Introduction to Embedded Systems Research: Machine Learning in the Internet-of-Things

Robert Dick

dickrp@umich.edu
Department of Electrical Engineering and Computer Science
University of Michigan

Outline

Deadlines and announcements

20 Oct.: R. P. Dick, L. Shang, M. Wolf, and S.-W. Yang, "Embedded Intelligence in the Internet-of-Things," *IEEE Design & Test of Computers*, Dec. 2019.

8 Nov.: B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: Perceptron, Madaline, and backpropagation," *Proc. IEEE*, vol. 78, no. 9, Sept. 1990.

8 Nov.: Final project checkpoint.

Final project checkpoint

By 8 Nov., you should have the main components of your project working together.

Be at the point of fine tuning, evaluation, and writing, which takes a month.

Revise your proposal again, explaining your progress, future plans, and schedule.

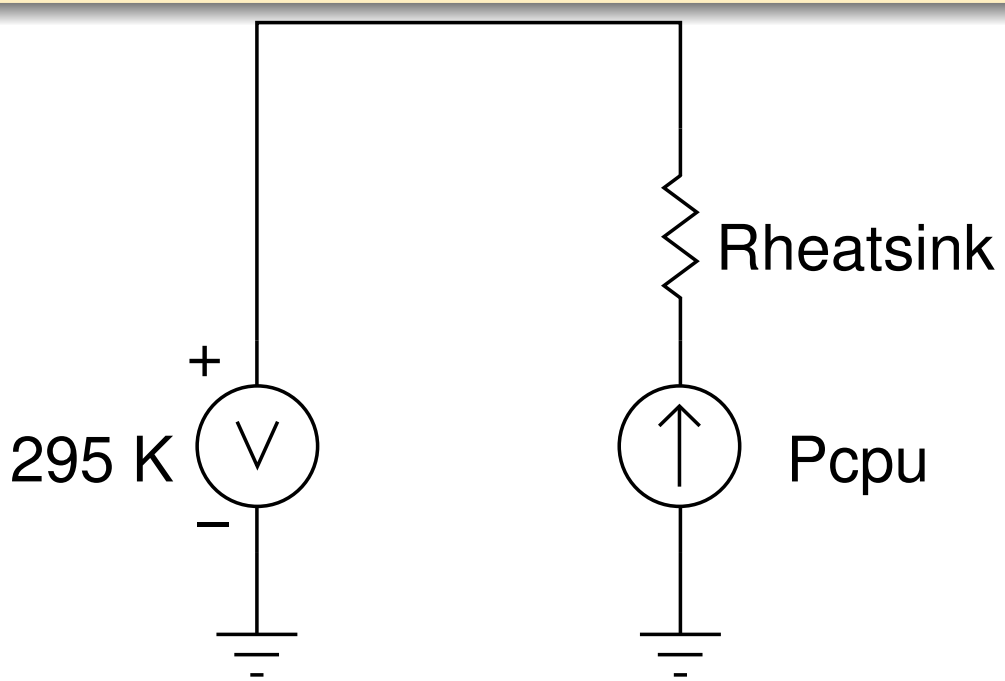
If you deviated from your schedule in Checkpoint 1, indicate how you plan to change your approach to enable completion of a working project.

9 Dec.: final project deadline.

Midterm

Please submit regrades by Thursday before class. I plan to resolve them all Thursday evening.

Only one question had a mean well below the exam mean: the ambient temperature question.



Outline

Applications

- Smart city: $350B (Markets and Markets).
- Smart homes: $31B (Statista).
- Wearables: $30B (Markets and Markets).
- Connected vehicles: $60B (Sheer Analytics and Insights).
- Networked manufacturing.
- Networked agriculture.
- Networked medical care.
- Smart retail and supply chain.
- Environmental management.

Wireless communication standards

Technology	Power (mW)	Range (m)	Typical rate (kb/s)
4G	1,000	70,000	10,000
5G	1,000	40,000	100,000
WiFi / 802.11(g)	250	140	20,000
Zigbee / 802.15.4	1–100	10–1,500	20–200
LoRaWAN	10	15,000	20
NB-IoT	100	15,000	250

Energy efficiency I

ARM Cortex A57

- Mid-range IoT application processor.
- 7.1 W at 1.9 GHz.
- 64-bit processor.
- 4 MIPS/MHz $\rightarrow$ 7.6 GIPS.
- Average instruction duration: 130 ps.
- 930 pJ/word.
- 15 pJ/bit.

LoRaWAN

- 10 mW.
- 20 kb/s.
- 0.5 µJ/b.

Energy efficiency II

34,000 bit computations per bit transmission.

MICAz was 625×.

Similar analysis for 5G and Arm: 670×.

Reliability and security

All the problems we learned about for other embedded systems, plus. . .

Large attack surface: sensors, algorithms, networks, and actuators.

No single company knows entire system design: formal methods impossible.

Large-scale system composed of heterogeneous components.

Fault processes are interdependent due to indirect coupling (environmental and social.

Identifying catastrophic system failure modes akin predicting financial systems, not isolated embedded systems.

Manually characterizing indirect relationships among IoT component fault processes is impractical.

Computational platforms

- Low-power GPUs likely to become available and common.
- Semi-custom accelerators for limited-width parallel MAC operations.
- Analog weight state memories.

Outline

Section outline

3. Machine learning
   - Manually designed classification algorithms
   - Machine learning
   - Neural networks perspective 1: biomimetic computation
   - Neural networks perspective 2: function approximation

Classification

Determining which class a new observation falls into.

Easy example

Student: 0.72, Protester: 0.14, Cat: 0.08, ...

Easy example

Cat: 0.81, Student: 0.03, ...

Moderate complexity example

Cat: 0.61, Student: 0.34, Protester: 0.14, ...

Impossible example

Cat: 0.48, Student: 0.45, ...

Manual algorithm design: feature extraction

- Bounding box using eye detection.
- Hairiness feature using edge detection: **scalar**.
- Image segmentation based on color.
- Pose classification based on segment shapes: **vector**.
- Image region color histograms: **vector**.

Bounding box using eye detection

Bounding box using eye detection

Manual algorithm design: feature extraction

Bounding box using eye detection.

Hairiness feature using edge detection: **scalar**.

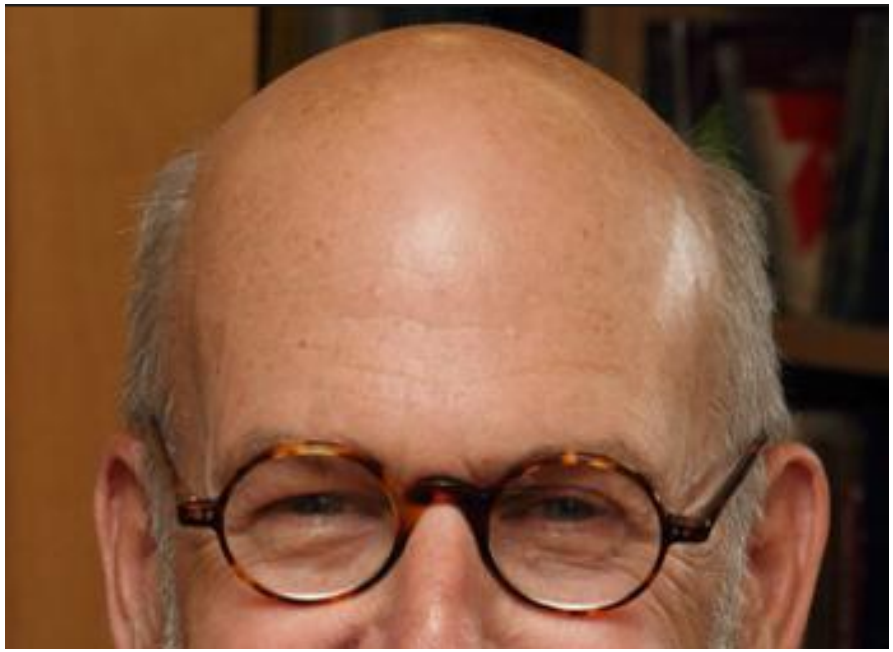
Image segmentation based on color.

Pose classification based on segment shapes: **vector**.

Image region color histograms: **vector**.

Hairiness feature using edge detection: **scalar**

Cat.



Not cat.

Hairiness feature (non-cat)

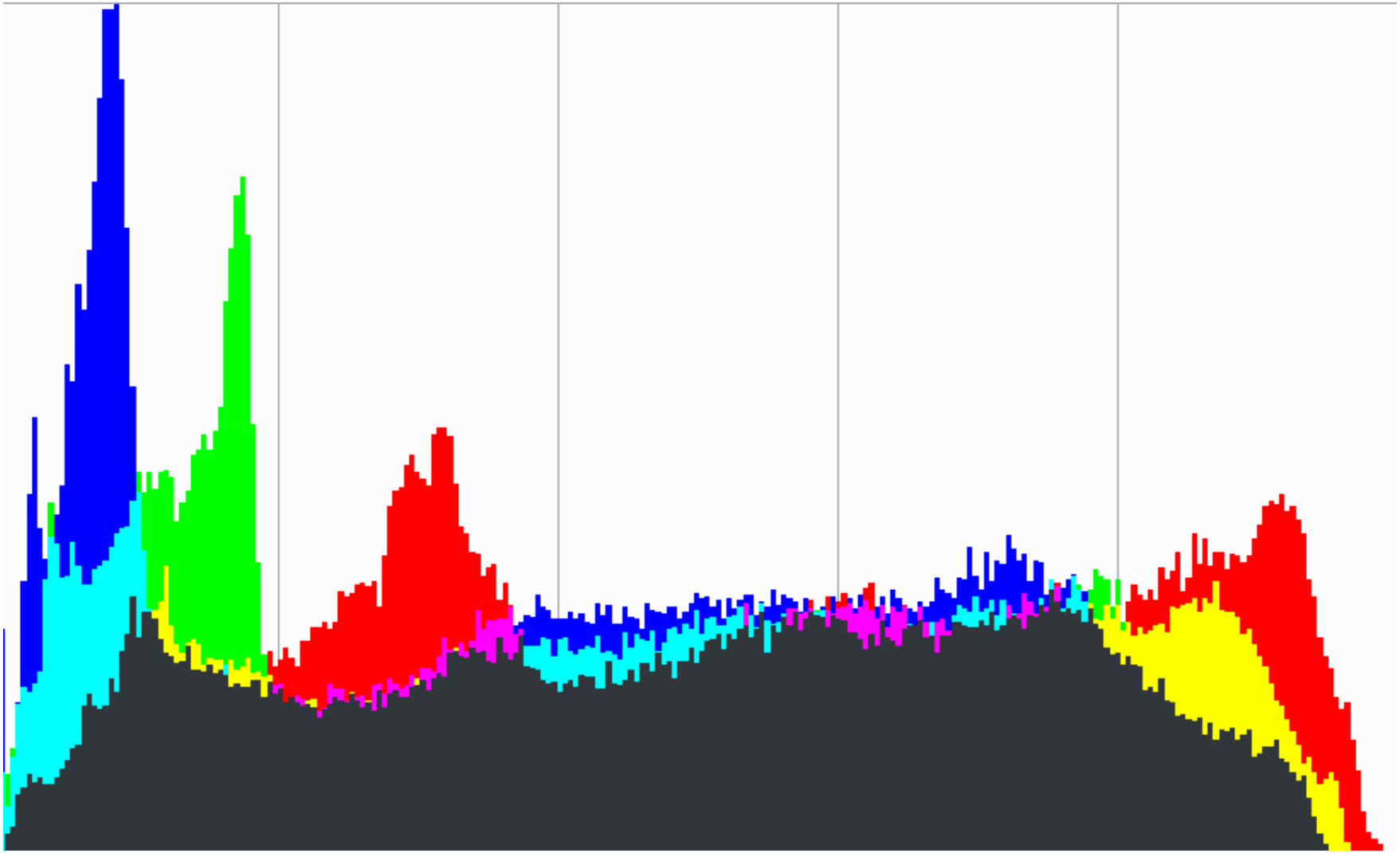
Cat.

Not cat.

Manual algorithm design: feature extraction

- Bounding box using eye detection.
- Hairiness feature using edge detection: **scalar**.
- Image segmentation based on color.
- Pose classification based on segment shapes: **vector**.
- Image region color histograms: **vector**.

Color histogram



Principal component analysis

Often inappropriate in production systems.

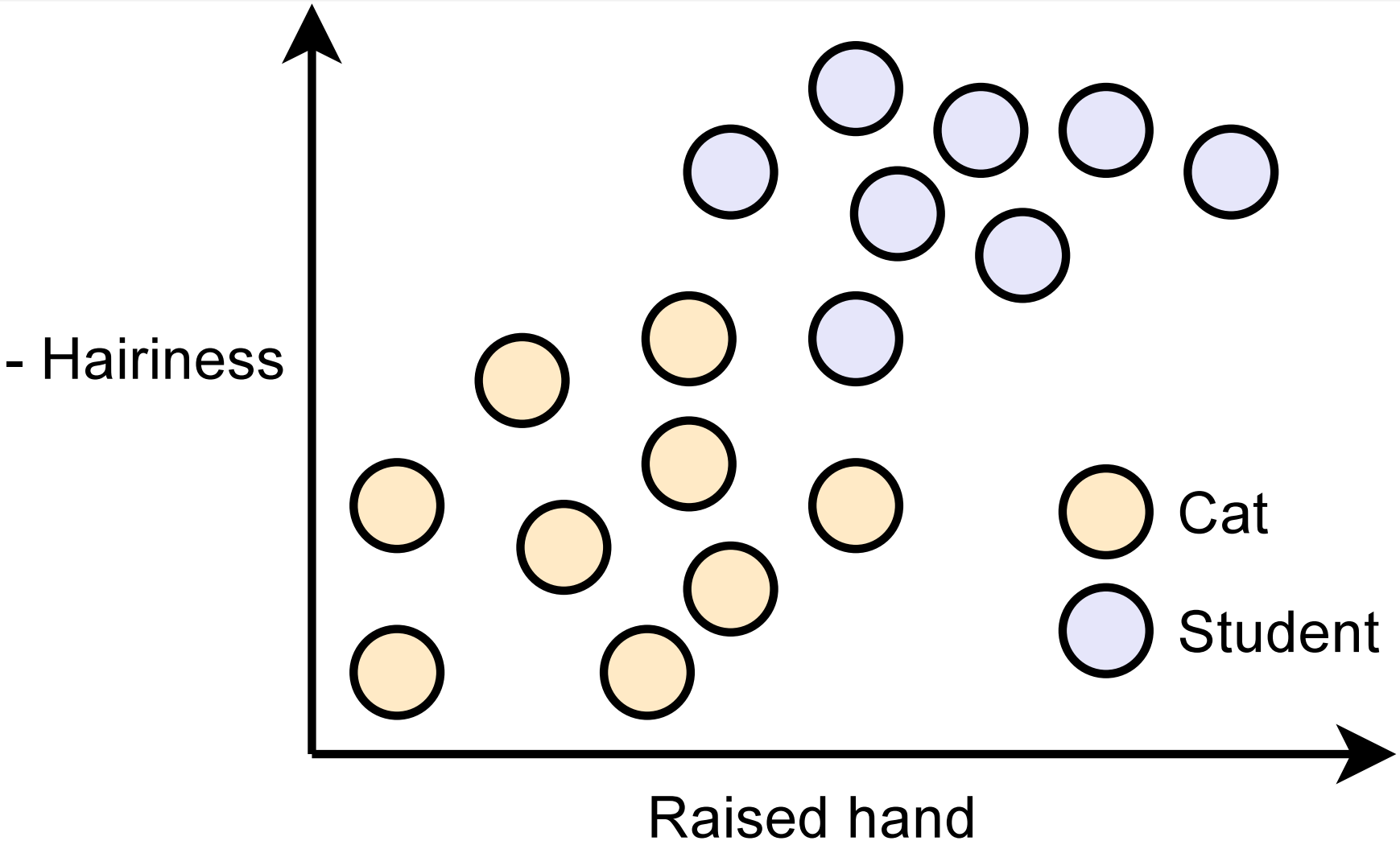
Valuable during learning process.

