

CS-UY 4563: Lecture 15

Neural Networks

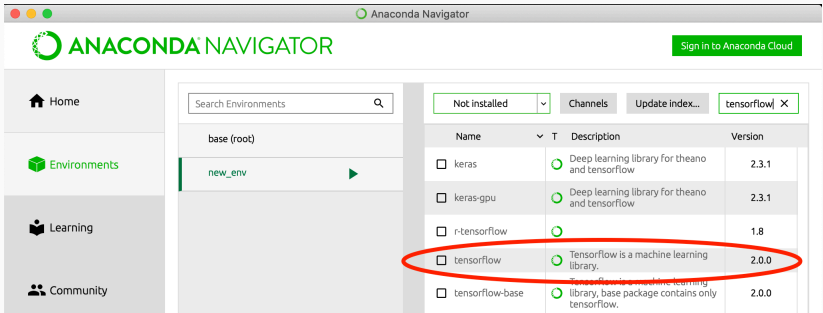
NYU Tandon School of Engineering, Prof. Christopher Musco

- Project topic/teams due on **tonight** via email.
 - Sign up for a meeting time after you send me the email.
- Lab due **Thursday night**.
- My office hours from 1-3pm today.

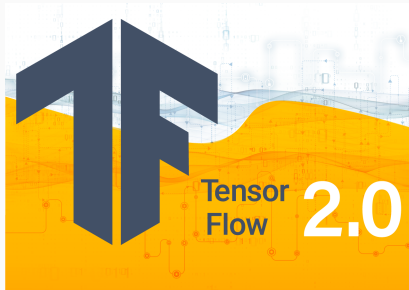
Lab `lab_mnist_partial.ipynb` due next Thursday, 4/9.

- Covers kernel logistic regression and SVMs, which should be useful in many projects.
- Requires **Tensorflow** (easiest way to load MNIST data).
- Popular ML library focused on neural networks: first released by Google in 2015. We will use along with the high-level **Keras** library (included in **Tensorflow**) for the next part of the class.





- Installation can be done directly in Anaconda navigator, via **pip**, or comes pre-installed in Google Collab.
- Install Version 2 so we're all on the same page.



Tensorflow 2 was only released in Fall 2019, so if you are having issues and need Google for answers, make sure to restrict the dates of your search.

The hot-topic in machine learning right now.

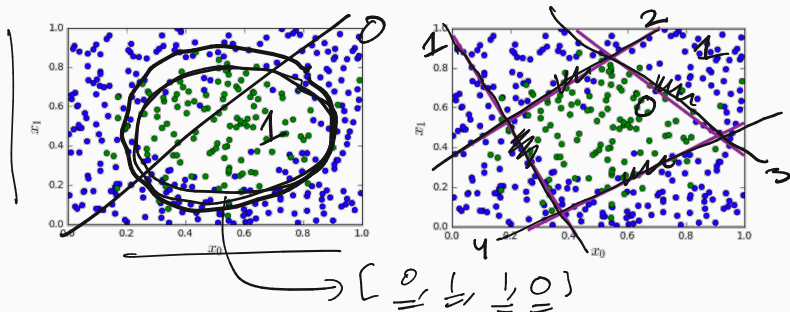


Focus of investment at universities, government research labs and funding agencies, and now large tech companies.

Studied since the 1940s/50s. **Why the sudden attention?** More on history of neural networks at the end of lecture.

SIMPLE MOTIVATING EXAMPLE

Classification when data is not linearly separable:

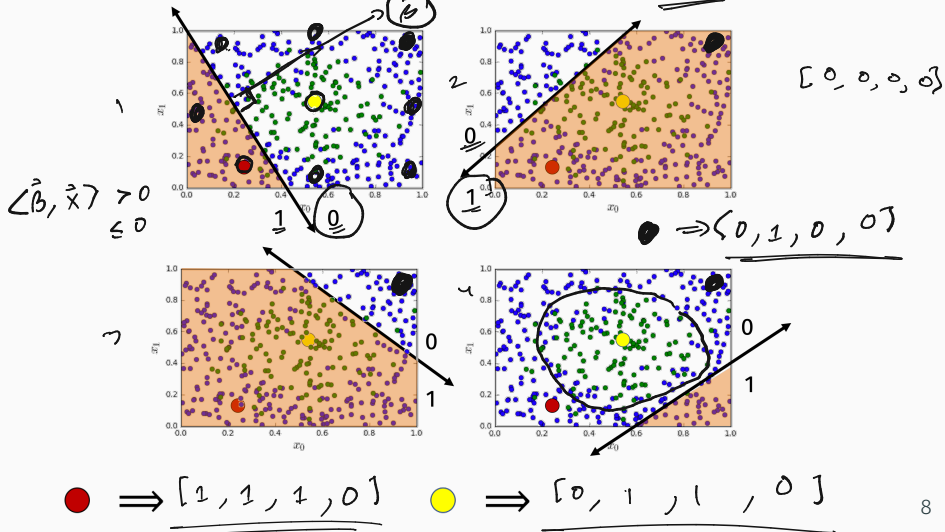


Could use feature transformations or a non-linear kernel.

Alternative approach: Divide the space up into regions using multiple linear classifiers.

SIMPLE MOTIVATING EXAMPLE

For each linear classifier $\vec{\beta}$ add a new 0,1 feature for every example $\vec{x} = [x_0, x_1]$ depending on the sign of $\langle \vec{x}, \vec{\beta} \rangle$.



SIMPLE MOTIVATING EXAMPLE

$$\begin{array}{c}
 \begin{bmatrix} .2, .8 \\ .5, .5 \\ \vdots \\ .5, 1 \end{bmatrix} = \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \end{bmatrix} \Rightarrow \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vdots \\ \vec{u}_n \end{bmatrix} = \begin{bmatrix} 0, 0, 1, 0 \\ 0, 1, 1, 0 \\ \vdots \\ 0, 0, 0, 0 \end{bmatrix}
 \end{array}$$

$N_H = 4$

Question: After data transformation, how should we map a new vectors \vec{u} to a class label?

$$\begin{array}{c}
 \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
 \end{array}$$

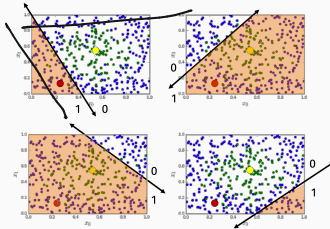
$$\begin{array}{c}
 \begin{bmatrix} 0, 0, 1, 0 \\ 0, 1, 1, 0 \\ \vdots \\ 0, 0, 0, 0 \end{bmatrix} \xrightarrow{?} \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}
 \end{array}$$

if $\langle \vec{u}_i, \vec{b} \rangle > 0$ output 1 learning
 ≤ 0 output 0

SIMPLE MOTIVATING EXAMPLE

Our machine learning algorithms needs to **learn two things**:

- The original linear functions which divide our data set into regions (their slopes + intercepts).



- Another linear function which maps our new features to an output label (typically by thresholding).

Solved by 2-layer neural net.

POSSIBLE MODEL

Input: $\vec{x} = [x_1, \dots, x_{N_I}]$ N_I = "number of input features"

Model: $f(\vec{x}; \Theta)$

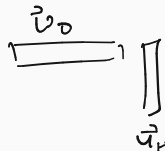
O = "output"

H = "hidden"

N_H = "number of hidden features"

feature transform $\left\{ \begin{array}{l} \cdot \vec{z}_H \in \mathbb{R}^{N_H} = \underline{W_H} \vec{x} + \vec{b}_H \rightarrow b_{H0} \\ \cdot \underline{\vec{u}}_H = [\vec{z}_H > 0] \end{array} \right.$

final linear classifier $\left\{ \begin{array}{l} \cdot z_O \in \mathbb{R} = \underline{W_O} \underline{\vec{u}}_H + b_O \\ \cdot \underline{u_O} = [z_O > 0] \end{array} \right.$



Parameters: $\Theta = [\underline{W_H} \in \mathbb{R}^{N_H \times N_I}, \underline{\vec{b}}_H \in \mathbb{R}^{N_H}, \underline{W_O} \in \mathbb{R}^{1 \times N_H}, \underline{b_O} \in \mathbb{R}]$.

W_H, W_O are weight matrices and \vec{b}_H, b_O are bias terms that account for the intercepts of our linear functions.

