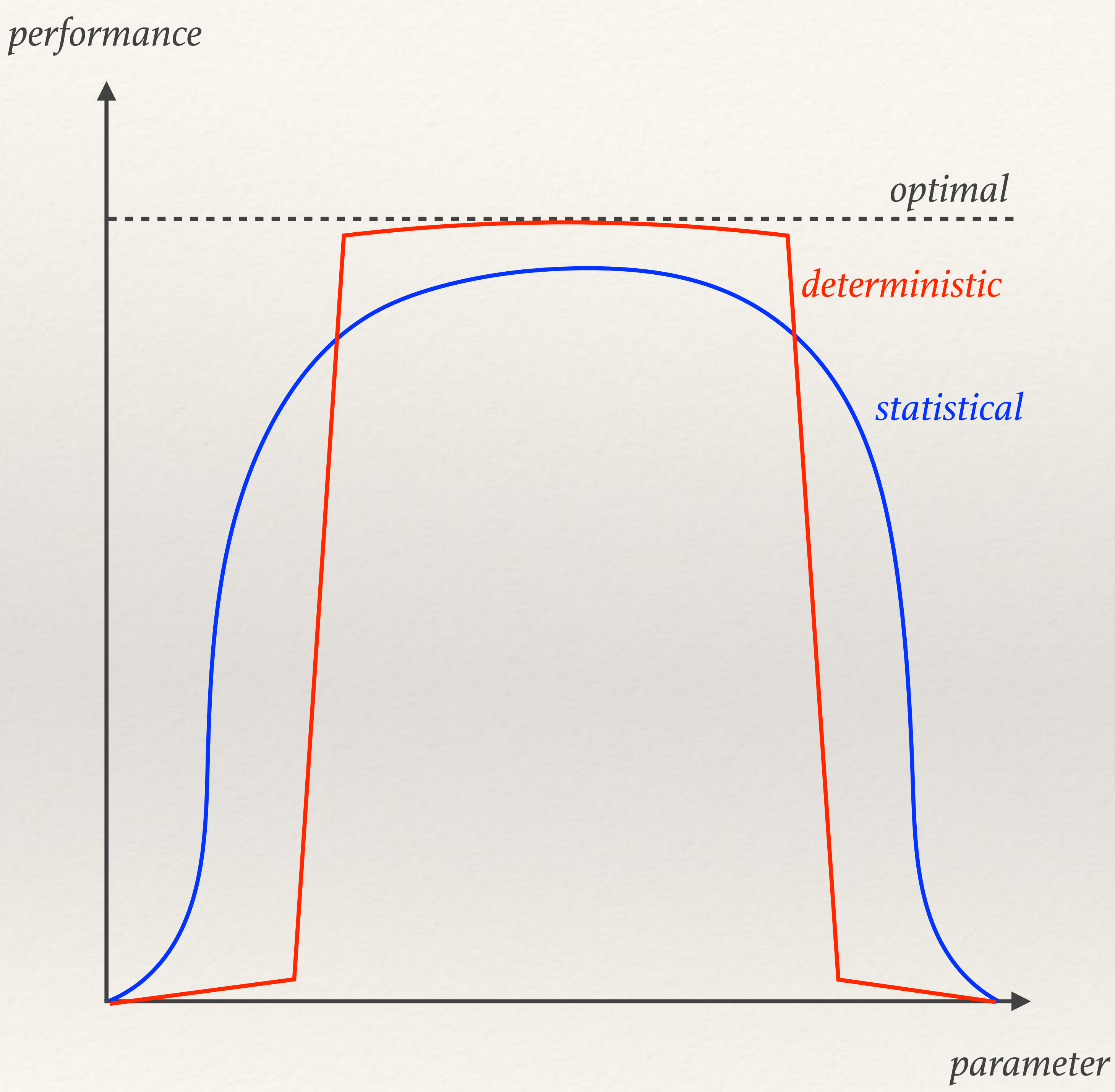
Beyond physics, the model may be too complex or just unknown.

A first-principles approach *does not apply* in these cases.

We need tools capable of model inference that can learn and utilize relevant information in the data

We can understand the model $\longrightarrow$ **We can understand the *data***

Statistical vs. Deterministic Modeling



deterministic models:

- high accuracy in a region of phase space
- bad failure modes outside domain of applicability

statistical models:

- reasonable accuracy across phase space
- graceful failure modes

which should we prefer?

Minimax Optimization

optimize for the best performance of the worst case

In most applications, we care about performance in the worst case:

- you want your bike/car/train/plane not to crash
- you want to avoid serious illness
- you buy insurance for costly rare events
- you prioritize products working over their features