Transforms a data set from an input space to an output space in which dimensions are orthogonal and are ordered by decreasing variance.

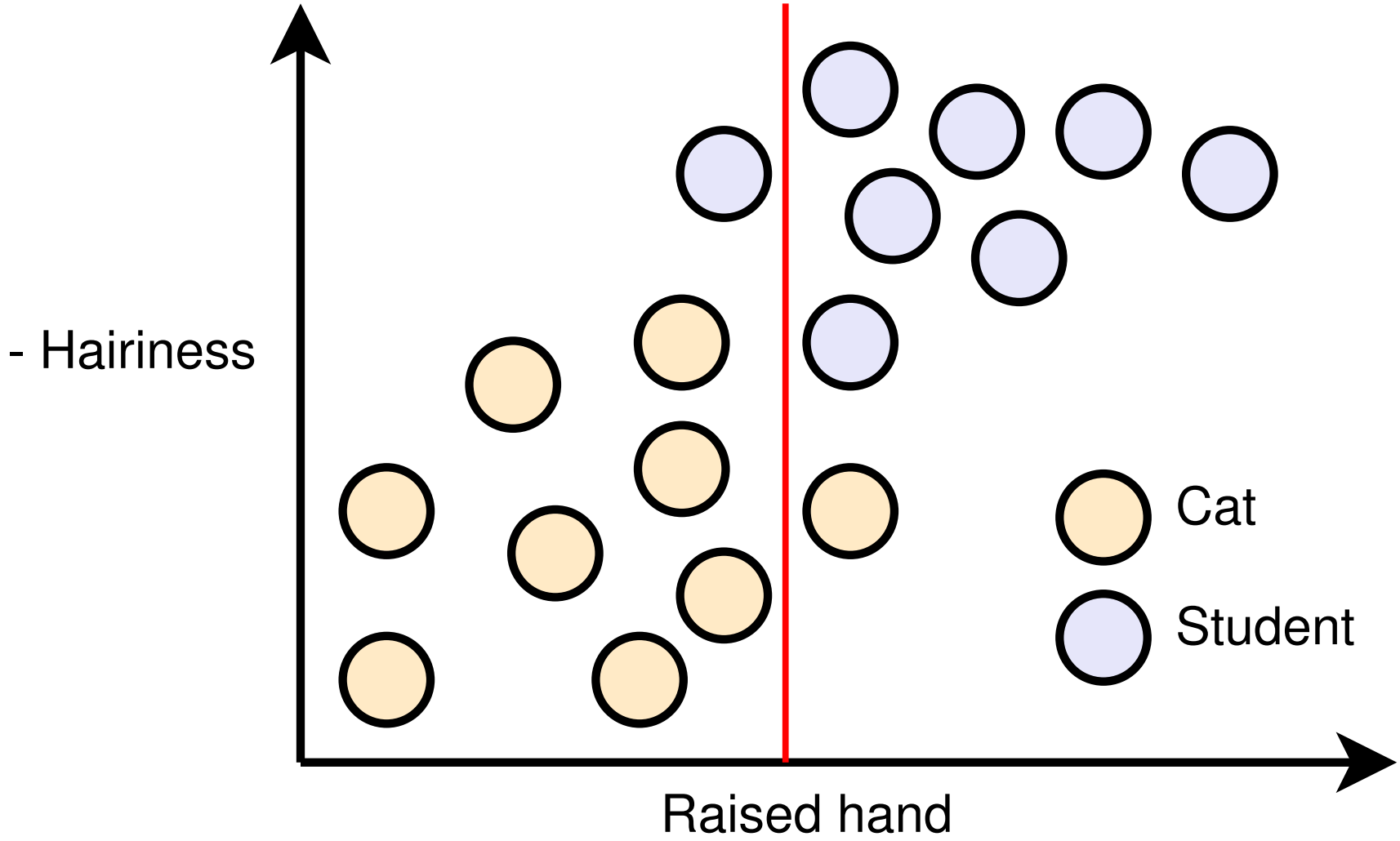
Too many dimensions to plot.

Truncation can be useful for data visualization.

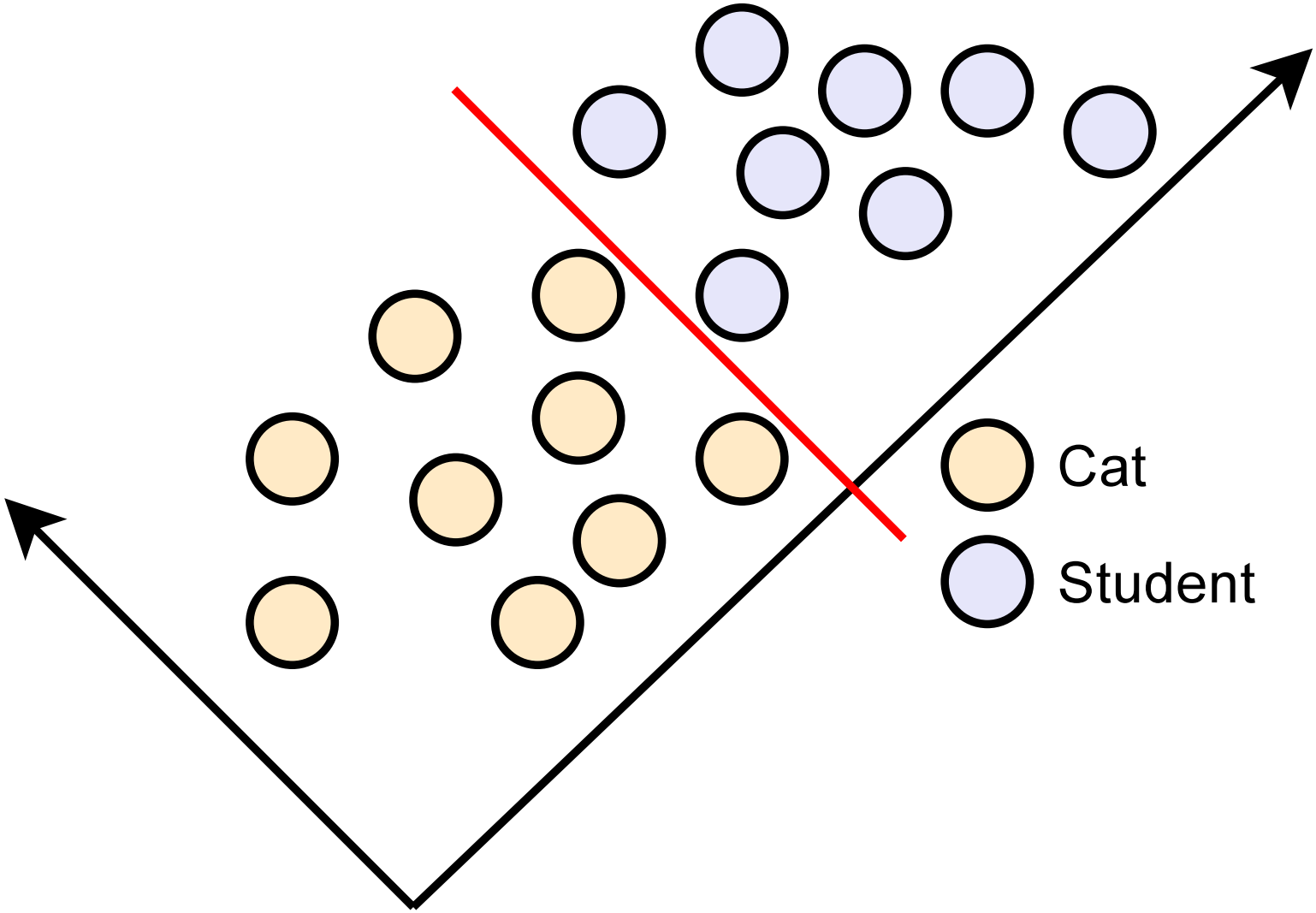
Two dimensions of original data (there are many more)



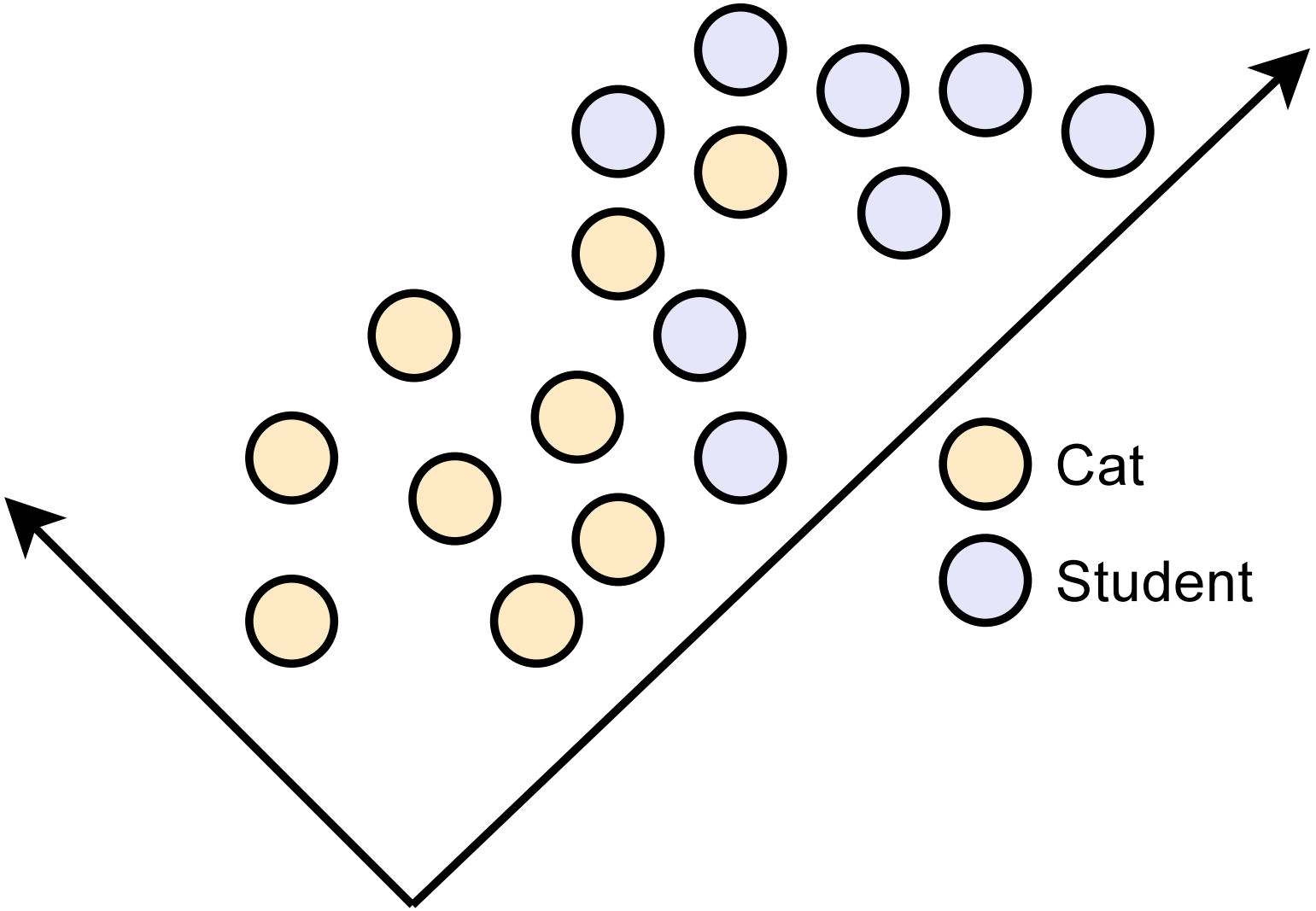
Can't use single feature for correct partitioning



First two principal components



Class overlap in feature space



Feature extraction

Old way of thinking

- Feature extraction is distinct from the rest of the machine learning process.
- Humans carefully determine ideal features.
- Statistical learning techniques determine how to partition space based on training examples.

New way of thinking

- Each stage of machine learning increases the level of abstraction.
- Going from pixels to edges.
- Going from edges to parallelograms.
- Automatically learn features from raw data, at a computational cost.

Section outline

3. Machine learning
   - Manually designed classification algorithms
   - Machine learning
   - Neural networks perspective 1: biomimetic computation
   - Neural networks perspective 2: function approximation

Manual algorithm design: feature extraction

Bounding box using eye detection.

Hairiness feature using edge detection: **scalar**.

Image segmentation based on color.

Pose classification based on segment shapes: **vector**.