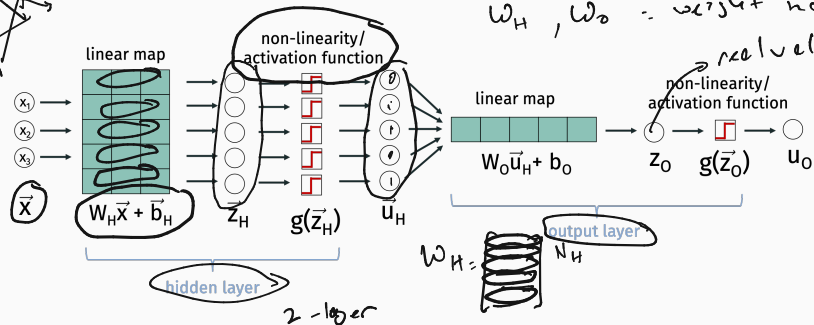
$$\left[\begin{array}{c|c|c|c|c|c|c|c} W_H & 1 & \vec{w}_H & \dots & 1 & W_O & 1 & b_O \end{array} \right]$$

POSSIBLE MODEL



Our model is function f which makes \vec{x} to a class label u_0 .¹

W_H, W_0 : weight matrices



This is called a “multilayer perceptron”: one of the oldest types of neural nets. Dates back to Frank Rosenblatt from 1958

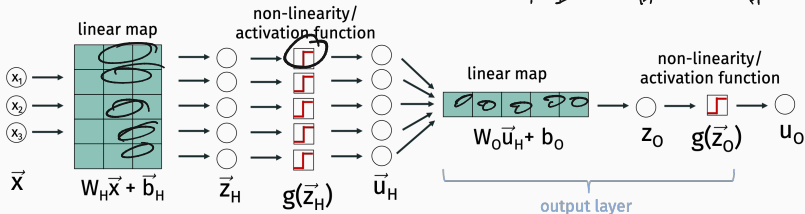
- Number of input variables $N_I = 3$ (# columns in W_H)
- Number of hidden variables $N_H = 5$ (# rows in U_H)
- Number of ~~hidden~~ variables $N_O = 1$

¹For regression, would cut off at z_0 to get continuous output.

POSSIBLE MODEL

Our model is function f which maps \vec{x} to a class label u_0 .

$$N_I \cdot N_H + N_H \cdot N_O$$



Training the model:

cross-entropy

gradient descent

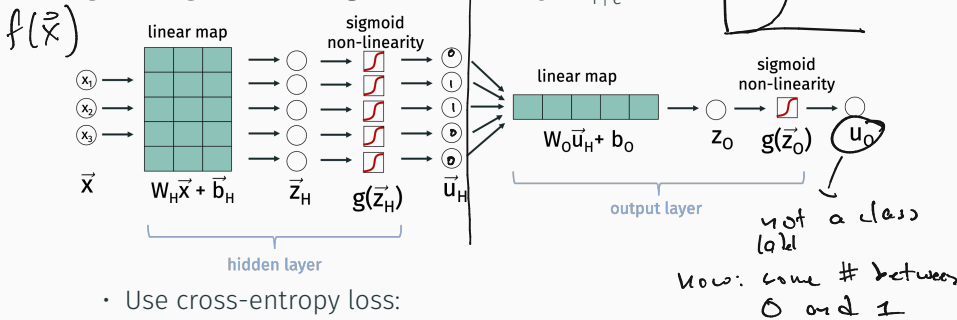
- Choose a loss function $L(f(\vec{x}, \Theta), y)$.
- Find optimal parameters: $\Theta^* = \arg \min_{\Theta} \sum_{i=1}^n L(f(\vec{x}_i, \Theta), y_i)$

How to find optimal parameters?

FINAL MODEL

A more typical model uses smoother activation functions, aka non-linearities, which are more amenable to computing gradients.

E.g. we might use the **sigmoid function** $g = \frac{1}{1+e^{-x}}$.



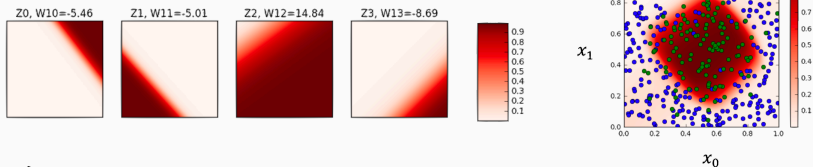
- Use cross-entropy loss:

$$L(f(\vec{x}_i, \Theta), y) = -y \log(f(\vec{x}_i, \Theta)) - (1 - y) \log(1 - f(\vec{x}_i, \Theta))$$

- We will discuss later exactly how to compute gradients.
- Will also discuss categorical cross-entropy/softmax loss for multi-class problems.