→ **User experience is most sensitive to performance in the worst cases**

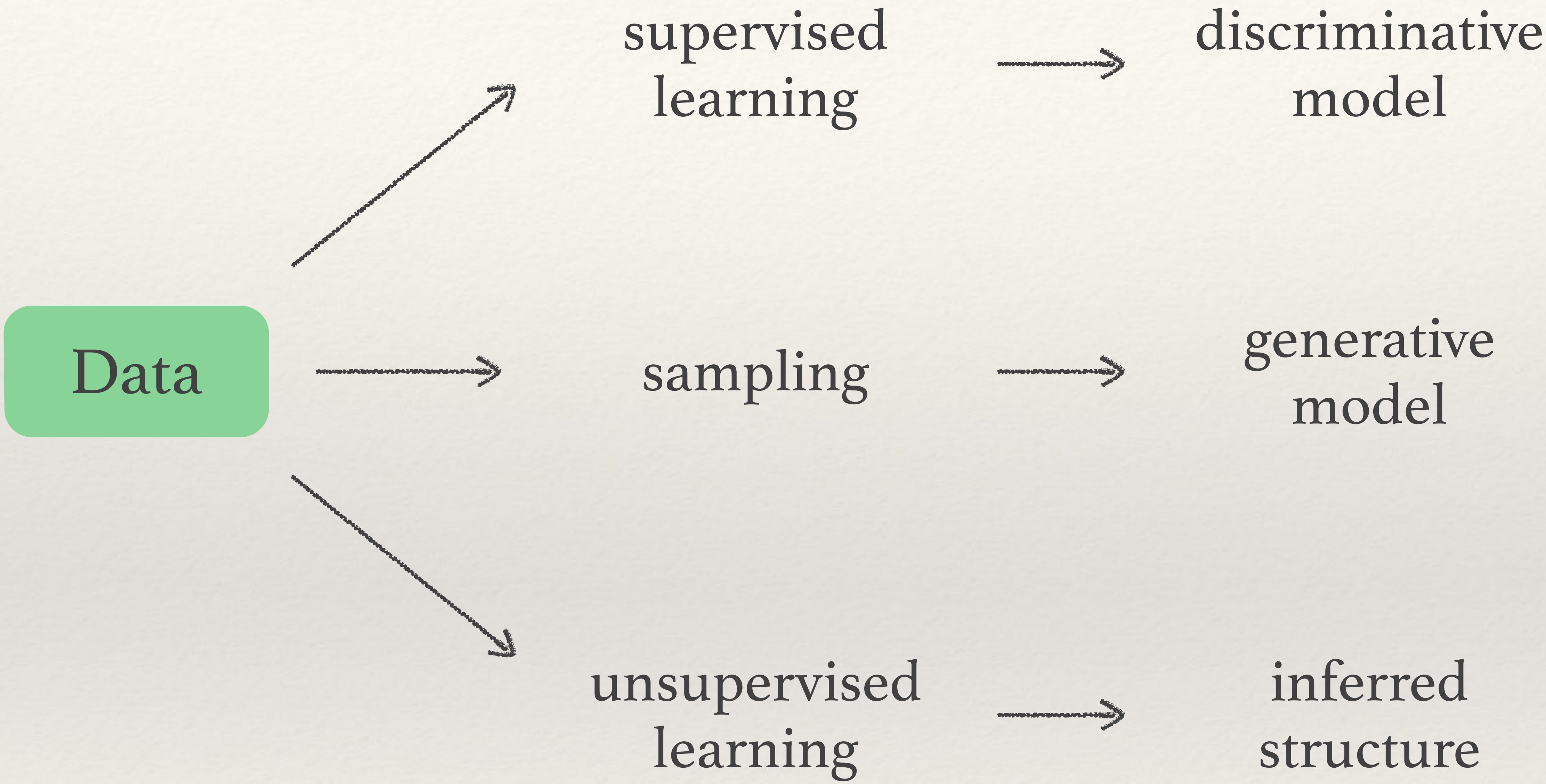
People tend to remember the worst parts of an experience
and base a valuation more heavily on that:

- the worst dishes at a meal
- reliability of a car
- annoyances in computer UX (e.g. Mac vs. Windows)
- *everything* about flying