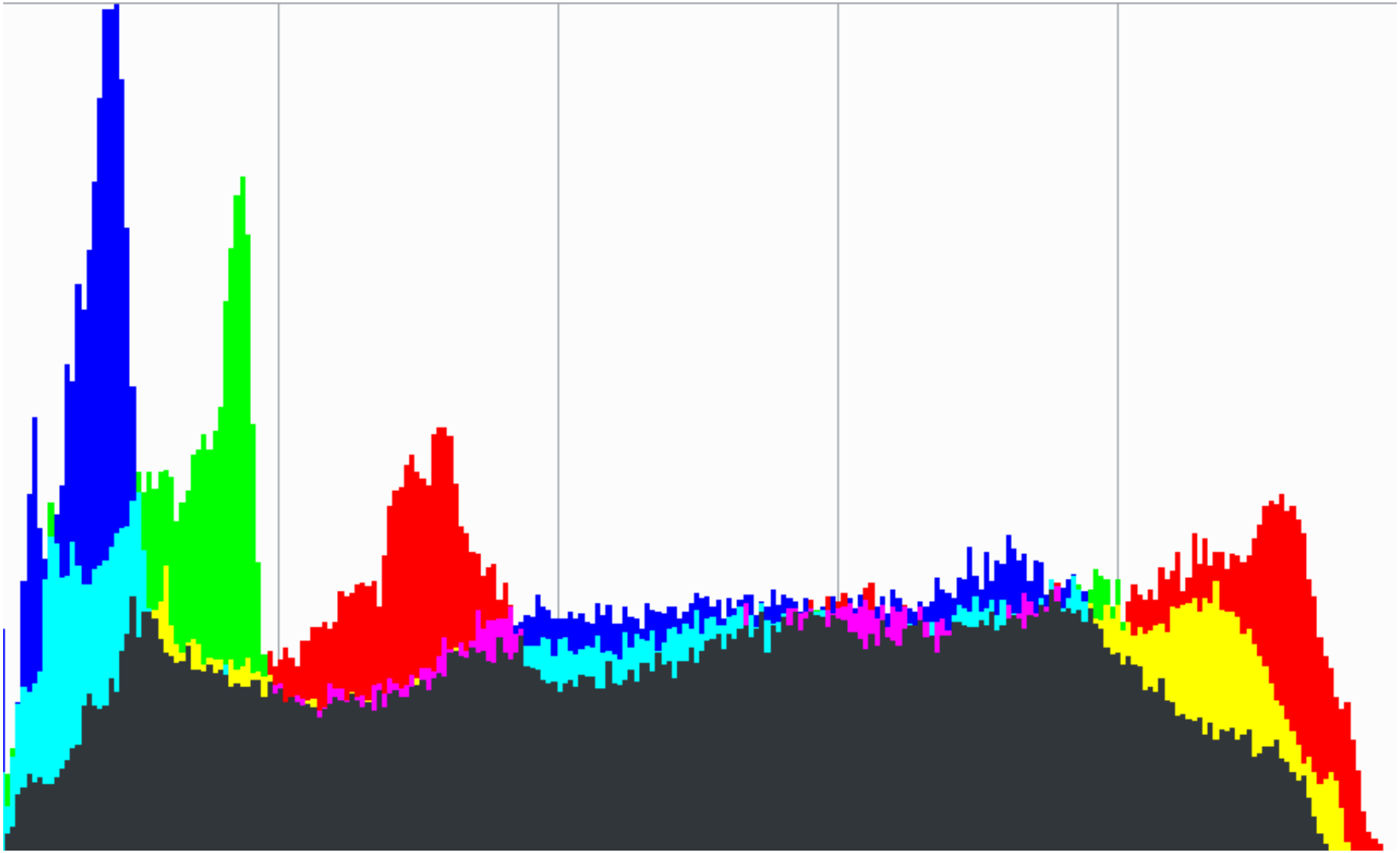
Image region color histograms: **vector**.

Hairiness feature

Manual algorithm design: feature extraction

- Bounding box using eye detection.
- Hairiness feature using edge detection: **scalar**.
- Image segmentation based on color.
- Pose classification based on segment shapes: **vector**.
- Image region color histograms: **vector**.

Color histogram



Principal component analysis

Avoid doing this in a production system.

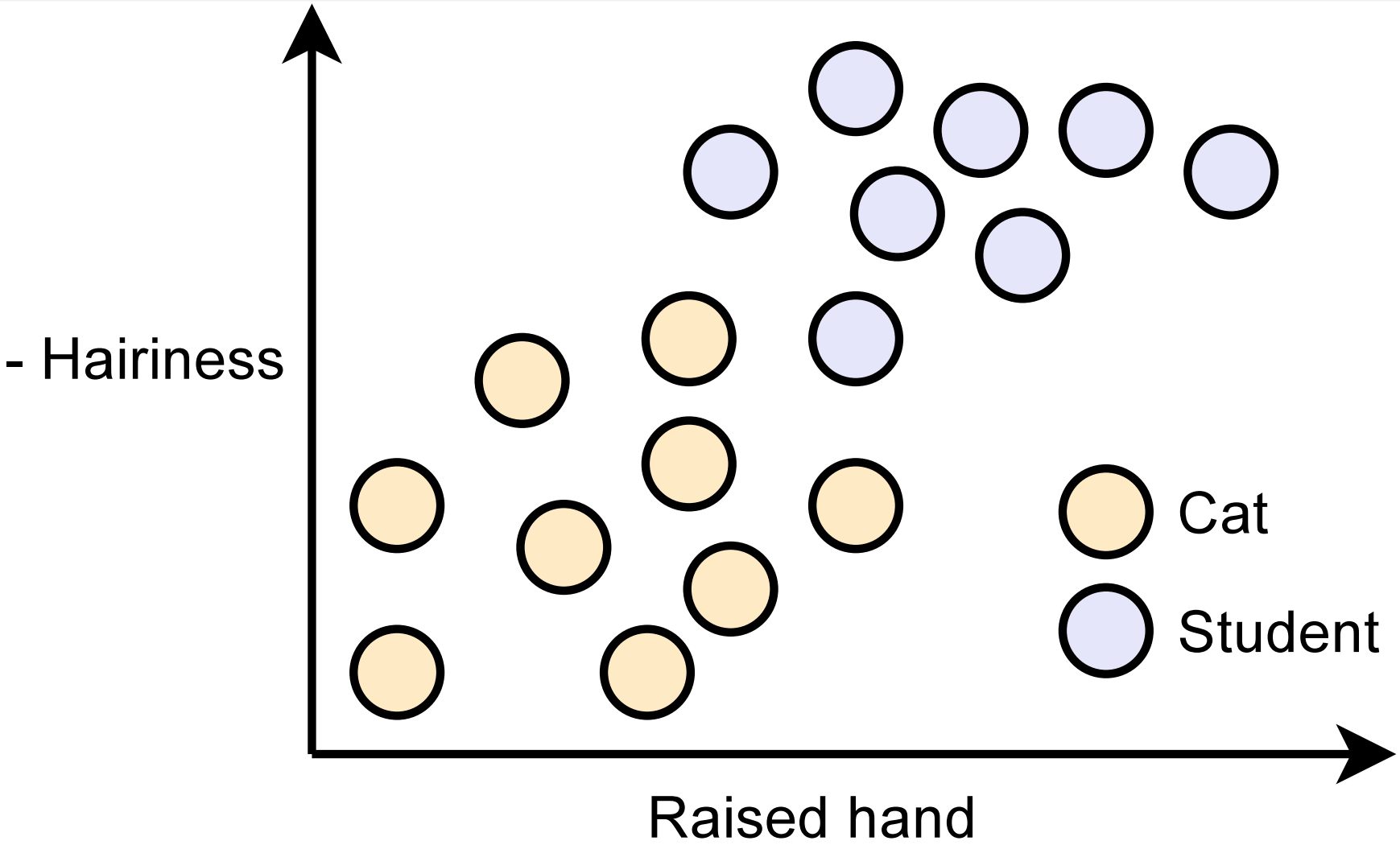
Valuable during learning process.

Transforms a data set from an input space to an output space in which dimensions are orthogonal and are ordered by decreasing variance.

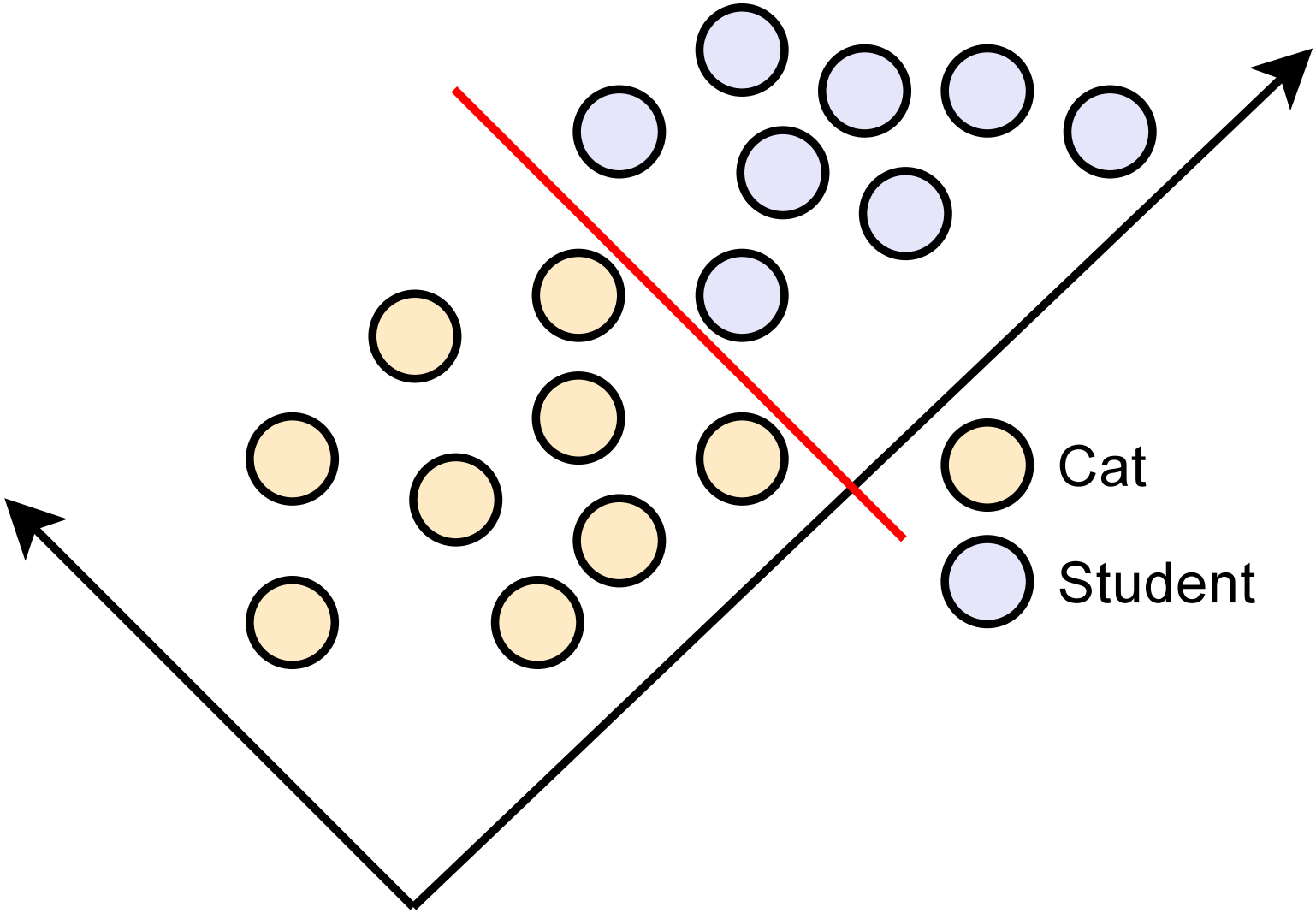
Too many dimensions to plot.

Truncation can be useful for data visualization.

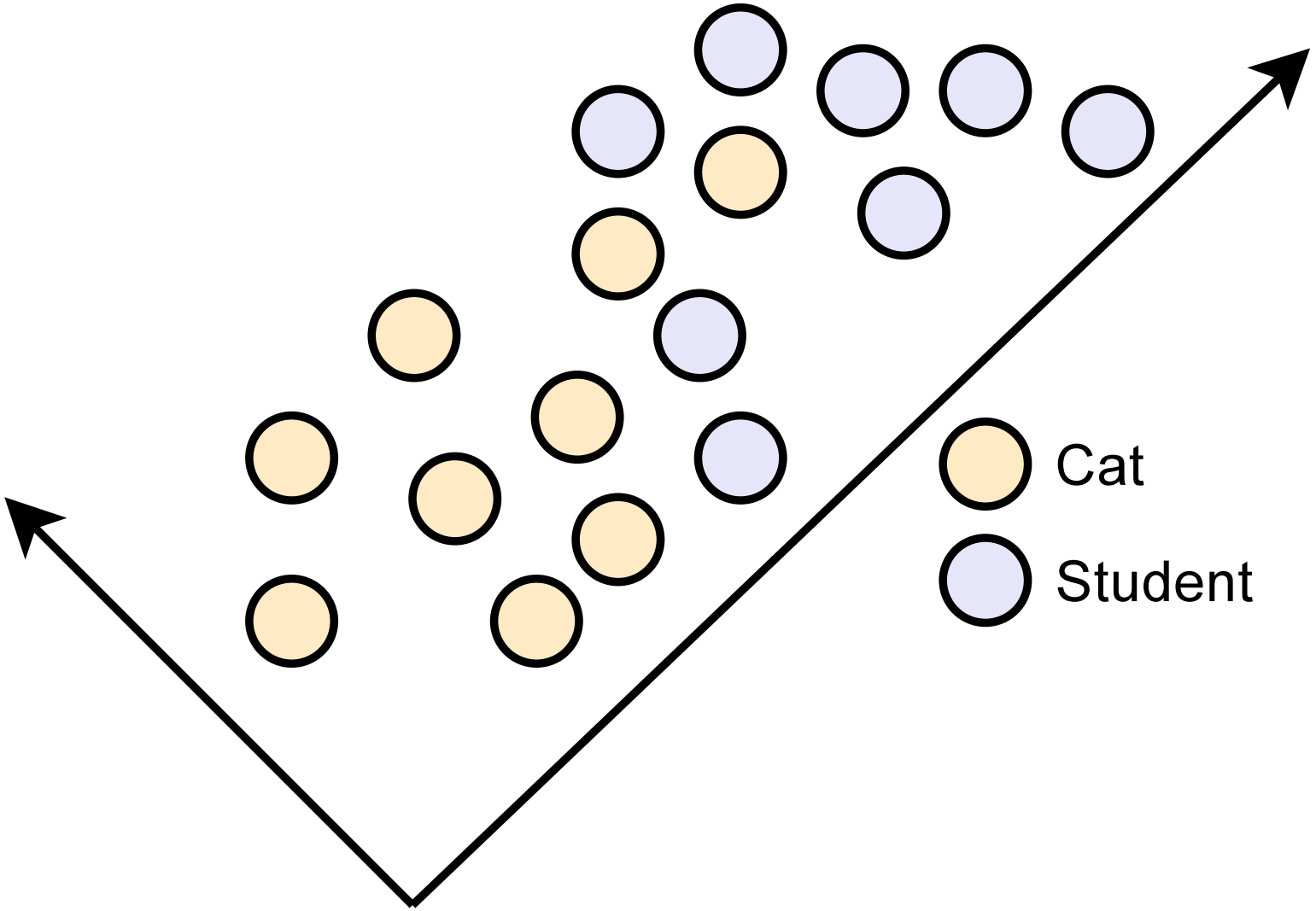
Two dimensions of original data (there are many more)