FINAL MODEL

Intuitively switching to a sigmoid activation is not that different from using the step function.²

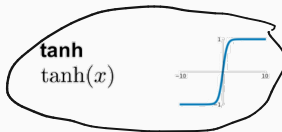
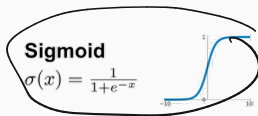


$$\vec{X} = [x_0, x_1] \rightarrow \{0, 1, 1, 0\}$$
$$\rightarrow [0.01, .9, .95, .06]$$

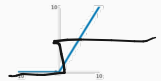
²Sometimes called the “Heaviside step function” in machine learning.

Things we can change in this basic classification network:

- More or less hidden variables.
- We could add more layers.
- Different non-linearity/activation function.
- Different loss function (more on that next class).



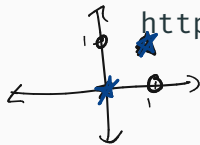
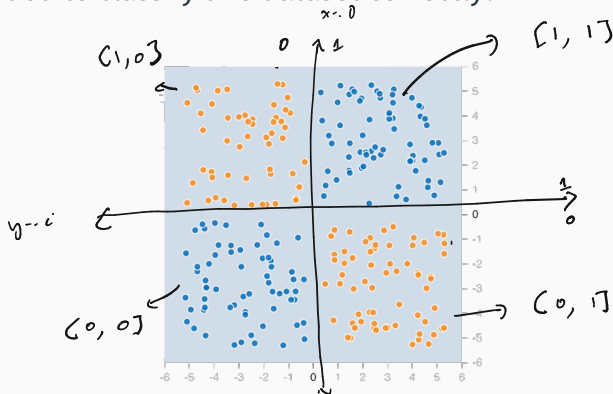
ReLU
 $\max(0, x)$



Rectified Linear Unit.

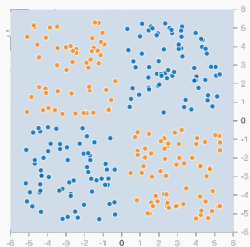
TEST YOUR INTUITION

How many hidden variables (e.g. splitting hyperplanes) would be needed to classify this dataset correctly?

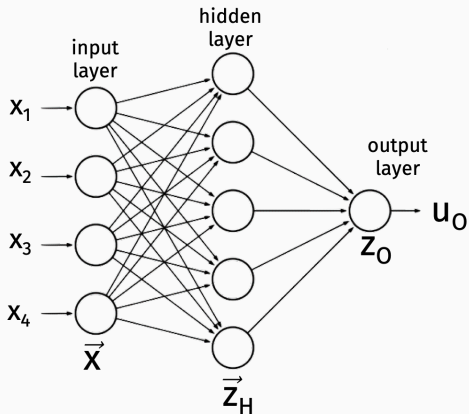


<https://playground.tensorflow.org/>

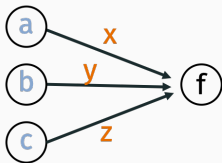
TEST YOUR INTUITION



Another common diagram for a 2-layered network:

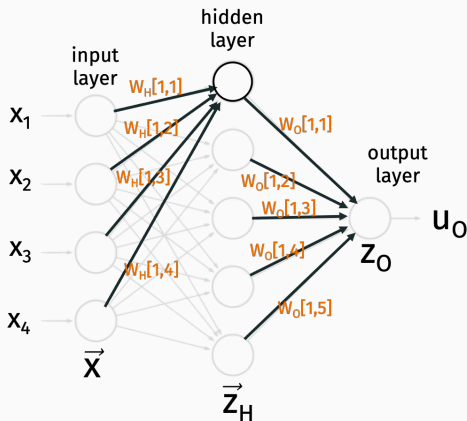


Neural network math:



$$f = ax + by + cz$$

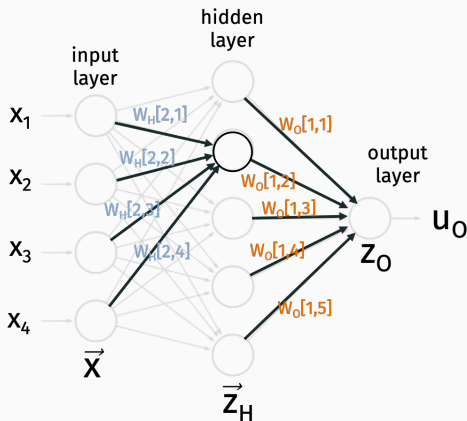
How to interpret:



\mathbf{W}_H and \mathbf{W}_O are our weight matrices from before.