A core tenet of machine learning

Learn expressive models

Understanding Datasets

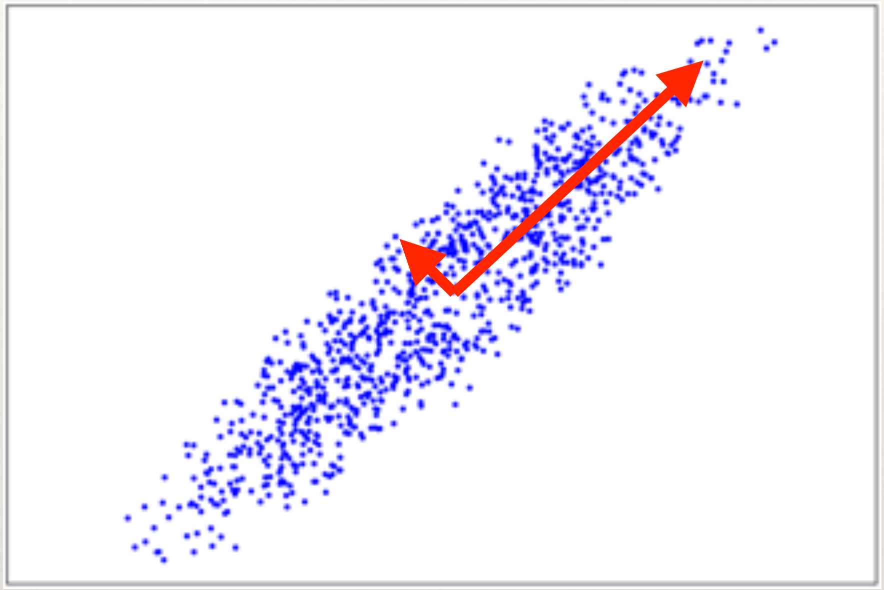


often, we want to transform the data into features as a first step

Feature Extraction

feature extraction: start with a linear model

PCA: rotate to a basis which maximizes the variance along principal directions



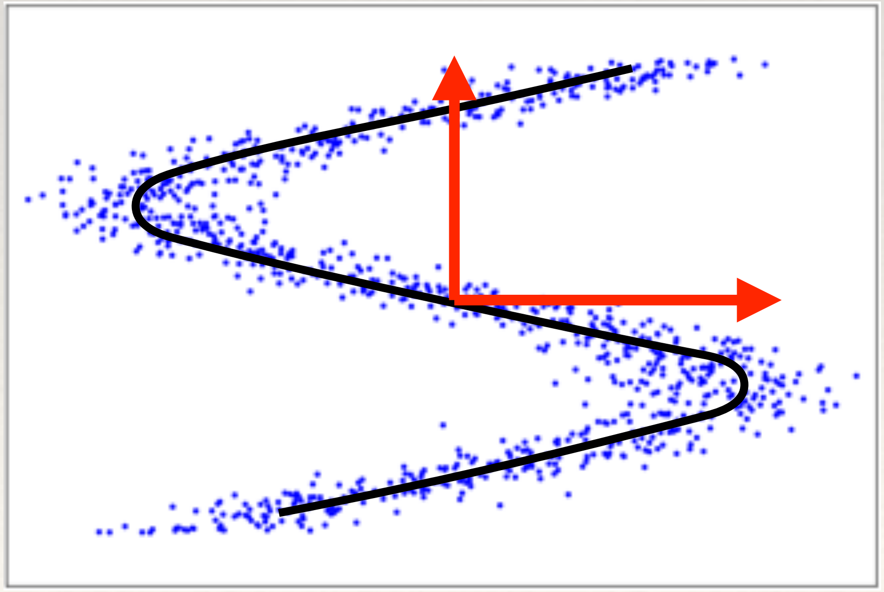
$$v = \mathbf{W}(x - \mu)$$

orthogonal transform

sample

feature means

can be ineffective for nonlinear manifolds



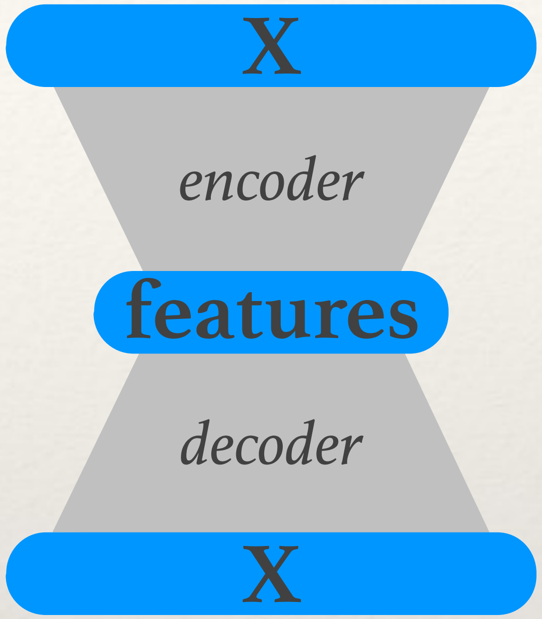
manifold learning tools:
manifold learning (e.g. isomap, LLE), autoencoders

Neural Networks and Autoencoders

autoencoder

$$\mathcal{M} : X \to X$$

a model that can reconstruct its inputs
with a constraint on intermediate features
(the encoded representation)



encoder and decoder: (stacks of) neural network layers

fully connected layer: $x \to f(\mathbf{W}x + b)$

input feature vector

activation function

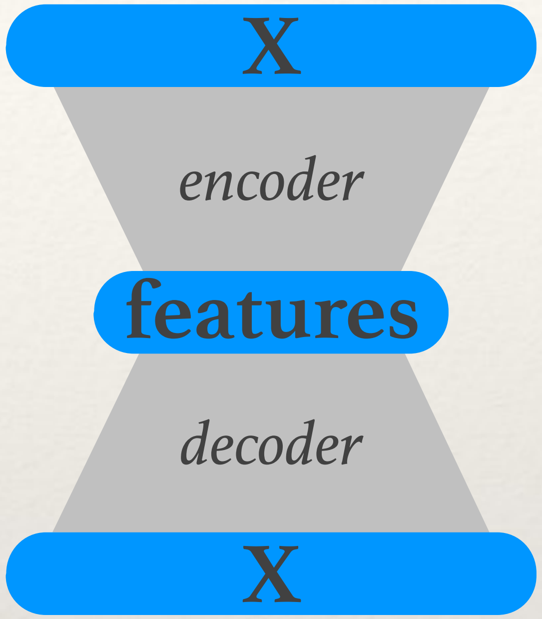
weight matrix

bias

Autoencoders

simplest autoencoder

encoder and decoder: single layers, tied weights



encoder: $y = f(\mathbf{W}x + b)$

full network:

$$x \to \mathbf{W}^{\mathrm{T}} y - b = \mathbf{W}^{\mathrm{T}} f(\mathbf{W}x + b) - b$$

network is trained to minimize reconstruction error

with a linear activation, the optimal solution is PCA
(where the bias removes the sample mean)

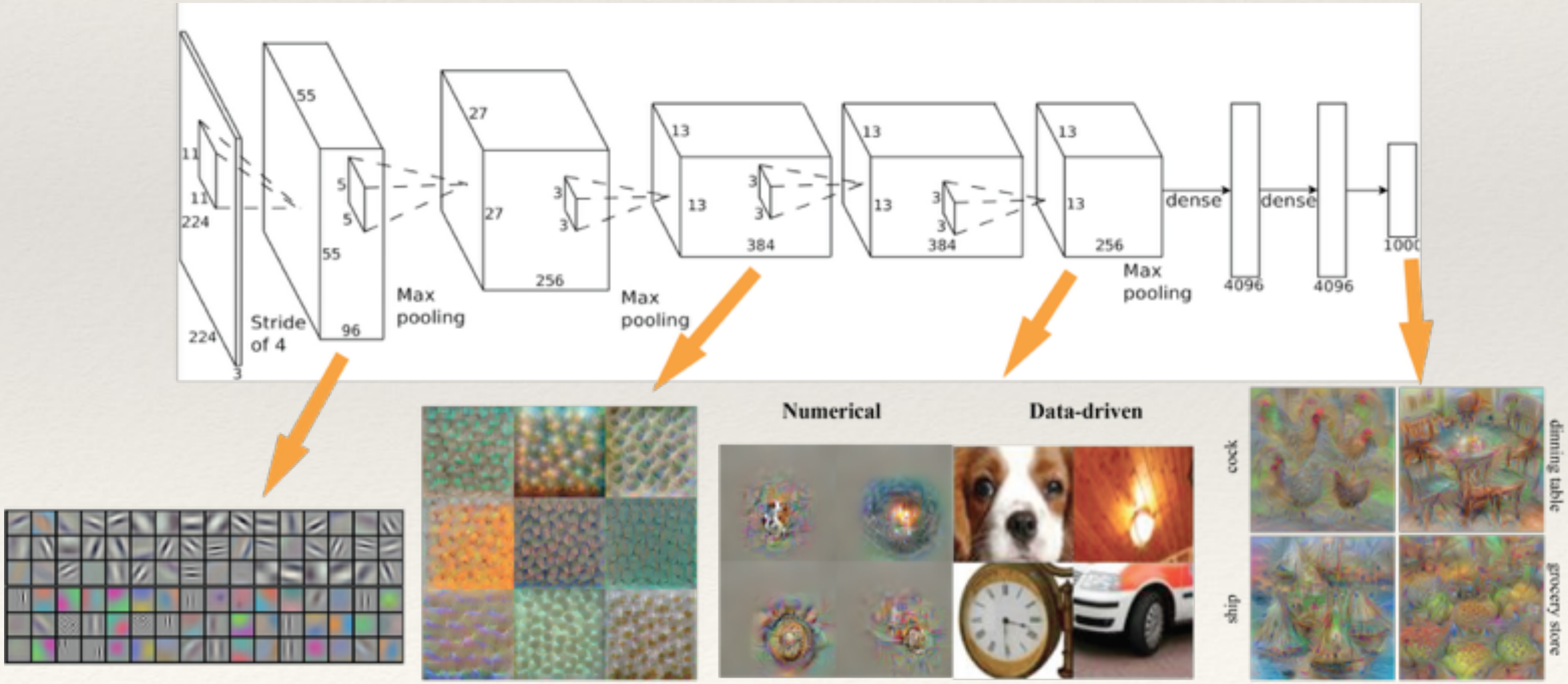
autoencoders are powerful tools for nonlinear manifold learning