First two principal components

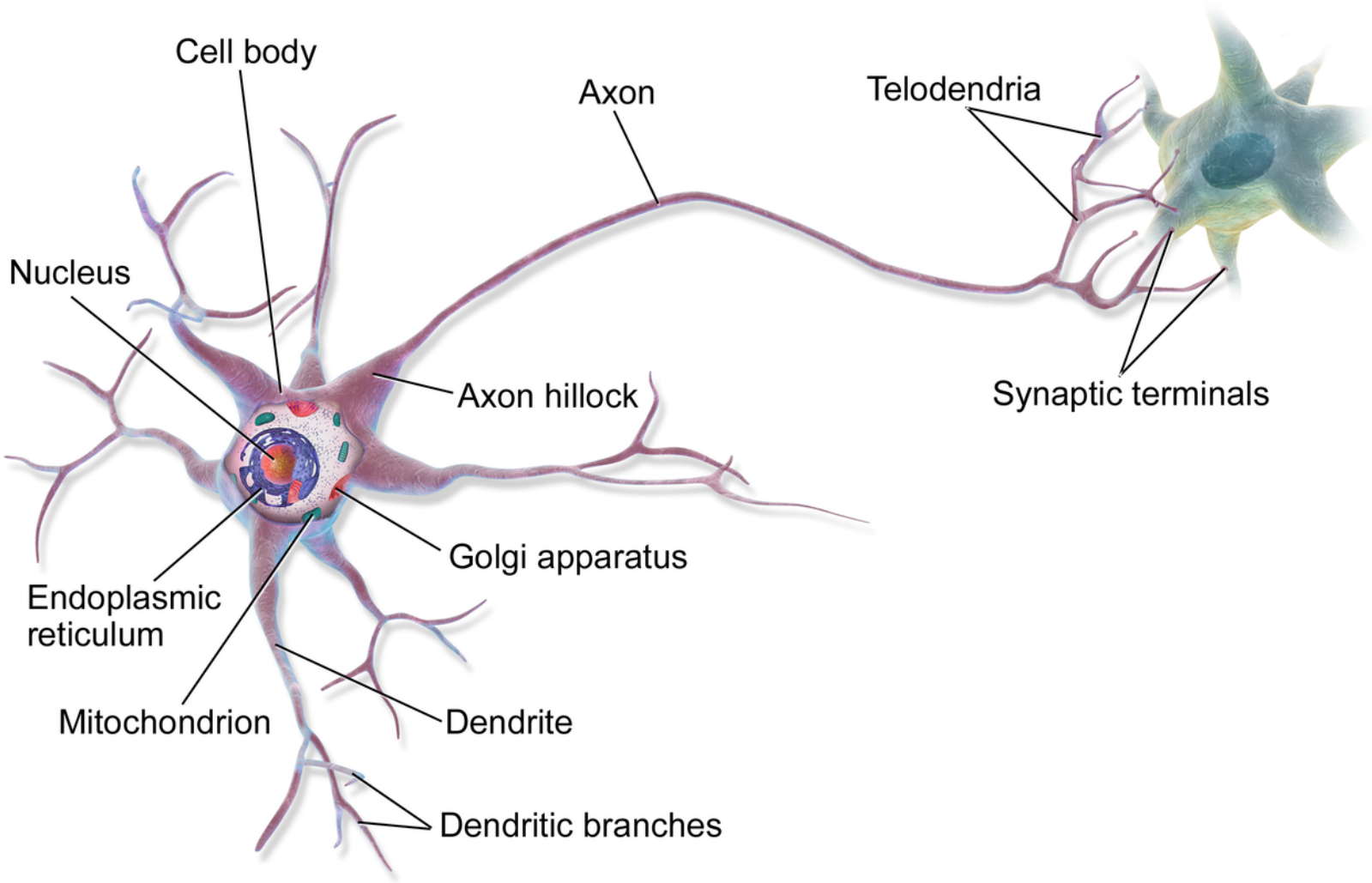


Cluster overlap in feature space



Section outline

Neuron


Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches

Structure

Many-layer redundant network.

Dendrite to axon flow is typical.

Cell body has a diameter of 4–100 μm.

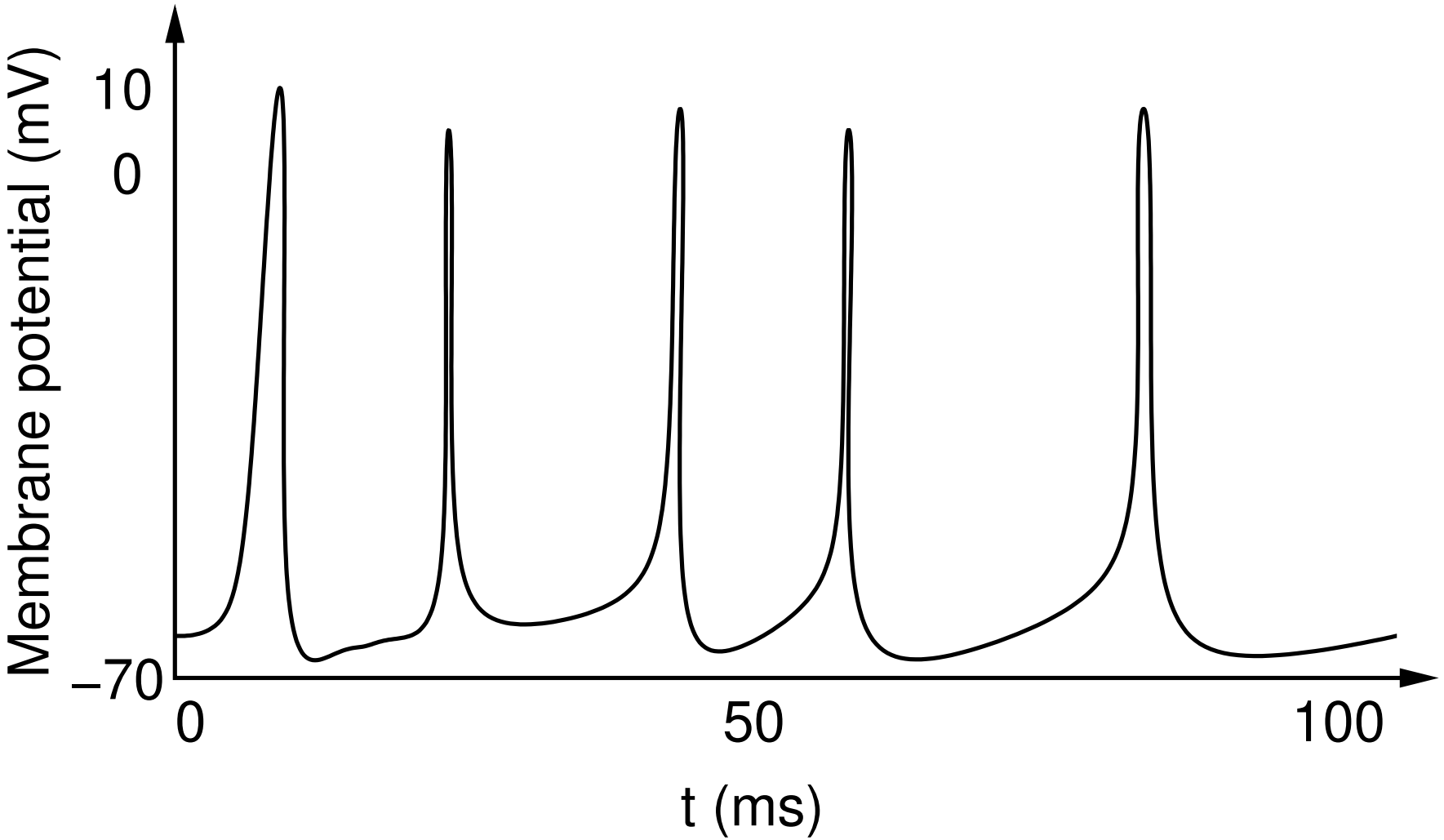
Axons are generally 10,000 times as long as the cell body diameter.

Axons move away from low-activity synapses to form other synapses.

Signalling

- Digital and analog exist.
- Digital used for longer distance propagation.
- Probably to resist noise.
- Analog used for shorter distance propagation.
- Digital generate discrete pulses by dumping ions into synapse.
- Most power consumed running ion pump to fight leakage.
- Frequency of firing indicates intensity.
- Fires when threshold reached.
- Near-simultaneous stimuli move farther toward threshold.

Spike / pulse train



Signalling

Integrate input delta functions, considering arrival times.

Fire when threshold reached.

Idle for refractory period.

Structural relationship with quality

What works better?

More or fewer?

Denser or less dense?

Aligned or unaligned?

High-power or low-power?

E. Genç, C. Fraenz, C. Schlüter, P. Friedrich, R. Hossiep, M. C. Voelkle, J. M. Ling, O. Güntürkün, and R. E. Jung, "Diffusion markers of dendritic density and arborization in gray matter predict differences in intelligence," *Nature Communications*, vol. 9, 2018.

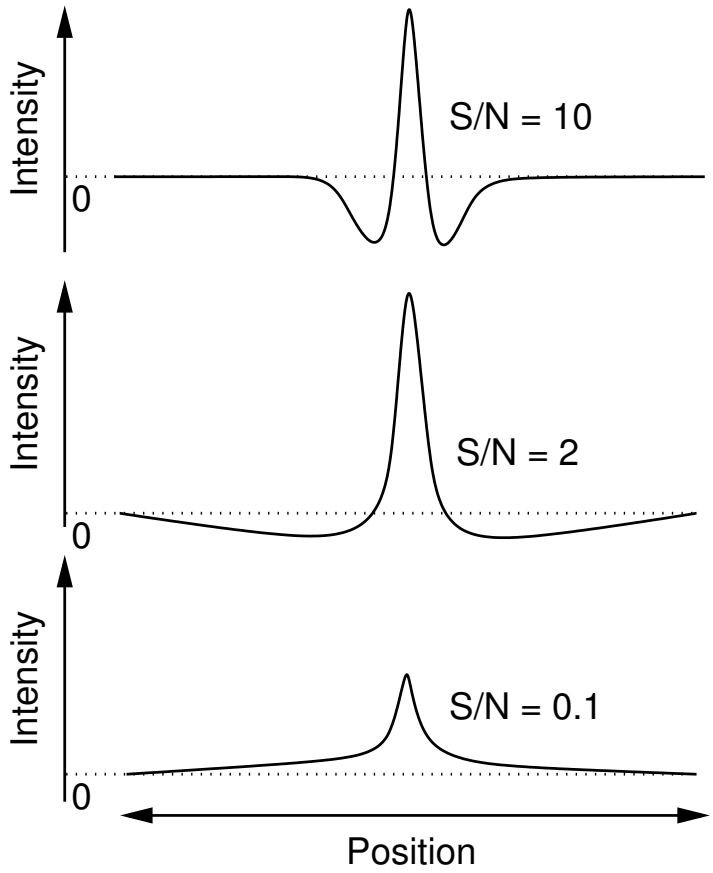
Optimization objectives?

2% mass, 20% power.

Probably

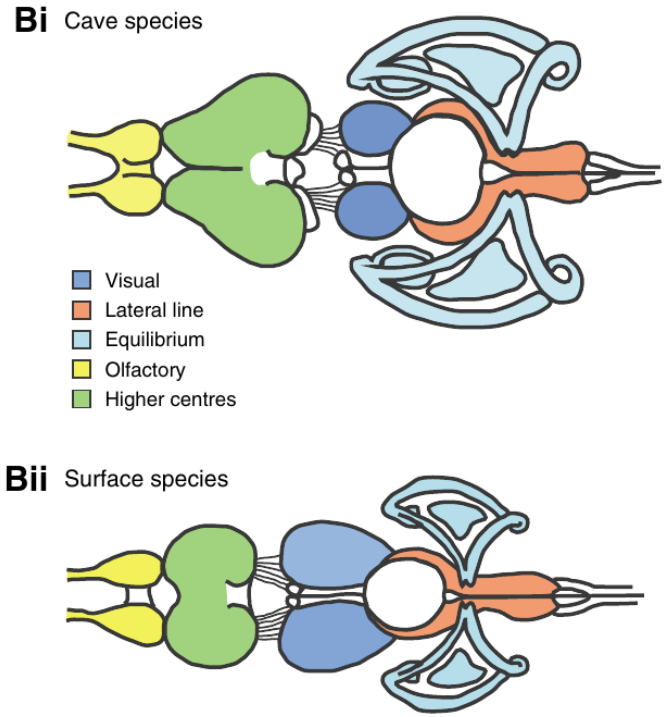
- power-constrained,
- estimated quality constrained, and
- energy-minimized.

Receptive field



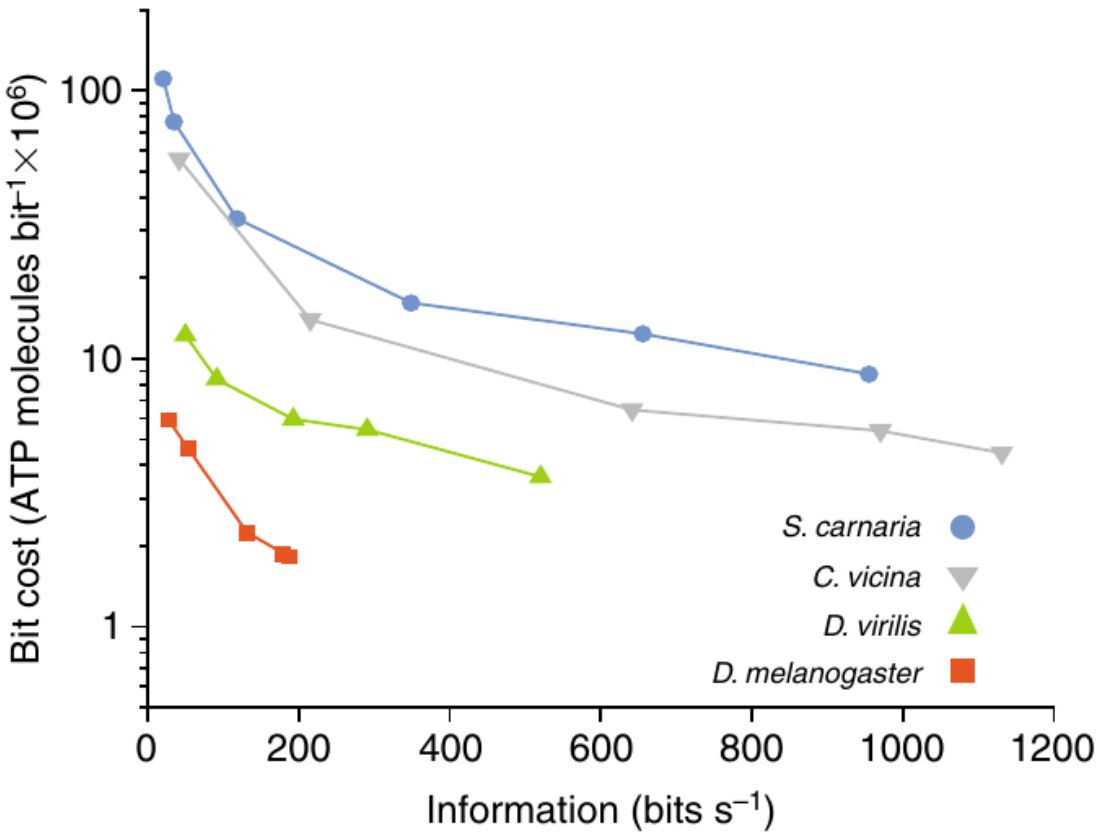
J. J. Atick and A. N. Redlich, "Toward a theory of early visual processing," *Neural Computation*, vol. 2, pp. 308–320, 1990.