Note: This diagram does not explicitly show the bias terms or the non-linearity.

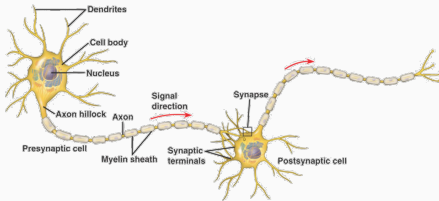
How to interpret:



W_H and W_O are our weight matrices from before.

Note: This diagram depicts a network with “fully-connected” layers.

Simplified model of the brain:

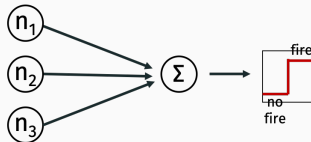


Dendrites: Input electrical current from other neurons.

Axon: Output electrical current to other neurons.

Synapse: Where these two connect.

A neuron “fires” (outputs a non-zero electric charge) if it receives enough cumulative electrical input from the neurons connected to it.



Output charge can be positive or negative (excitatory vs. inhibitory).

Inspired early work on neural networks:

- 1940s Donald Hebb proposed a Hebbian learning rule for how brains neurons change over time to allow learning.
- 1950s Frank Rosenblatt's Perceptron is one of the first "artificial" neural networks.
- Continued work throughout the 1960s.

Main issue with neural network methods: They are hard to train. Generally require a lot of computation power. Also pretty finicky: user needs to be careful with initialization, regularization, etc. when training. Often requires a lot of experimentation to get right.

EARLY NEURAL NETWORK EXPLOSION

Around 1985 a few groups (re)-discovered the **backpropagation algorithm** which allows for efficient training of neural nets via **gradient descent**. Along with increased computational power this lead to a resurgence of interest in neural network models.

Backpropagation Applied to Handwritten Zip Code Recognition

Y. LeCun
B. Boser
J. S. Denker
D. Henderson
R. E. Howard
W. Hubbard
L. D. Jackel

AT&T Bell Laboratories Holmdel, NJ 07733 USA

The ability of learning networks to generalize can be greatly enhanced by providing constraints from the task domain. This paper demonstrates how such constraints can be integrated into a backpropagation network through the architecture of the network. This approach has been successfully applied to the recognition of handwritten zip code digits provided by the U.S. Postal Service. A single network learns the entire recognition operation, going from the normalized image of the character to the final classification.

Good performance on problems like digit recognition.

In the 1990s and early 2000s, kernel methods, SVMs, and probabilistic methods began to dominate the literature in machine learning:

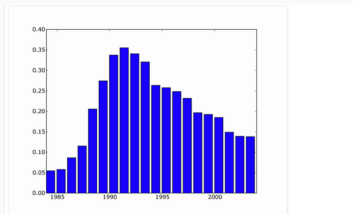
- Work well “out of the box”.
- Relatively easy to understand theoretically.
- Not too computationally expensive for moderately sized datasets.

Fun blog post to check out from 2005:

<http://yaroslavvb.blogspot.com/2005/12/trends-in-machine-learning-according.html>

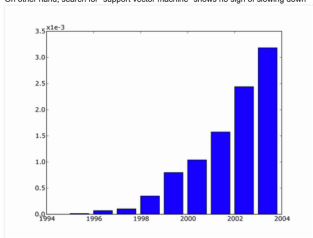
NEURAL NETWORK DECLINE

Finding trends in machine learning by search papers in Google Scholar that match a certain keyword:



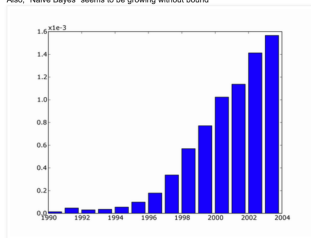
You can see a major upward trend starting around 1985 (that's when Yann LeCun and several others independently rediscovered backpropagation algorithm), peaking in 1992, and going downwards from then.