Building Expressive Models

deep learning

models constructed from many simple transformation layers

challenges: effective learning algorithms and architectures, intelligent uses of data



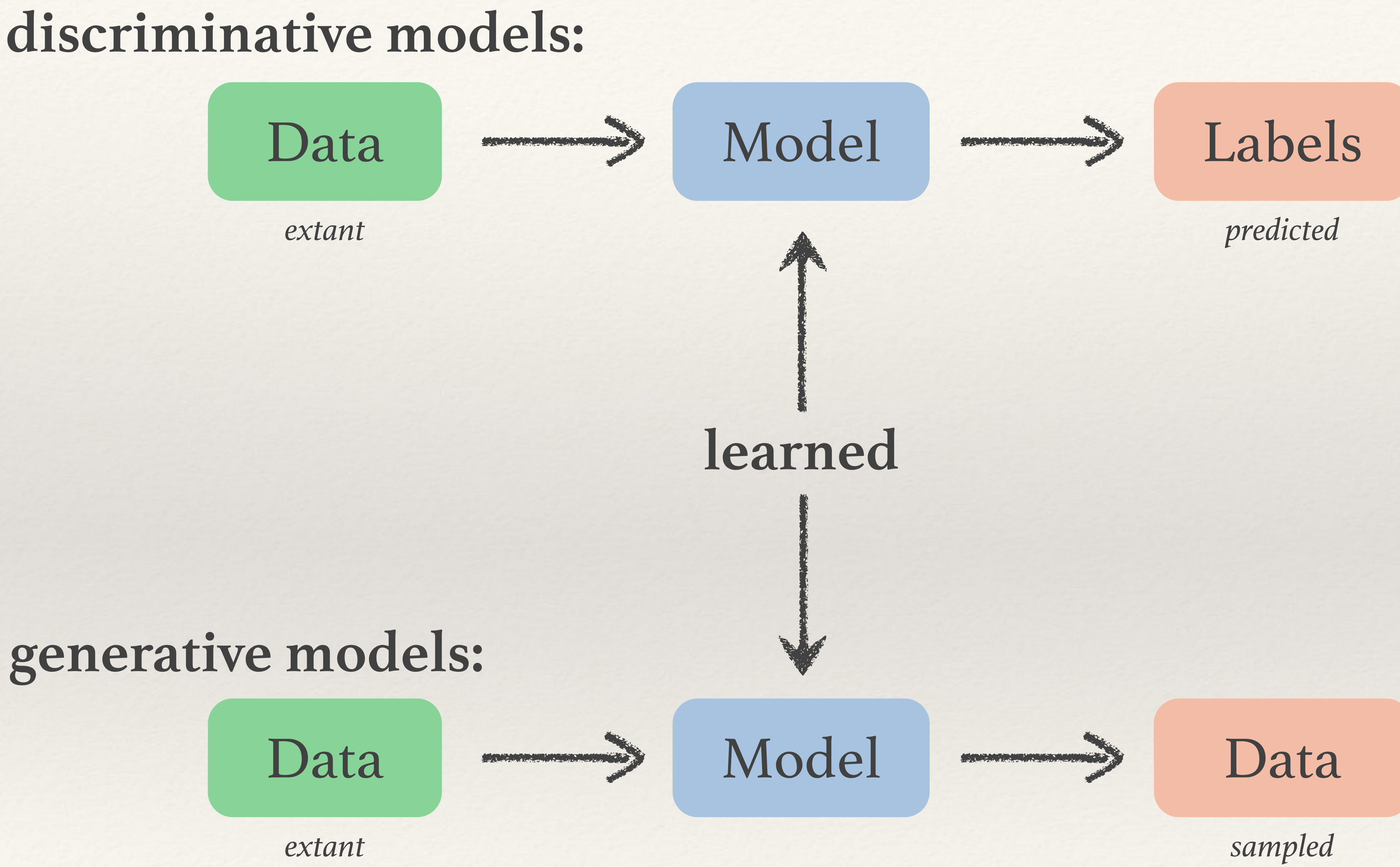
Conv 1: Edge+Blob

Conv 3: Texture

Conv 5: Object Parts

Fc8: Object Classes

How do we do machine learning?



How do we do machine learning?

discriminative models:

Data → Model → Labels

extant *predicted*

$$p(y \,|\, x)$$

conditional probability of labels

$$\mathcal{M} : x \to y$$

joint probability distribution

$$p(x, y)$$

generative models:

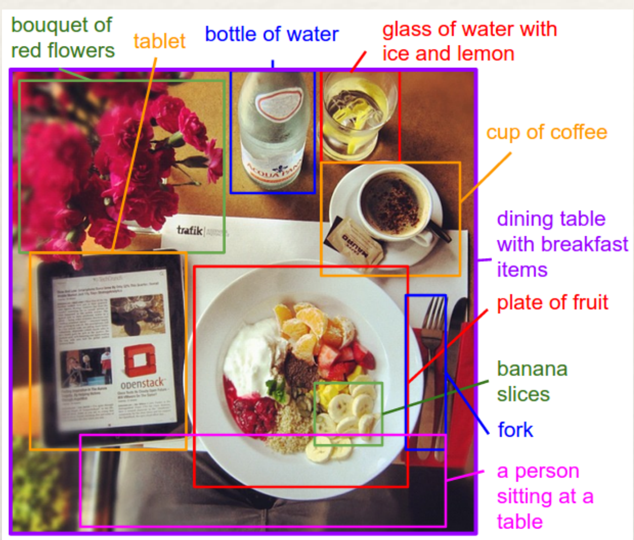
Data → Model → Data

extant *sampled*

Discriminative Models

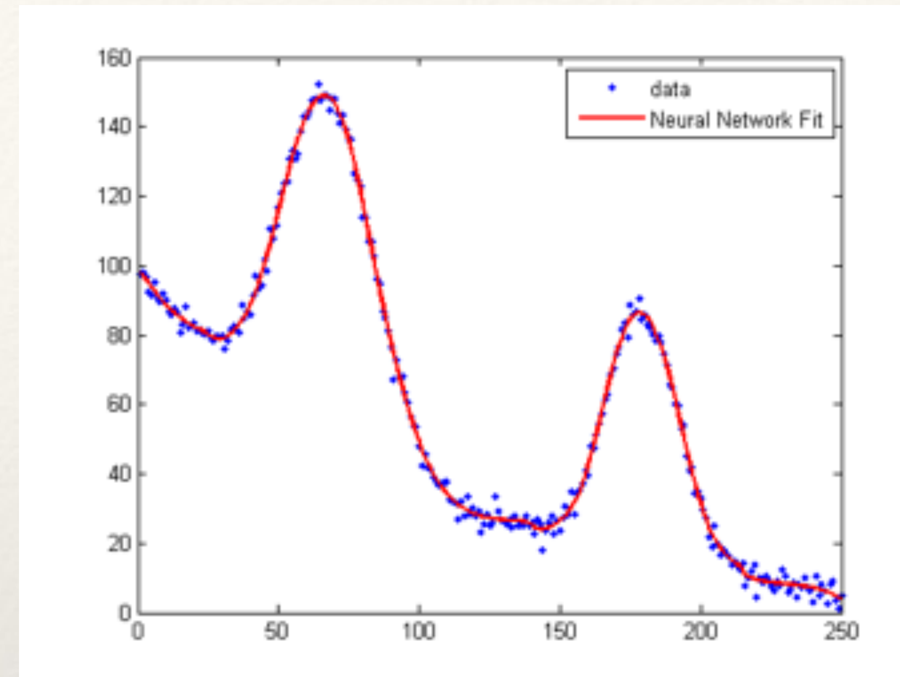
categorical discrimination

→ "cat"