Fish adaptation



J. E. Niven and S. B. Laughlin, "Energy limitation as a selective pressure on the evolution of sensory systems," *J. Experimental Biology*, vol. 211, pp. 1792–1804, Apr. 2008.

Insect signal processing



Roughly 5× energy efficiency within a species.

Why can't they gate?

Section outline

3. Machine learning
   - Manually designed classification algorithms
   - Machine learning
   - Neural networks perspective 1: biomimetic computation
   - Neural networks perspective 2: function approximation

Simple multivariate linear regression

Set parameters of hyperplane to minimize cost function relative to data.

Minimizing (root) mean squared error is common.

Fails for non-linear functions.

Non-linear also possible.

Neural networks are a way to approximate complex functions based on samples from the distribution.

Neural networks with linear activation functions

Can well approximate linear functions.

Number of layers does not change possible set of functions.

Neural networks with non-linear activation functions

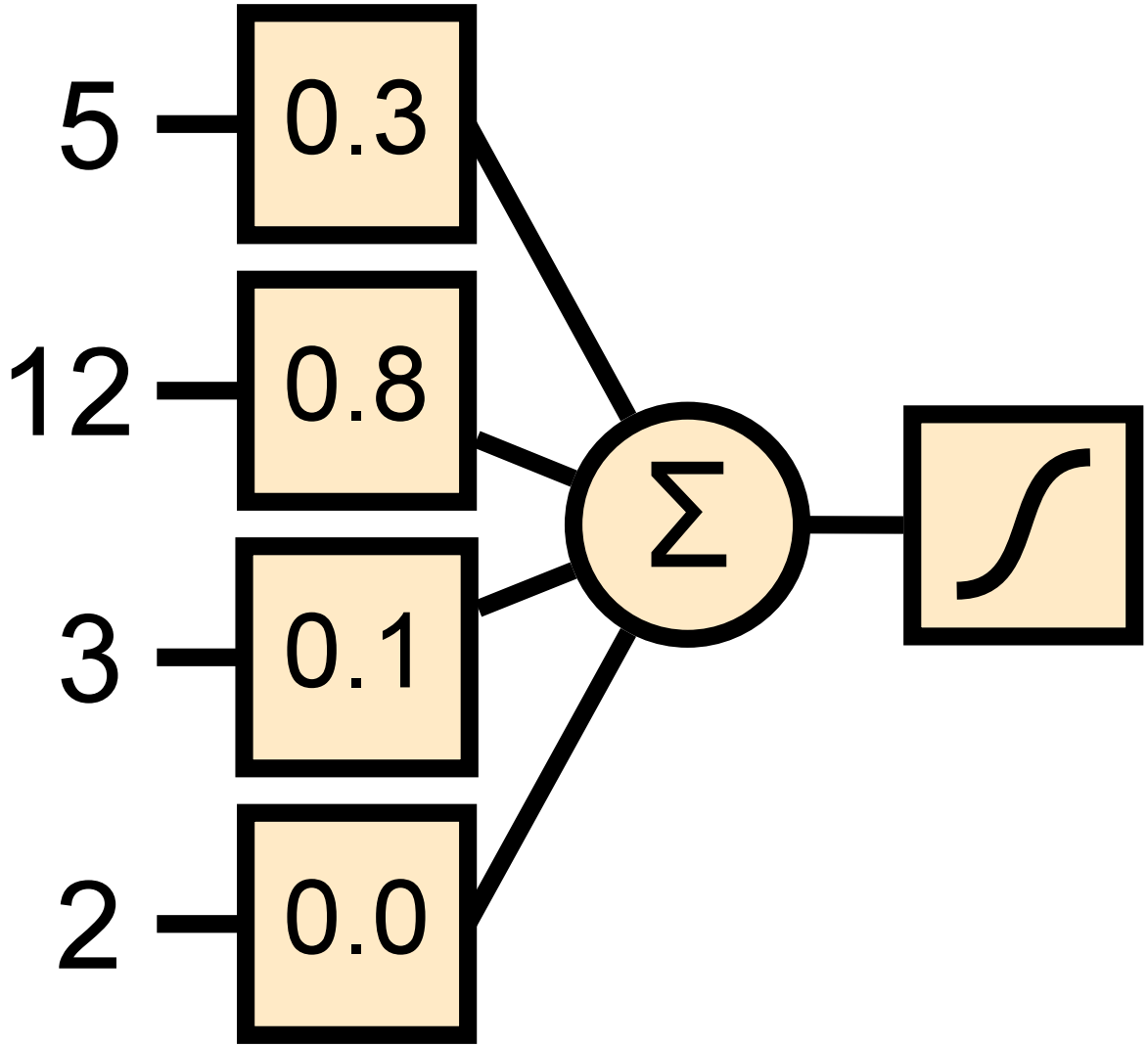
Can approximate non-linear functions.

Increasing number of layers may improve, or degrade, learning rate.

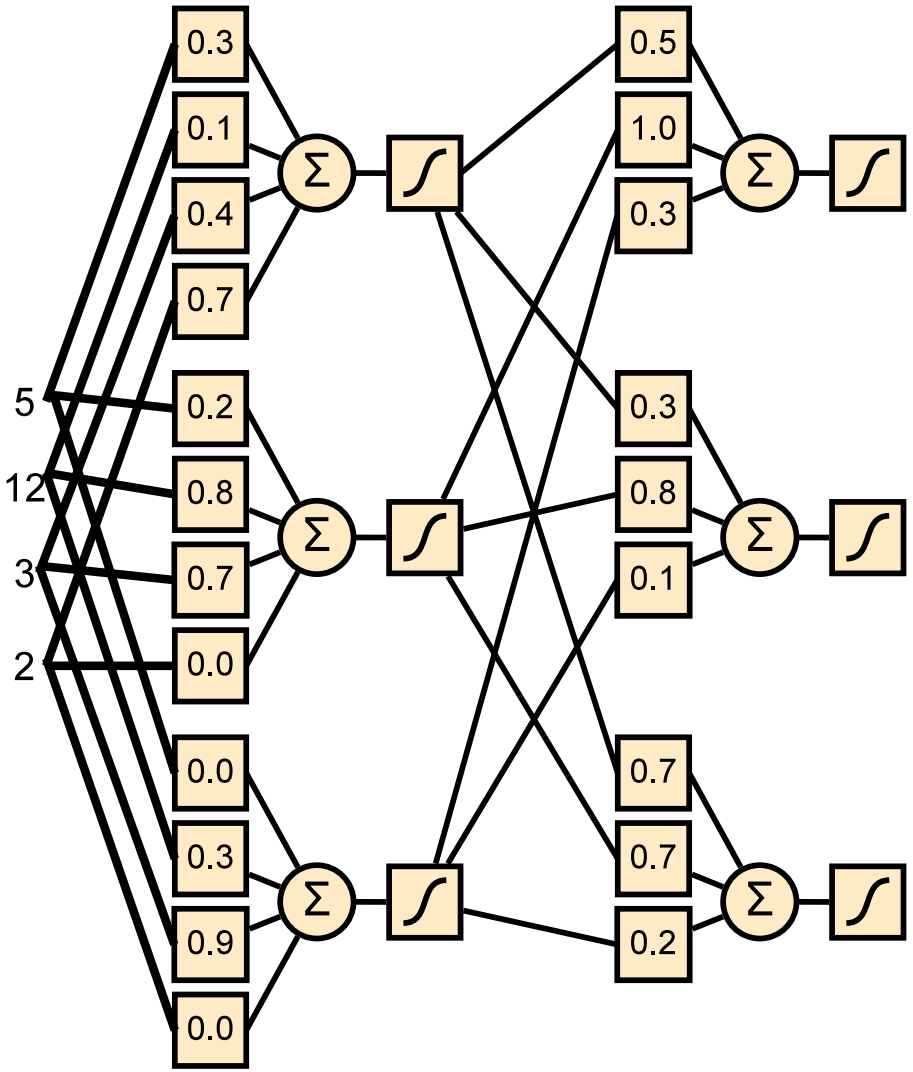
Allows simple non-linearities to be composed into more complex ones.

See S. Li, J. Jiao, Y. Han, and T. Weissman, "Demystifying ResNet," *arXiv 1611.01186*, 2016.

Neural networks



Neural networks



What does it do?

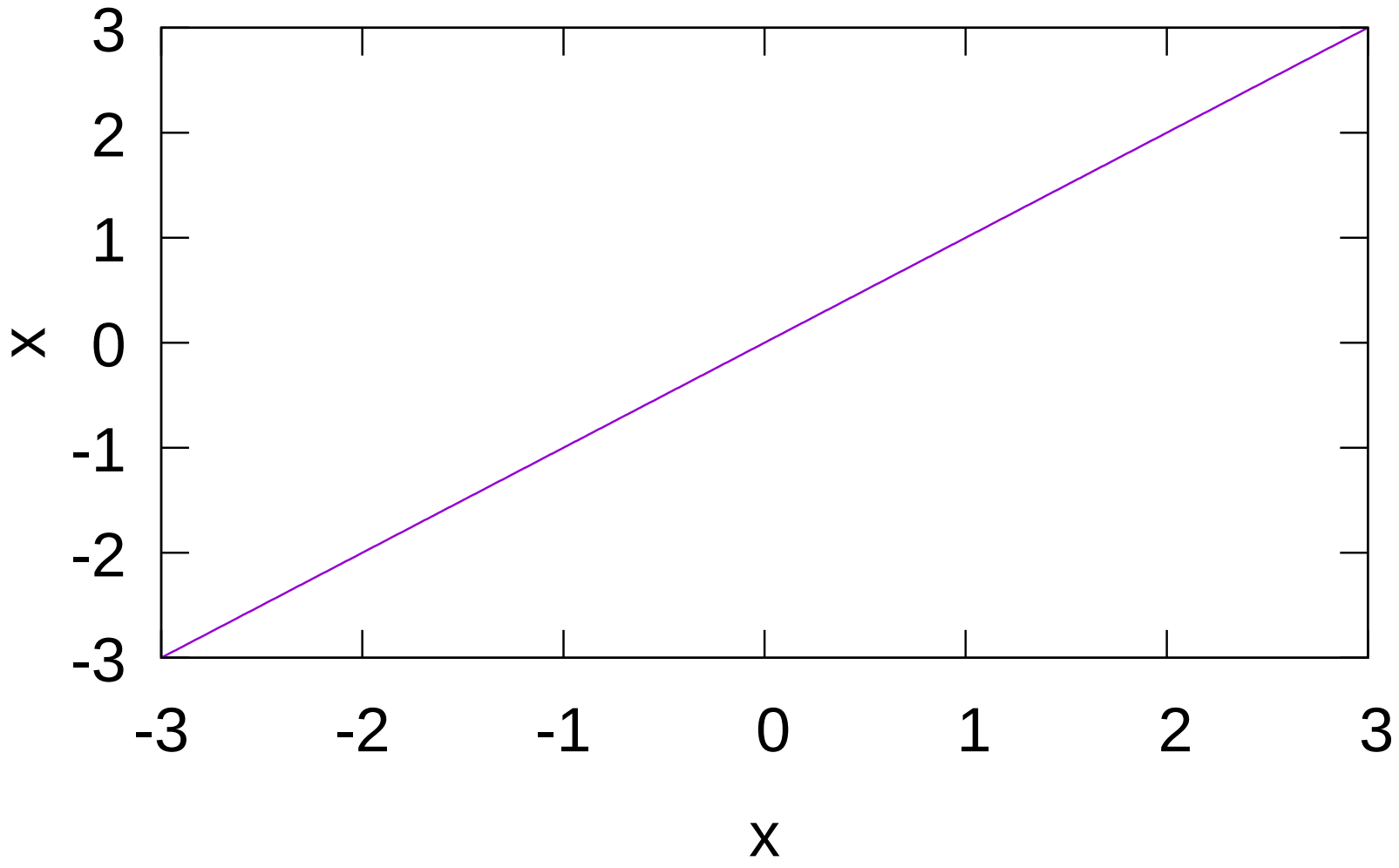
- Computes non-linear functions of inputs via forward propagation.
- Two or more layers sufficient for any Boolean function, but hard to train.
- Under some circumstances, deeper networks easier to train.
- How to train?