On other hand, search for "support vector machine" shows no sign of slowing down



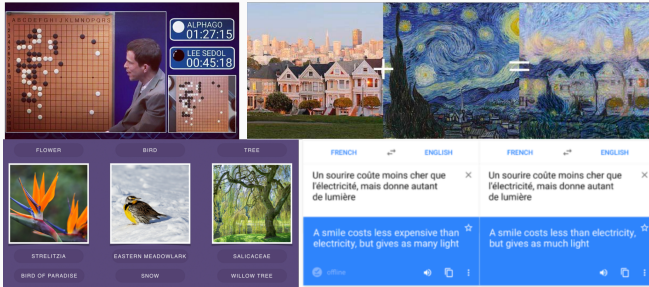
(1995 is when Vapnik and Cortez proposed the algorithm)

Also, "Naive Bayes" seems to be growing without bound




If I were to trust this, I would say that Naive Bayes research the hottest machine learning area right now

MODERN NEURAL NETWORKS



Recent state-of-the-art results in game playing, image recognition, content generation, natural language processing, machine translation, many other areas.

All changed with the introduction of AlexNet and the 2012 ImageNet Challenge...




[Explore](#) [Download](#) [Challenges](#) [Publications](#) [Updates](#) [About](#)

14,197,122 Images, 21841 synsets indexed

Not logged in. [Login](#) | [Signup](#)

ImageNet is an image database organized according to a hierarchical structure (currently only the nouns), in which each node of the hierarchy is depicted by hundreds of images. Currently we have an average of over five hundred images per node. We hope that ImageNet will become a useful resource for researchers, educators, students and all of you who share an interest in pictures.

[Click here](#) to learn more about ImageNet, [Click here](#) to join the ImageNet mailing list.



What do these images have in common? *Find out!*

All changed with the introduction of AlexNet and the 2012 ImageNet Challenge...

team name	team members	filename	flat cost	hie cost	description
NEC-UIUC	NEC: Yuanqing Lin, Fengjun Lv, Shenghuo Zhu, Ming Yang, Timothee Cour, Kai Yu UIUC: LiangLiang Cao, Zhen Li, Min-Hsuan Tsai, Xi Zhou, Thomas Huang Rutgers: Tong Zhang	flat_opt.txt	0.28191	2.1144	using sift and lbp feature with two non-linear coding representations and stochastic SVM , optimized for top-5 hit rate

2010 Results

Team name	Filename	Error (5 guesses)	Description
SuperVision	test-preds-141-146.2009-131- 137-145-146.2011-145f.	0.15315	Using extra training data from ImageNet Fall 2011 release
SuperVision	test-preds-131-137-145-135- 145f.txt	0.16422	Using only supplied training data
ISI	pred_FVs_wLACs_weighted.txt	0.26172	Weighted sum of scores from each classifier with SIFT+FV, LBP+FV, GIST+FV, and CSIFT+FV, respectively.

2012 Results

Why 2012?

- Clever ideas in changing neural network architectures.
- Wide-spread access to GPU computing power (CUDA and publicly available Nvidia GPU first released in 2007).

2019 TURING AWARD WINNERS

“For conceptual and engineering breakthroughs that have made deep neural networks a critical component of computing.”



Yann LeCun



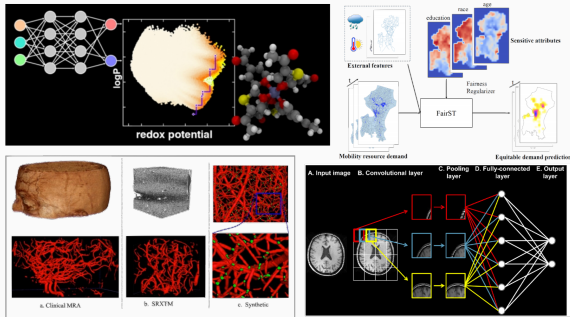
Geoff Hinton



Yoshua Bengio

USE OUTSIDE OF COMPUTER SCIENCE

Neural networks are flexible and relatively easy to understand conceptually, so they are also having impact in application areas outside of computer science.



Researchers working in the medicine, natural sciences, engineering, etc. are having a lot of luck implementing and applying these models to their data problems.