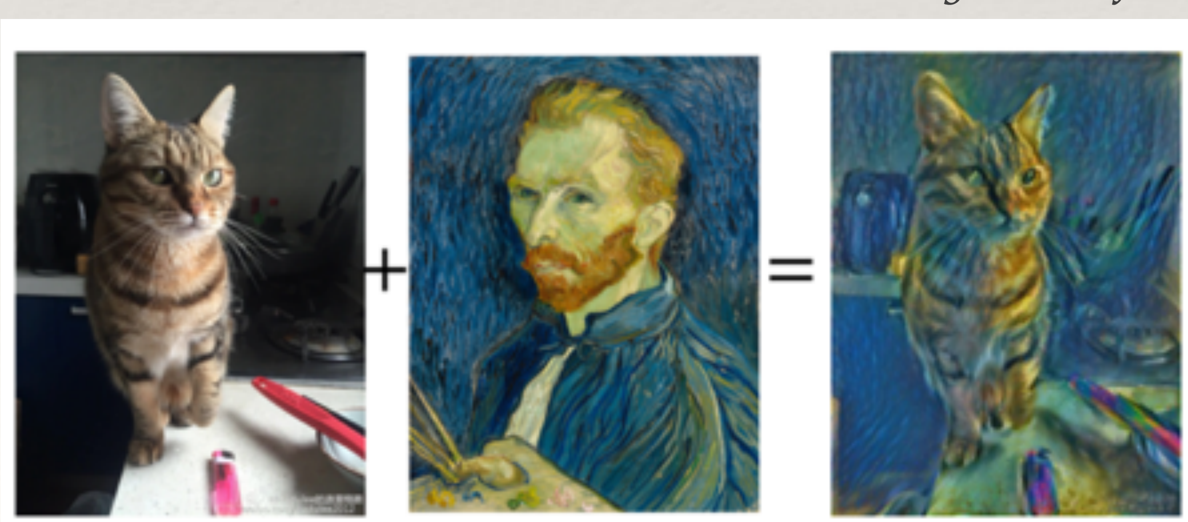


semantic labeling

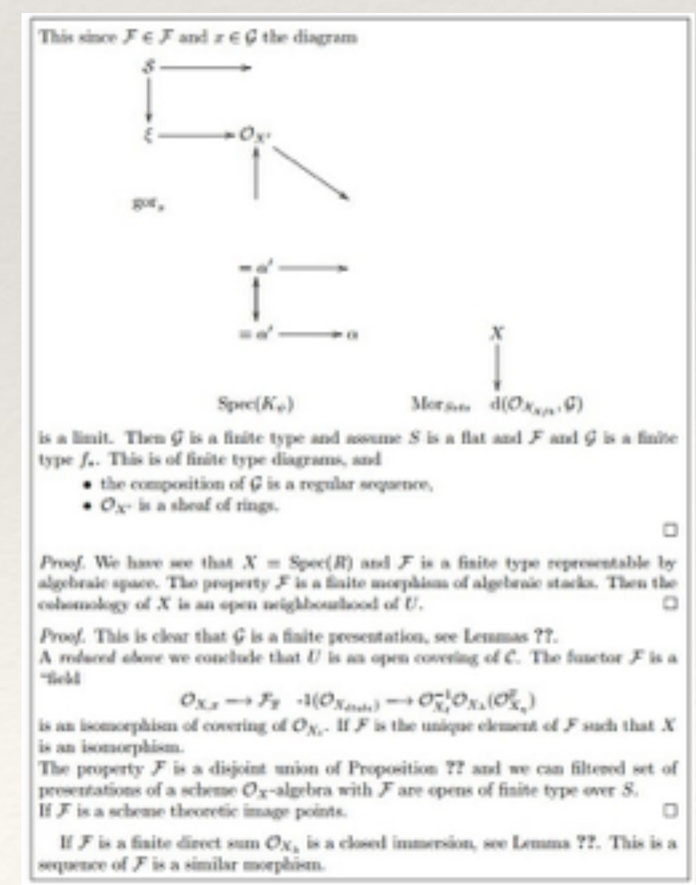
regression



style transfer



AIs will eventually replace us all



Generative Models

learn a probability distribution: how to sample from data

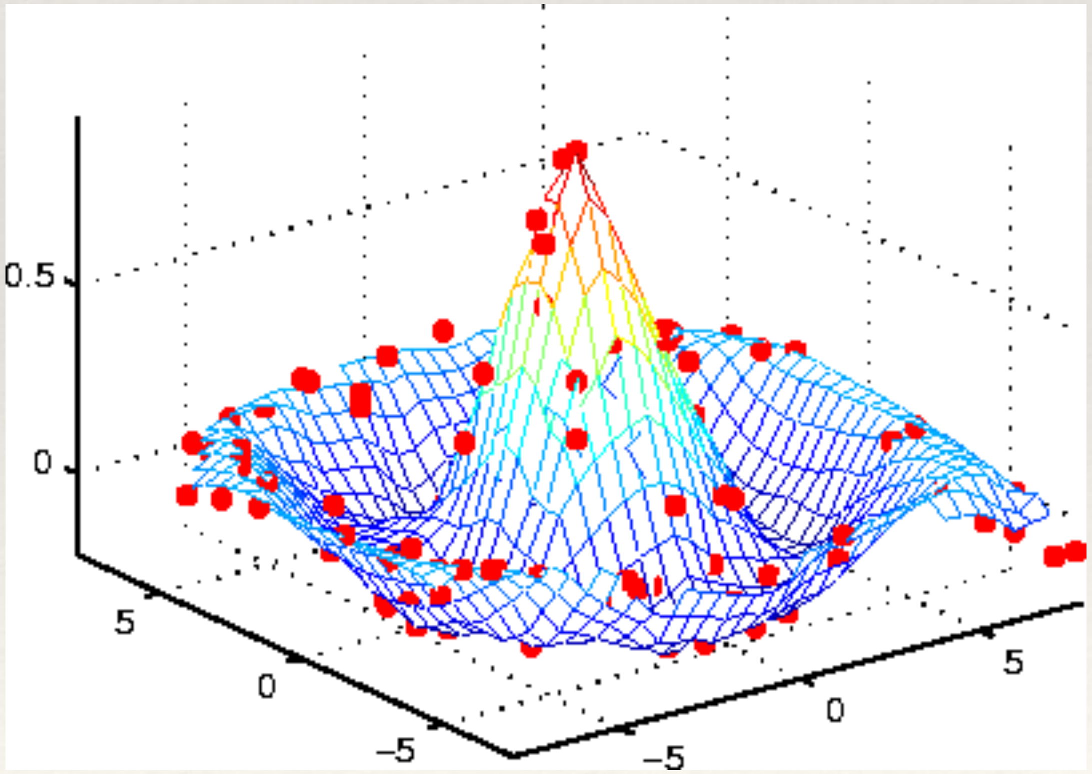
$$\widehat{J}(\Phi)\mathrm{d}\Phi \longrightarrow J(\Phi)\mathrm{d}\Phi$$

Jacobian
(observed via sampling)

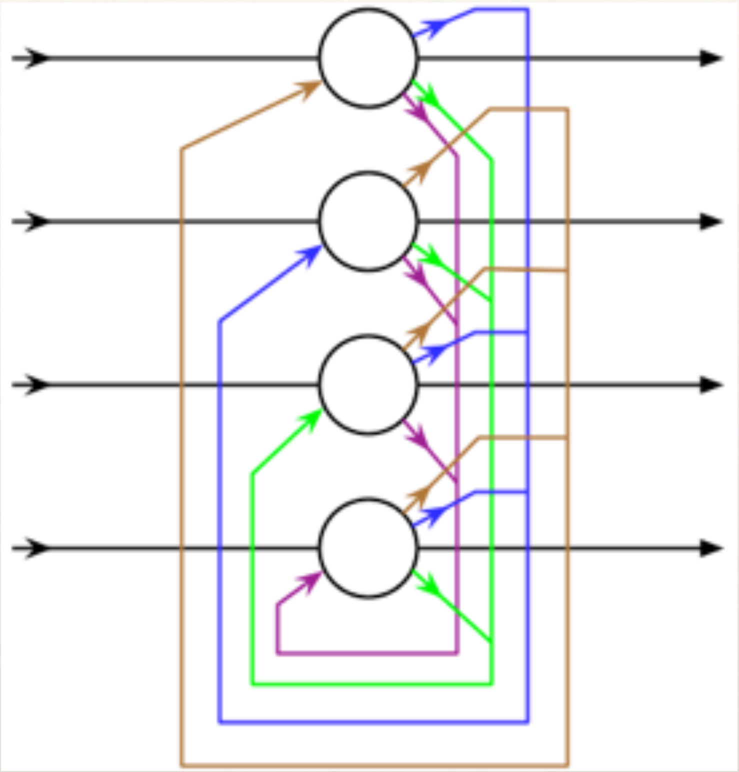
learned Jacobian

given points, learn the underlying distribution

physicists do this all the time



Hopfield Networks



early recurrent neural network (1982)

pairwise connections between nodes

$\{s_i\}$ binary states (-1, +1)

$\{W_{ij}\}$ connection strengths (couplings)

update rule: $$s_i = \begin{cases} +1 \text{ if } \sum_j W_{ij} s_j \geq \theta_i \\ -1 \text{ otherwise} \end{cases}$$

threshold

energy function: $$E = -\frac{1}{2}\sum_{i,j} W_{ij} s_i s_j + \sum_i \theta_i s_i$$

under repeated updates, the network converges to a local minimum in the energy function

The Ising Model

the Hopfield network energy function is similar to an Ising model:

$$E = -\sum_{<i\,j>} J_{ij} s_i s_j - \mu \sum_i h_i s_i$$

Hopfield: all sites 'adjacent'

note that the probability of a given state is dependent on the partition function:

$$P_\beta(s) = \frac{e^{-\beta E(\sigma)}}{Z_\beta}$$

$$Z_\beta = \sum_\sigma e^{-\beta E(\sigma)}$$

why Ising models?

simple models that embody the Hebbian learning rule:
neurons that fire together, wire together

concept can be used to store "memories" in the network:
attractor states that are local minima in the energy function

Stochastic Networks

while Hopfield networks are deterministic,
Ising models are probabilistic - like generative models

$$\Delta E_i = E(i\ \text{on}) - E(i\ \text{off})$$ energy difference between on/off states

state is activated with probability given by the Boltzmann distribution:

$$\beta \Delta E_i = \ln p_{i\,\text{on}} - \ln p_{i\,\text{off}}$$

$$p_{i\,\text{on}} = \frac{1}{1 + \exp(-\beta \Delta E_i)} = \sigma(-\beta \Delta E_i)$$