Math for one synthetic neuron

$$y = f\left(\sum_{k=1}^{n} i_k w_k\right)$$

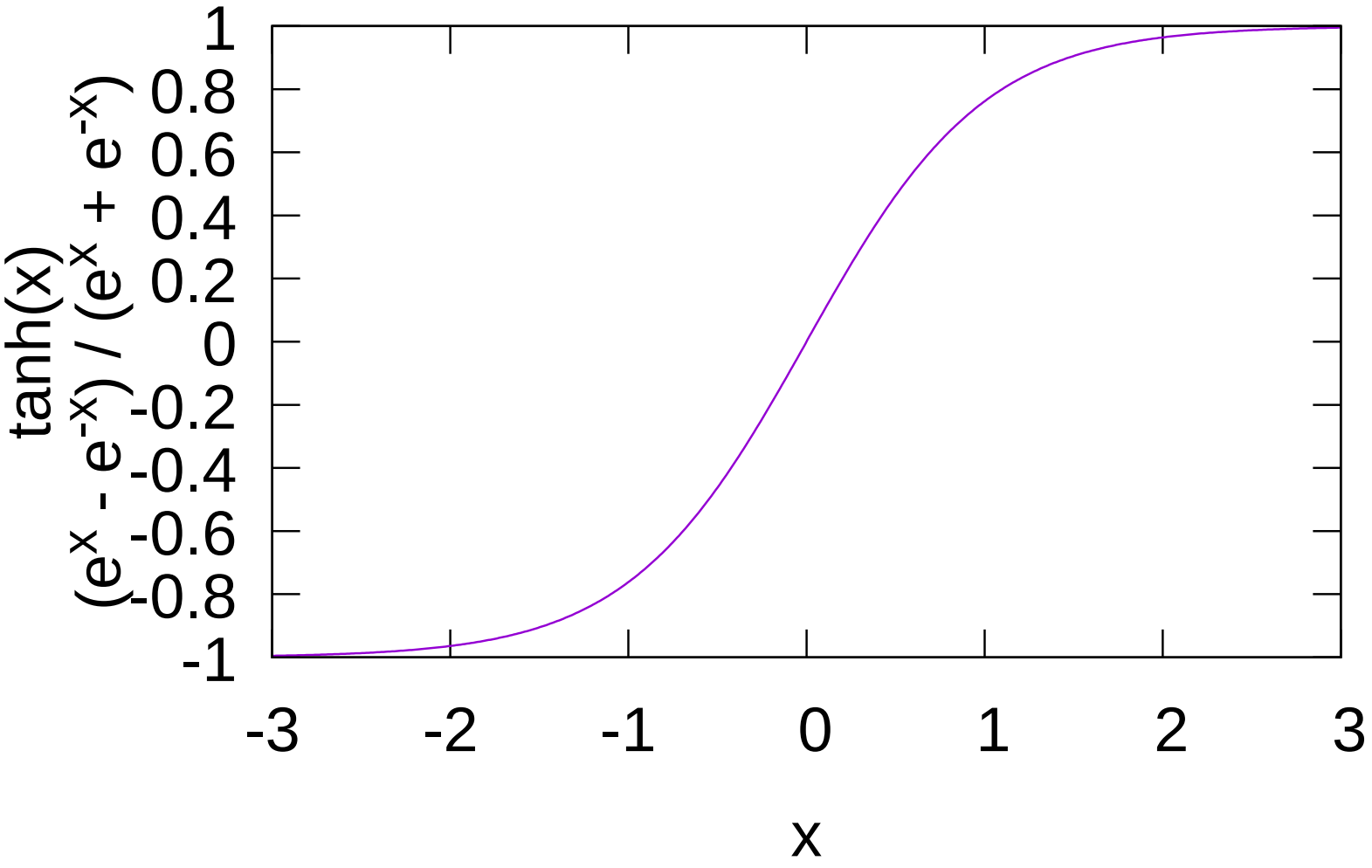
- n: number of inputs
- i: inputs
- w: input weights
- $f()$: activation function, commonly tanh()
- y: output

Example activation functions: identity



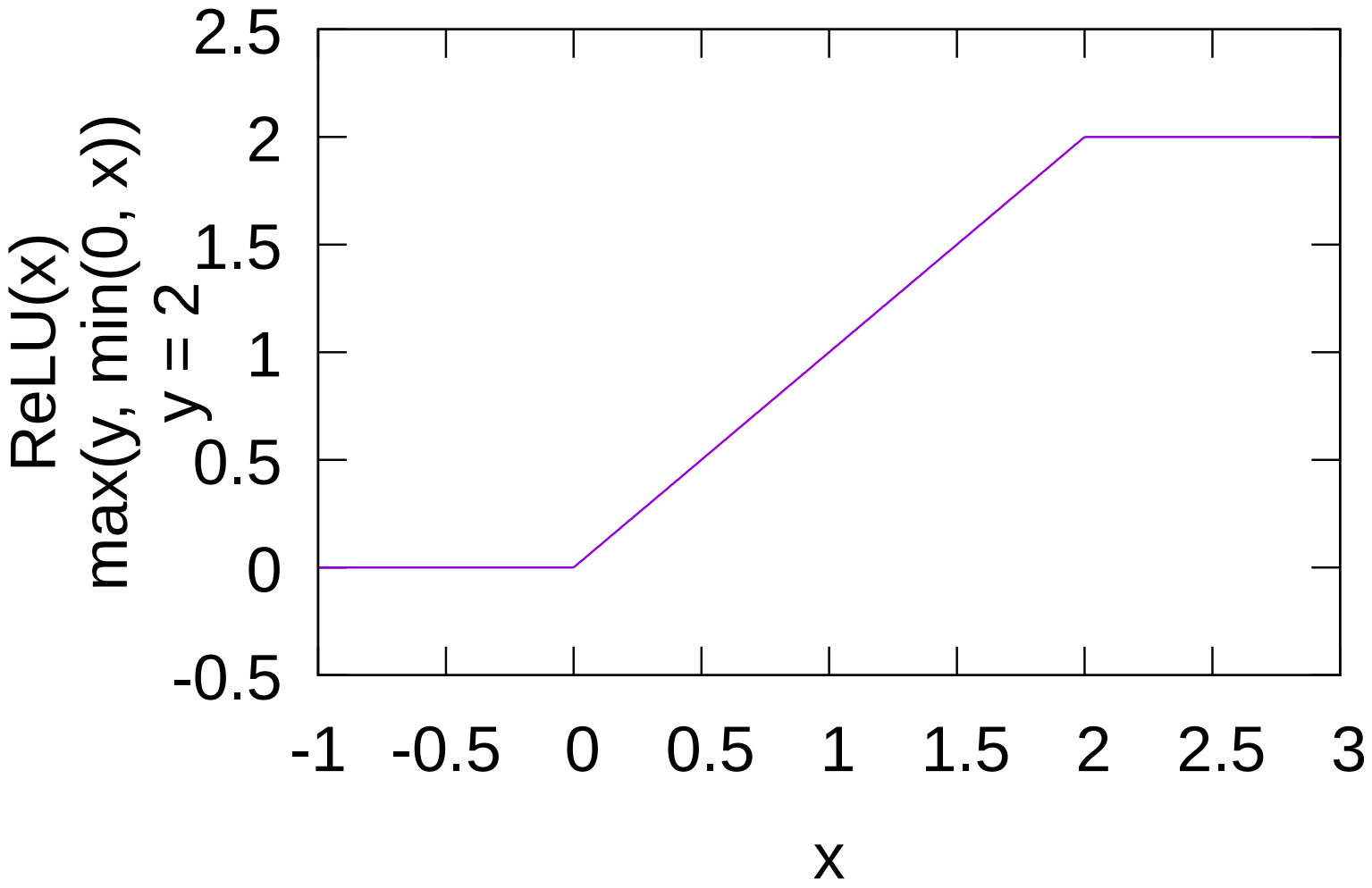
Not a very general function. Included as a base case.

Tanh



Was historically widely used.

Rectified linear



Widely used in state-of-the-art networks.

Types of learning

Unsupervised

- No teacher/trainer.
- Group similar (in feature space?) things together.

Supervised

- Teacher/trainer labels training samples.
- Learn to classify new samples as specified by teacher.

Training: gradient descent via randomized weight changes

- Change a weight.
- Check whether the results improved on training set.
- If so, keep the change.
- Too slow for most applications.

Training: backpropagation

Forward propagate training sample.

Determine error at outputs relative to correct output.

Propagate backward to compute node-level errors.

For each weight

- Multiply output error by input to find gradient.
- Subtract α times the gradient from the weight.
- $0 \leq \alpha \leq 1$.

This updates many weights every training event.

Back-propagation fundamentals

To enable rapid convergence for high-dimension networks, use gradient descent in weights.

Intuition: efficiently computing the change in all neuron's outputs w.r.t. changes in weights, potentially in prior layers.

Chains pose a problem: exponential number of paths through network.

Use chain rule to express each gradient as a function of those in subsequent layer only.

But there is more to it than this. . .

Linear in depth: every layer depends only on subsequent layer.

Quadratic in width for fully-connected layers.

Study Lagrange multipliers and see Y. LeCun, "A theoretical framework for back-propagation," in *Proc. Connectionist Models Summer School*, 1988, pp. 21–28.

Chain rule

$$F(x) = f(g(x))$$
$$F'(x) = f'(g(x))g'(x)$$

History

Even with these foundations, ML wasn't widely practical.

What changed?

Back-propagation was important ('70s and '80s for application to NN).

Most fundamentals from '80s and '90s.

Computers became fast enough for naïve implementations to work.

Sufficient data became available in some areas to enable training.

Back-propagation equations I

$$\delta^L = \nabla_a C \odot \sigma'\left(z^L\right)$$

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'\left(z^l\right)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

δ^L is the vector of errors associated with the output layer, L.

$\nabla_a C$ is the vector of the partial derivatives of the cost function w.r.t. the output activations.

$\odot$ is the Hadamard product.

Back-propagation equations II

$\sigma'\left(z_j^L\right)$ is the rate of change of the activation function σ at z_j^L.

δ^l is the vector of errors associated with layer l.

w^{l+1} is the weight matrix associated with layer l.

b_j^l is bias vector for layer l and neuron j.

w_{jk}^l is the row j, column k weight for layer l.

a_j^l is the activation for the jth neuron in layer l.

Following M. Nielsen's notation.

Deep learning

No universally agreed upon definition.

Using many-layer non-linear transformations to extract information of increasingly higher levels of abstraction from feature vectors or raw data.

Or learning parameters of a statistical model using multiple hidden layers between input and output.

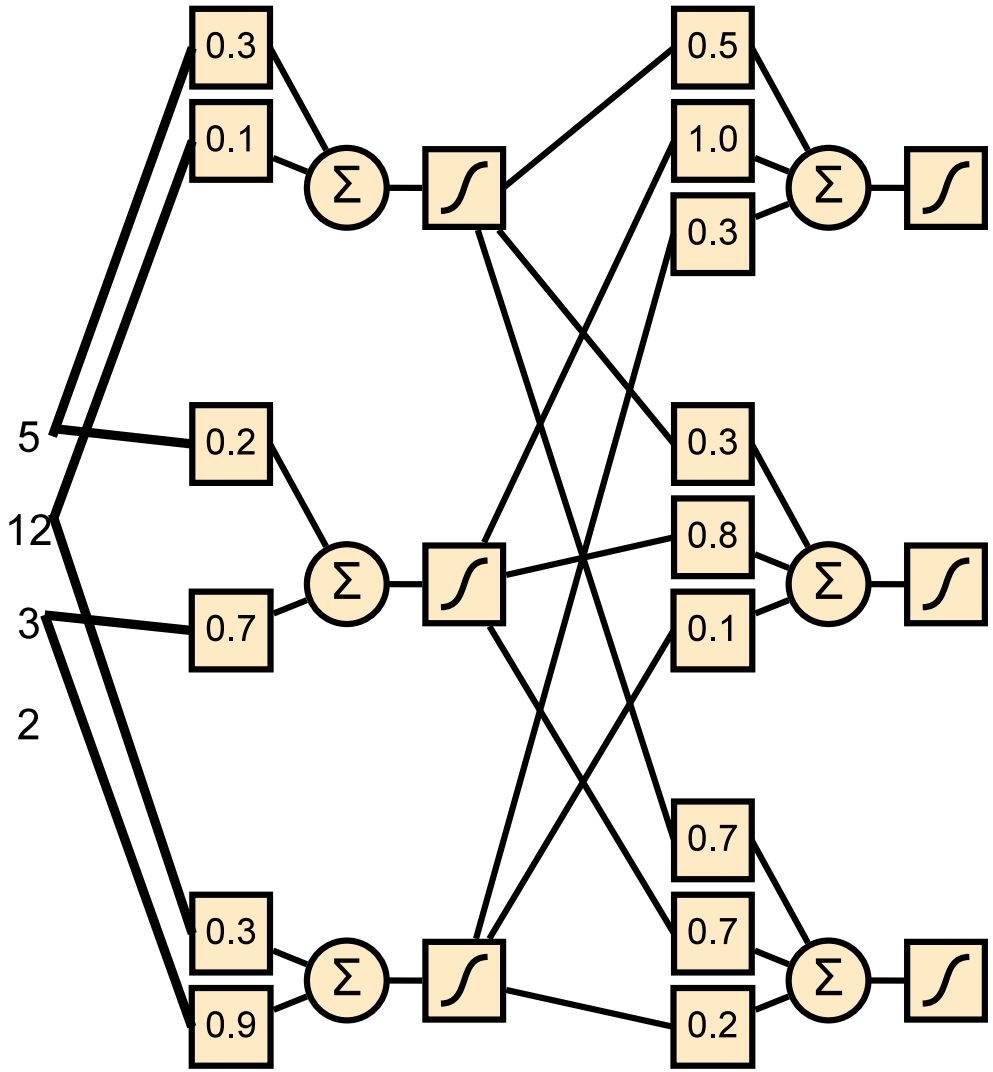
Deep learning applications

- Optical character recognition.
- Speech to text conversion.
- Games.
- Generation of realistic false video and audio of you.
- Text and speech interpretation.

Examples

https://playground.tensorflow.org

Incomplete connectivity



Convolutional neural networks

First layers apply learned convolution kernels to input.

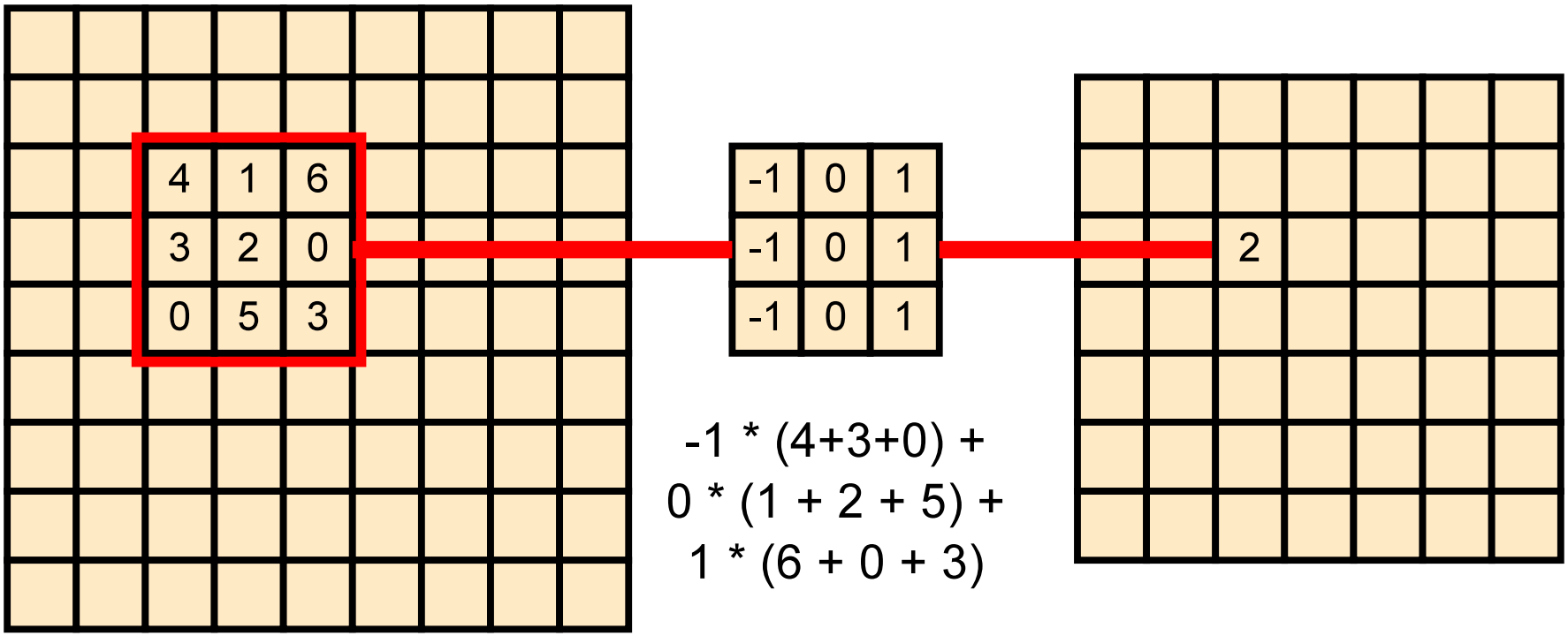
Heavily used for vision applications.

Convolution

$$K = \begin{bmatrix} k_0^0 & k_0^1 & k_0^2 \\ k_1^0 & k_1^1 & k_1^2 \\ k_2^0 & k_2^1 & k_2^2 \end{bmatrix}$$

$$\forall_{i=0+1}^{i\leq w-2} \forall_{j=0+1}^{j\leq w-2} g_{j-1}^{i-1} = \sum_{m=0,n=0}^{m\leq 2,n\leq 2} k_n^m \cdot p_{j+n}^{i+m}$$

Convolution example



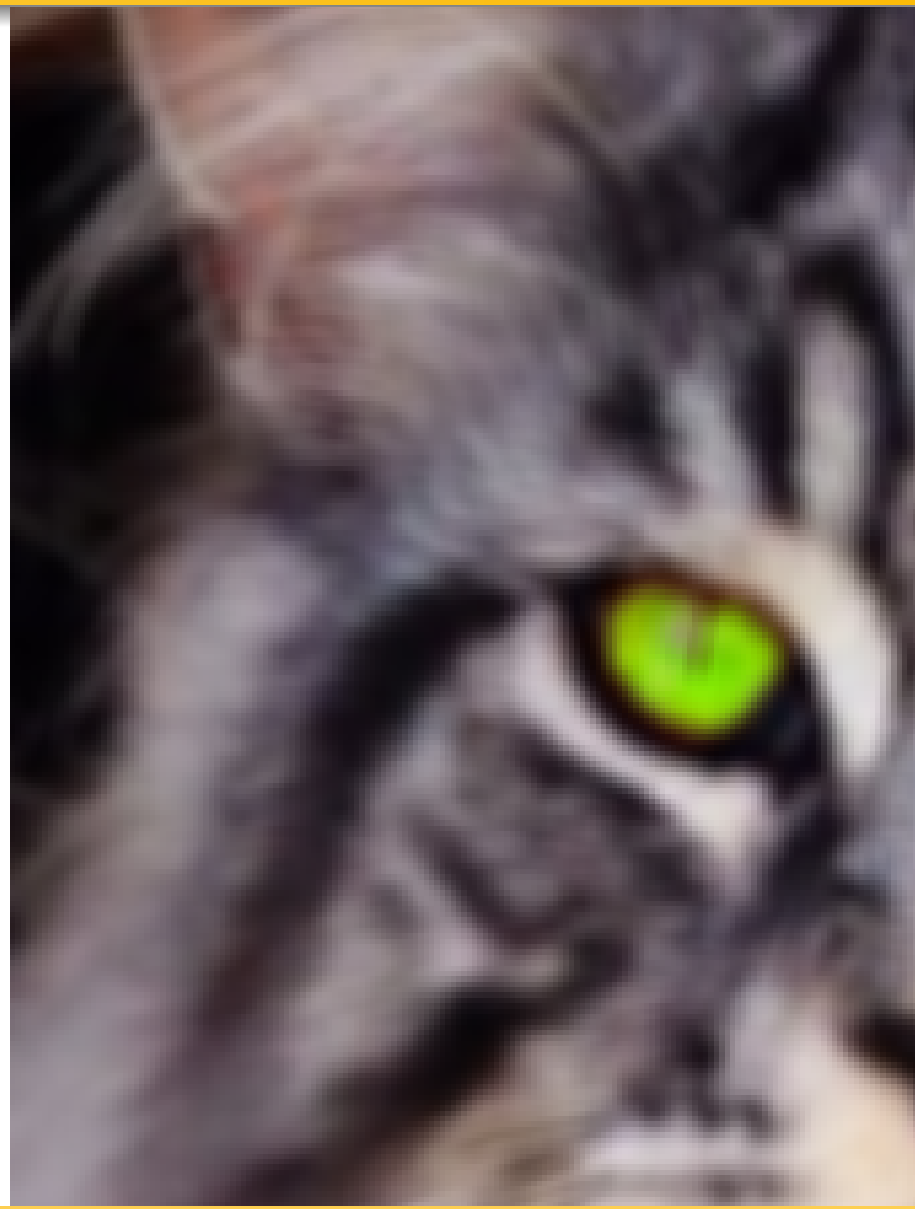
Vertical edge detection

$$K = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Blurring

$$K = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Blurring



CNN

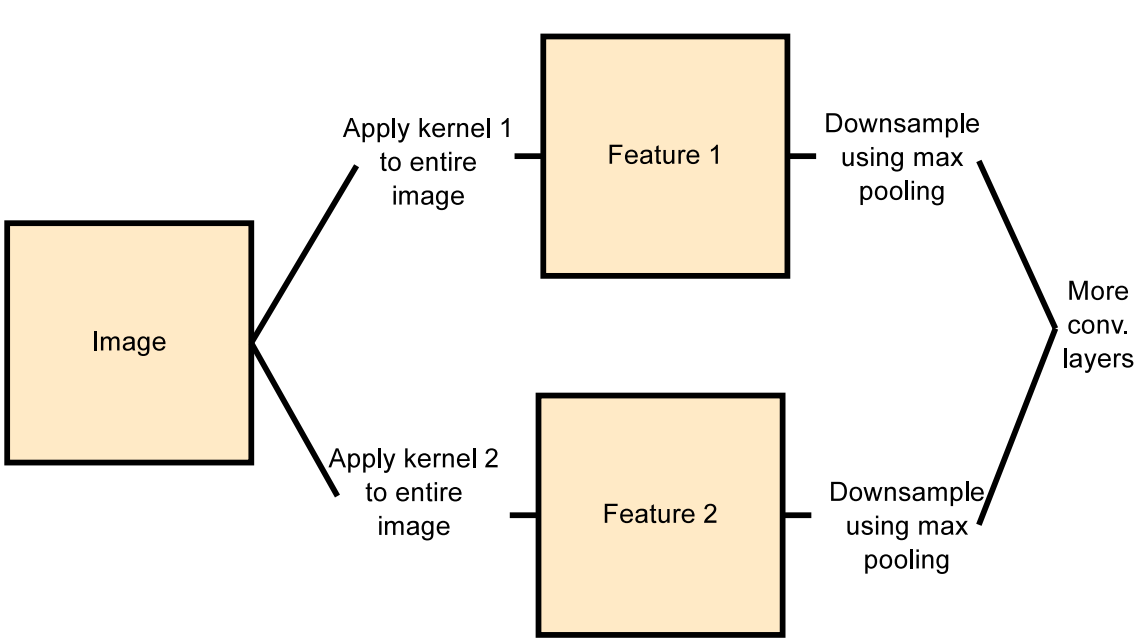
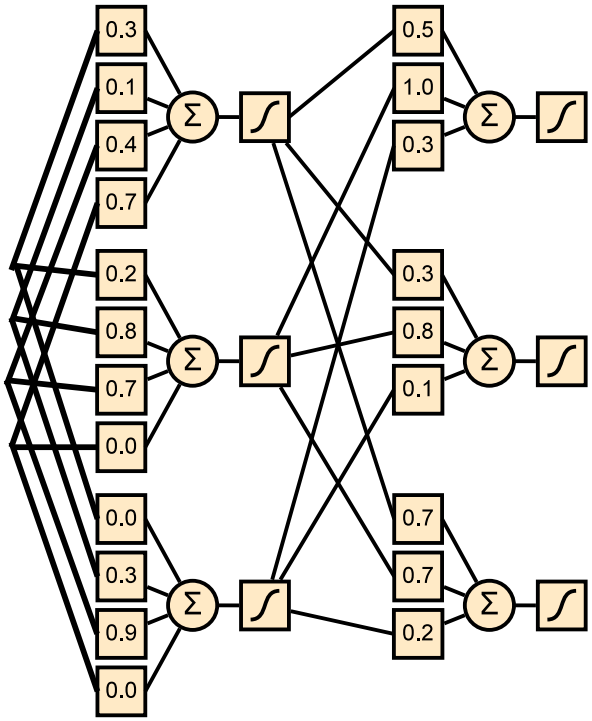
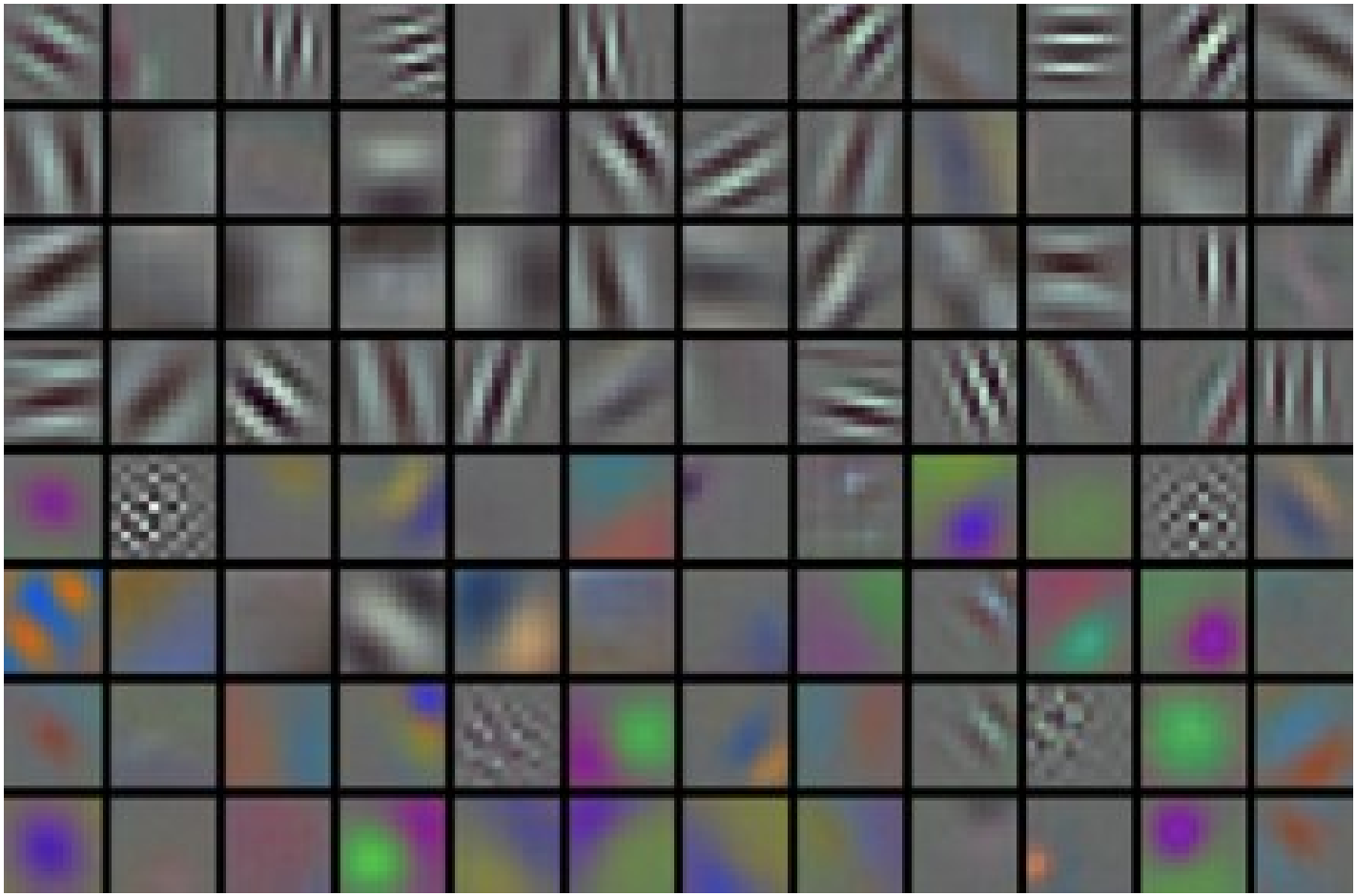


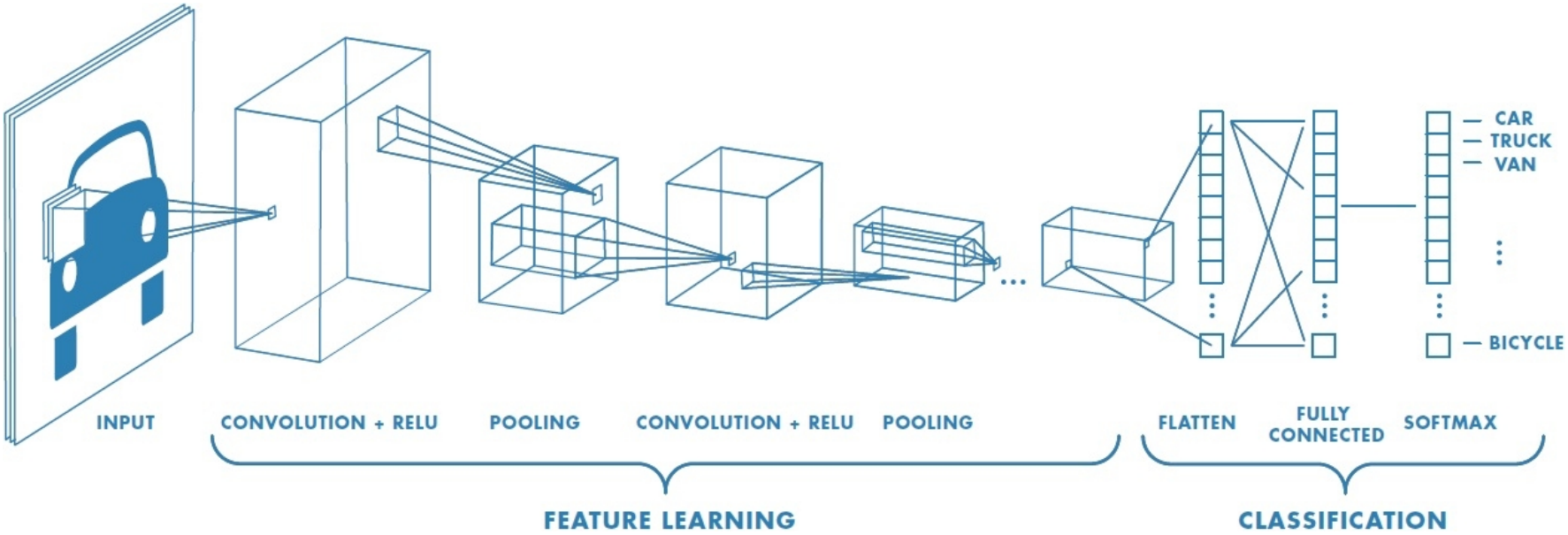
Image
Apply kernel 1 to entire image
Feature 1
Downsample using max pooling
Apply kernel 2 to entire image
Feature 2
Downsample using max pooling
More conv. layers


0.3
0.1
0.4
0.7
0.2
0.8
0.7
0.0
0.0
0.3
0.9
0.0
Σ
Σ
Σ
0.5
1.0
0.3
0.3
0.8
0.1
0.7
0.7
0.2
Σ
Σ
Σ

First-layer convolutional kernels



Convolutional neural network structure



Courtesy MathWorks.