this type of network is a **Boltzmann machine**

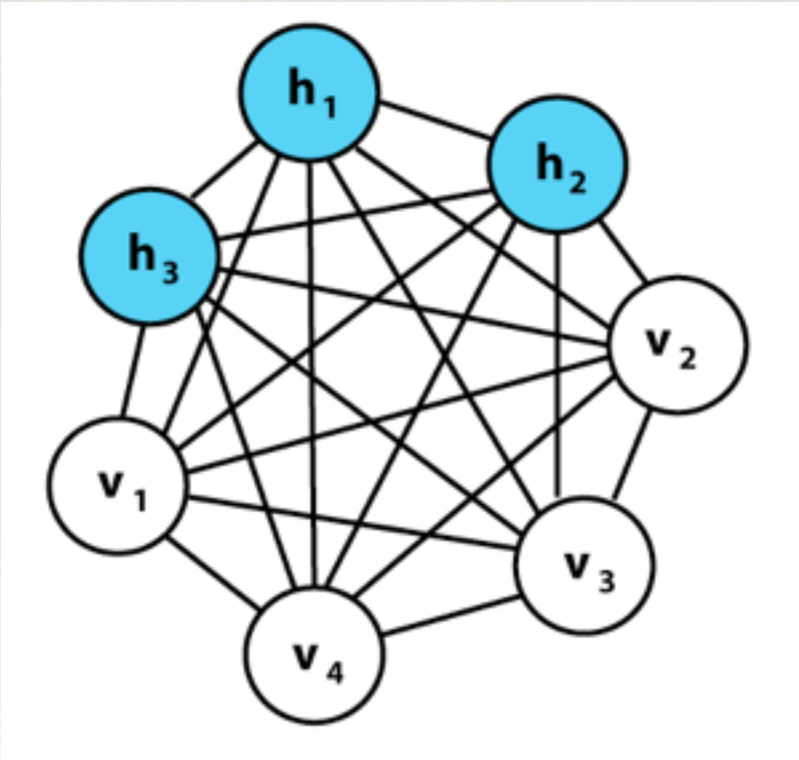
Boltzmann Machines

$$p_{i\,\mathrm{on}} = \frac{1}{1+\exp(-\beta\Delta E_i)} = \sigma(-\beta\Delta E_i)$$

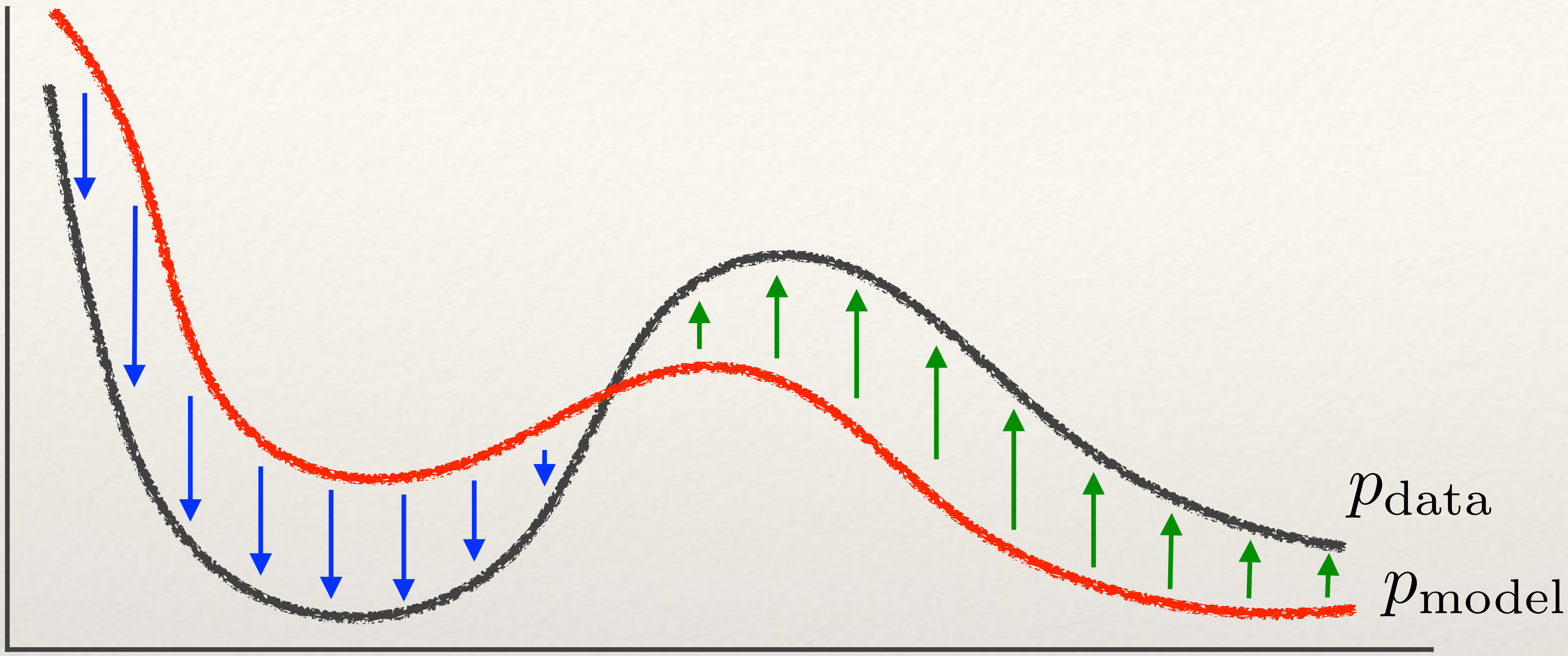
update step: sample sites and set their states according to the Boltzmann distribution; repeat until thermal equilibrium obtained

Structurally, we build a Boltzmann machine from visible (external) and hidden (internal) units

Q: how do we set the weights? (learning/training)



Training Generative Models



Training: the joint distribution of the model should be adjusted towards the true data distribution

$$\frac{\partial \ln p}{\partial W_{ij}} = \left(\langle p_{ij} \rangle_{\text{data}} - \langle p_{ij} \rangle_{\text{model}} \right)$$

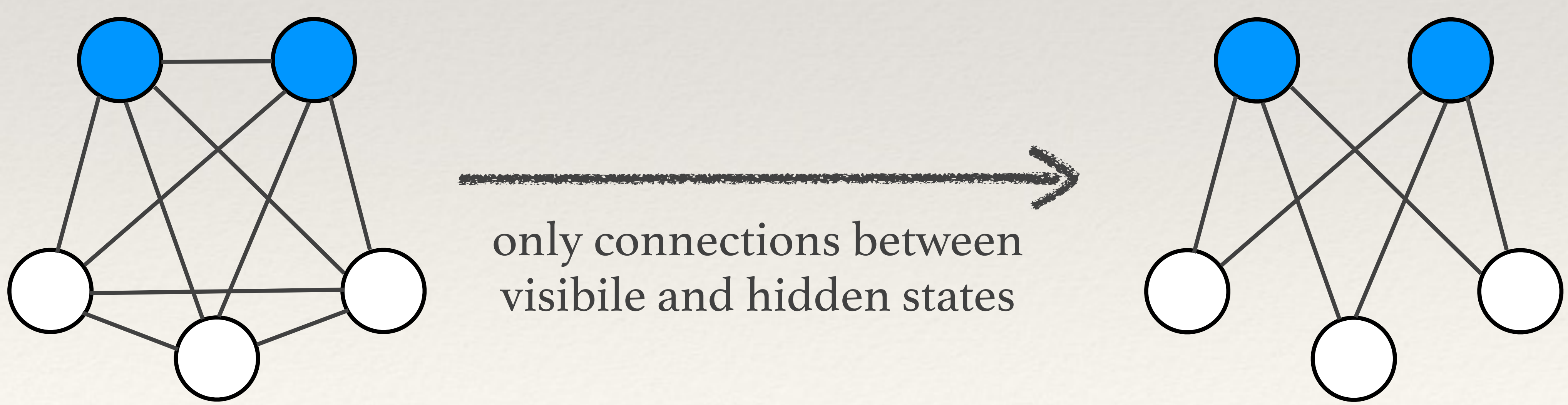
probability of *i* and *j* being on

Restricted Boltzmann Machines

training Boltzmann machines is challenging:

- because all states in the network are connected, sampling is extremely time-intensive (single weight updates must propagate through the entire network)
- current training algorithms becomes ineffective beyond small networks

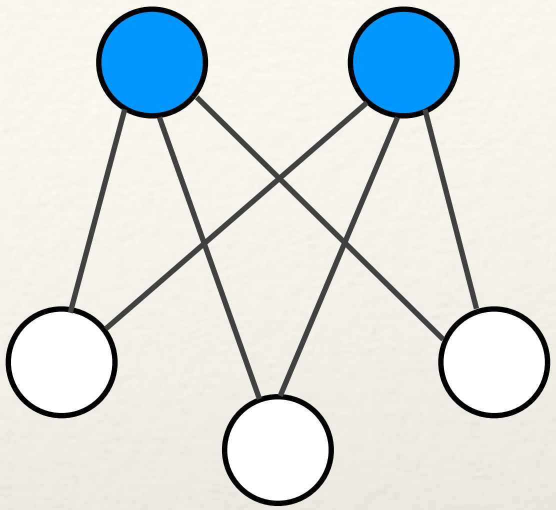
one solution: **restricted Boltzmann machines**



Restricted Boltzmann Machines

RBM energy function:

$$E = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_{i,j} W_{ij} v_i h_j$$



gradient:

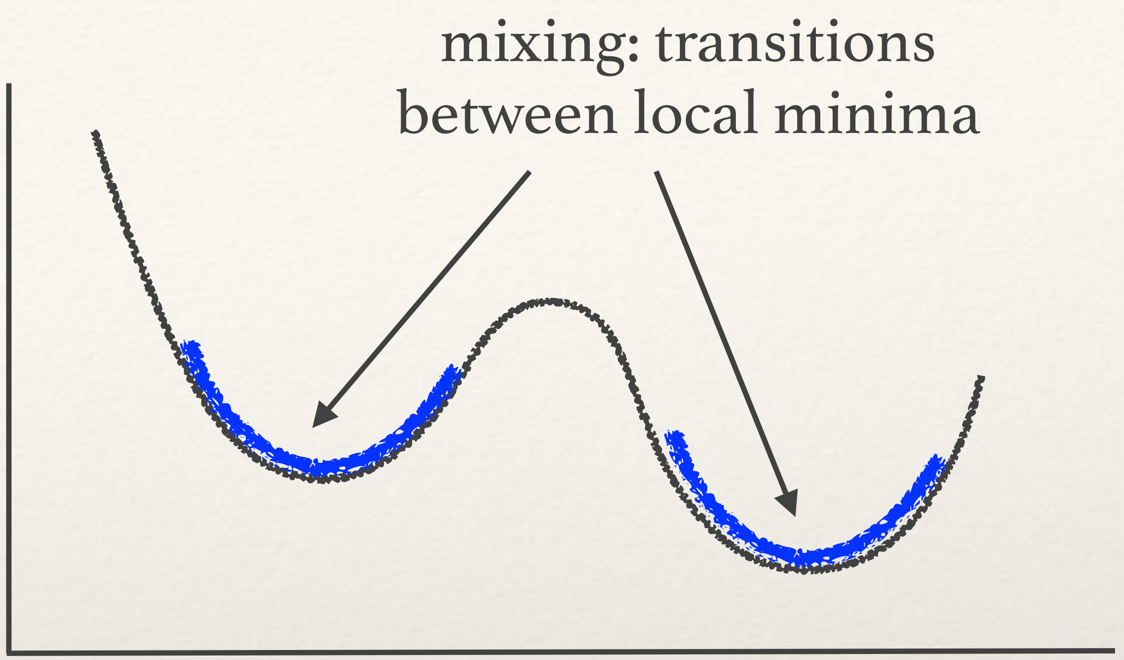
$$\frac{\partial \ln p}{\partial W_{ij}} = \left(\langle v_i h_j \rangle_{\mathrm{data}} - \langle v_i h_j \rangle_{\mathrm{model}}\right)$$

because the visible states are independent and the hidden states are independent, each group can be collectively sampled
- Gibbs sampling

RBMs can be effectively trained and used to build expressive models (deep belief networks)

Challenges in Generative Models

- sampling is challenging: e.g. obtaining *mixing*
- no reliable software packages for training and using RBMs and general Markov process models
- demonstrating a diversity of applications (compete with NNs)



my interests:

- improving sampling methods for generative models using physical systems
- developing useful software tools for experimenting with generative models
- exploring the connection between statistical physics models and ML - are there good architectures arising from other types of models?

Machine learning is making an enormous impact

The New York Times

Artificial Intelligence Swarms Silicon Valley on Wings and Wheels

Machine learning is making an enormous impact

The New York Times

Artificial Intelligence Swarms Silicon Valley on Wings and Wheels

And physicists will play a major role in it

WIRED

MOVE OVER, CODERS—PHYSICISTS WILL SOON RULE SILICON VALLEY

Machine learning is making an enormous impact

The New York Times

Artificial Intelligence Swarms Silicon Valley on Wings and Wheels

And physicists will play a major role in it

WIRED

MOVE OVER, CODERS—PHYSICISTS WILL SOON RULE SILICON VALLEY

We even share the same problems

The New York Times

Artificial Intelligence's White Guy Problem

Summary

Machine Learning is just great:

- diverse applications
- exploding interest
- amazing opportunities
- deep roots in statistical physics
- many open questions

Physicists have the tools to make fundamental contributions to machine learning, both inside and beyond physics