Convolutional neural network applications

- Face recognition.
- Scene labeling.
- Image classification.
- Action recognition.
- Human pose estimation.
- Document analysis.
- Natural language processing.
- Speech recognition.
- Text classification.
- Audio classification.

Temporal sequences: unrolling

Select n discrete time steps into the past.

For each, provide additional inputs.

Implications: (often very) superlinear increase in network size.

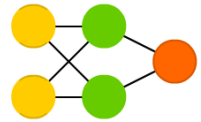
A mostly complete chart of

Neural Networks

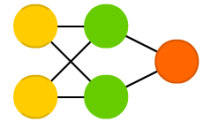
- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)

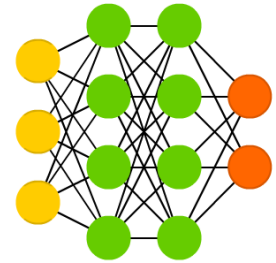
Feed Forward (FF)



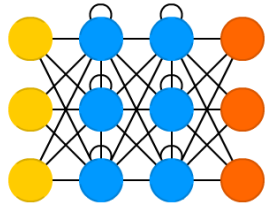
Radial Basis Network (RBF)



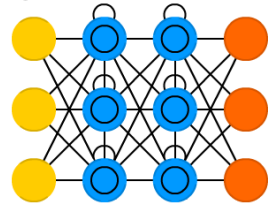
Deep Feed Forward (DFF)



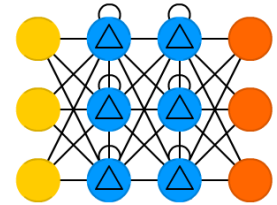
Recurrent Neural Network (RNN)



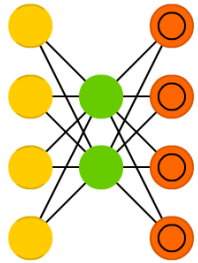
Long / Short Term Memory (LSTM)



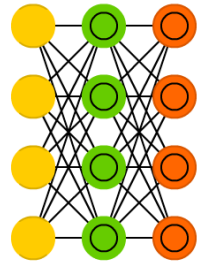
Gated Recurrent Unit (GRU)



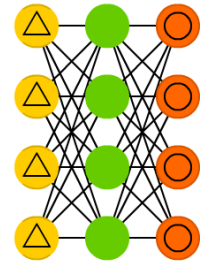
Auto Encoder (AE)



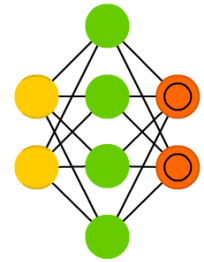
Variational AE (VAE)



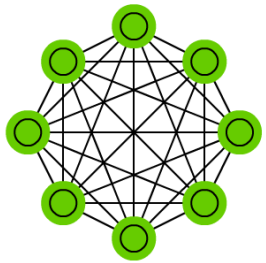
Denoising AE (DAE)



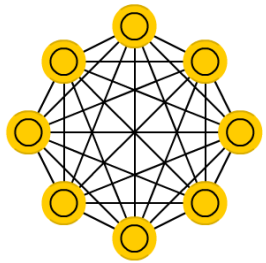
Sparse AE (SAE)



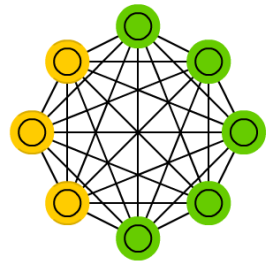
Markov Chain (MC)



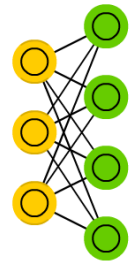
Hopfield Network (HN)



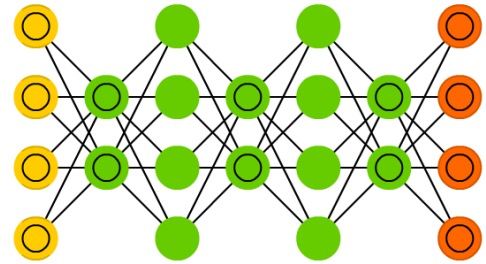
Boltzmann Machine (BM)



Restricted BM (RBM)



Deep Belief Network (DBN)



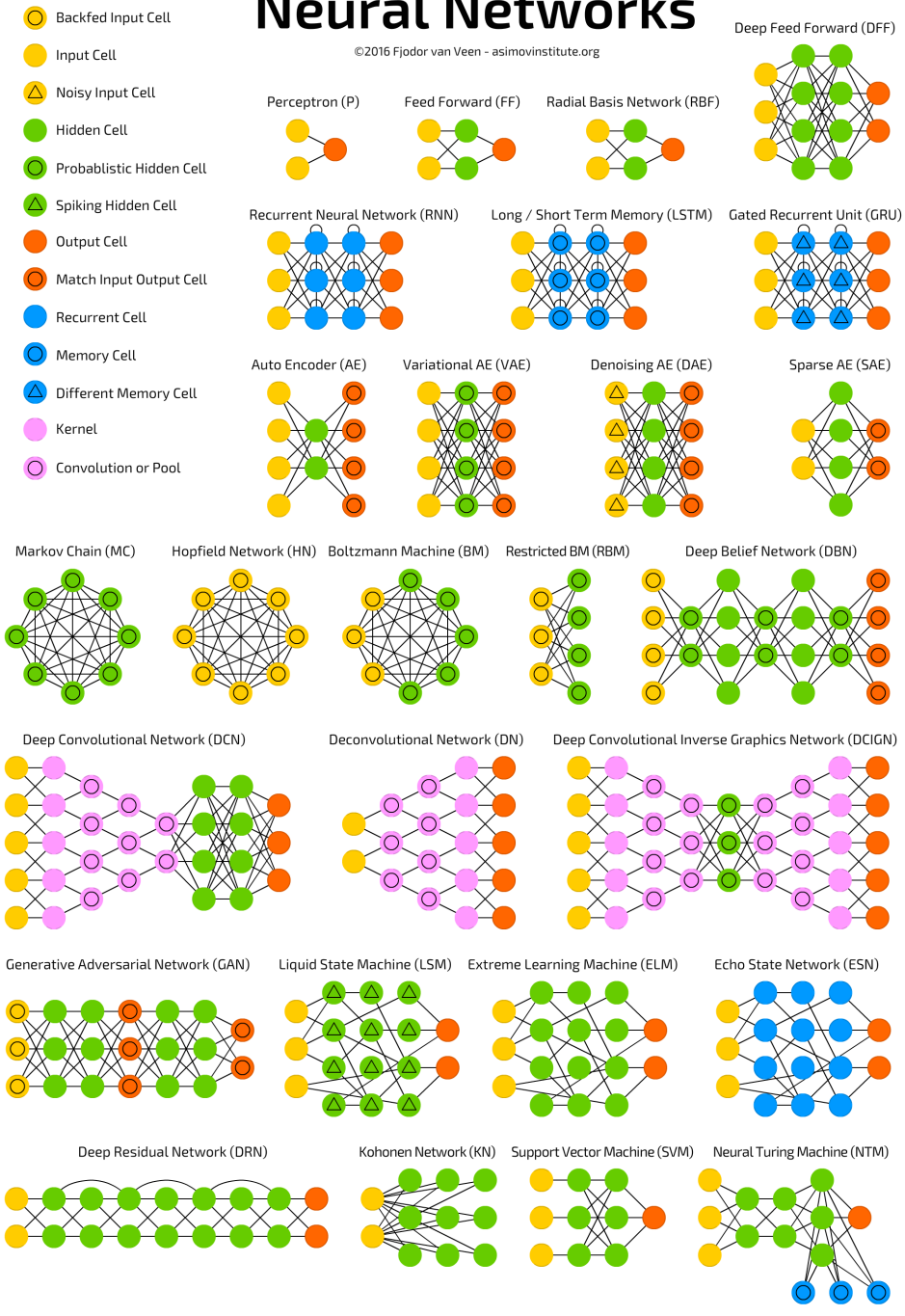
Deep Convolutional Network (DCN)

Deconvolutional Network (DN)

Deep Convolutional Inverse Graphics Network (DCIGN)

A mostly complete chart of

Neural Networks



Outline

Lecture format

What comes next is best interactive.

Take notes, as the slides merely provide an outline.

If it isn't clear where ideas are coming from, or what their context is, please ask (you aren't the only one who is curious).

There is no rush. The goal is to learn.

Overview of field

Unusually fast-moving.

Surprisingly emperically driven: tendency to latch on to spurious conclusions and neglect/miss/rename historical findings.

Can be very resource-intensive.

Not mature: just now moving to understand and make efficient.

Parsimony

The simplest explanation consistent with the evidence should be preferred.

Ptolemaic $\rightarrow$ Copernican shift.

Just a heuristic, but has proven so powerful that it is intertwined with the scientific method.

Deep neural networks often have this, and regularization techniques are often used to enhance it.

Sometimes it undermines test performance.

Regularization

Introducing a, usually structural, cost or constraint into the loss function.

For example, penalizing for weights far from zero or rewarding networks for which subsets produce high accuracy (drop-out).

Training, validation, and test

Training data distribution may not match test distribution: covariate shift.

In underparameritzed models, "over"-training to exactly match training distribution data points might work against simplicity bias.

In overparameterized models, exact memorization is typical, so a more fundamental mode of thinking is required.

This is a place where concentrating on the actual problem, instead of a simplified but more manageable version that mismatches reality, can help.

Goal: Learn the (unknown) test distribution.

Can we estimate the most likely deviation basin of training and test distributions and optimize the loss for the estimated distribution of test distributions?

Pruning

Pruning first fails.

Filters are not naturally orthonormal.

This undermines pruning.

Orthonormalization dramatically improves accuracy / efficiency for pruned networks.

Causal learning

Correlations lack directionality.

Directionality required to distinguish causes from effects and root from proximal causes.

Can use interventions, in which all but one variable are held constant and that one variable is modified, then observe effect.

There are also heuristics, often parsimony-based, to construct causal graphs.

Relatively little activity in this field, so far, but may be necessary for GAI.

Spurious cues

Spurious: not using properties fundamental to the inference decision.

Spurious cues are often simpler than more fundamental properties.

Very recent work on this topic.

Context: overparameterized deep networks.

Networks not linearly connected use different mechanisms.

All optimal basins have similar loss and are connected by quadratic-structure valleys.

Outline

Deeply embedded machine learning

Intelligent mobile and IoT devices can solve many problem.

- Transportation.
- Personalized health care.
- Athletic training.
- Robotic assistants.

But it can be slow and energy-hungry.

1/3–1/2 the energy in autonomous cars goes to the machine learning systems.

... and that's without learning!

Neural network efficiency

Until recently, neural networks were grossly inefficient.

Review

- '80s and '90s: fundamentals established
- '00s and '10s: large data sets become available and computers become fast enough to handle conventional networks

What's next?

Low-power embedded devices

Why not just send everything to the "cloud"†

- Communication energy commonly dwarfs computation energy.
- Bit communication takes 670×–33,000× the energy per bit computed.
- Low-power communication standards are slow.

Requirements

- Solve hard analysis and decision making problems.
- Consume little energy.
- Fast.
- Without much communication.
- On wimpy, battery-powered IoT and mobile devices.

† Somebody else's computers.

Fundamental problem

Waste in

- capture,
- transfer, and
- computation.

Examples

- Needlessly high value or parameter precision.
- Useless parameters.
- Useless input data.
- Useless training events.
- Redundancy in learning representation.
- Many more.

Recent efficiency-related advances

- Narrow weights.
- Pruning.
- Decoupled training / use.
- Appropriate architectures.
- Heterogeneous sampling / analysis.
- Many more coming...

What's next?

Now practical to automatically train statistical models for better performance in many vision and gaming applications than humans.

Efficiency is rapidly increasing.

Smart things will be common.

Federated / distributed learning for non-IID data.

Exploiting sparse data.

Product design will frequently involve the use, modification, and development of machine learning algorithms.

Guesses about future

Machine learning technologies will be very widely used by those who most love power over others to automatically manipulate the flow of information through governmental and private censorship and propaganda.

Don't like it? Then don't create it. Instead develop technologies to circumvent and counteract such controls.

For every person, there will be hundreds or thousands of tiny, inexpensive devices that understand their surroundings and make decisions based on them.

You, the designers, have the responsibility to decide what goals these devices have, and who they will serve.

Efficient algorithms

Omit computation on useless input data.

Eliminate redundancy within analysis networks

- Prune near zero weight edges after training.
- Consider only local relationships at some network depths.

More efficient training methods, e.g., residual networks.

Reduce edge weight precision.

Biomimetic signal processing

Pattern electronic systems using design idioms from energy-efficient biological systems.

Mimic structure of application-specific signal processing pipelines.

Application-specific hardware

Mirror network structure with hardware.

Enable parallel operations.

Improve communication latency and throughput among algorithmically adjacent computational elements.

Specialized devices that efficiently mimic neural properties.

Note that even the brain uses digital encoding for long-range transmission.

Partitioned networks

Partial of analysis on edge devices, part in cloud.

Designing algorithms with low-cost cut points is challenging.

Could adapt to changing computation/communication costs.