Introduction to Machine Learning

UNIVERSITY OF OREGON

Lecture 1


https://www.techemergence.com/what-is-machine-learning/

CTEQ Summer School 2019
University of Pittsburgh
July 2019

Bryan Ostdiek

What is Machine Learning for?

Getting information from data

Labeled Data

Unlabeled Data

Supervised Learning

- Classification
- Numerical predictions
- Etc

Unsupervised Learning

- Clustering
- Anomaly detection
- GANS
- Etc

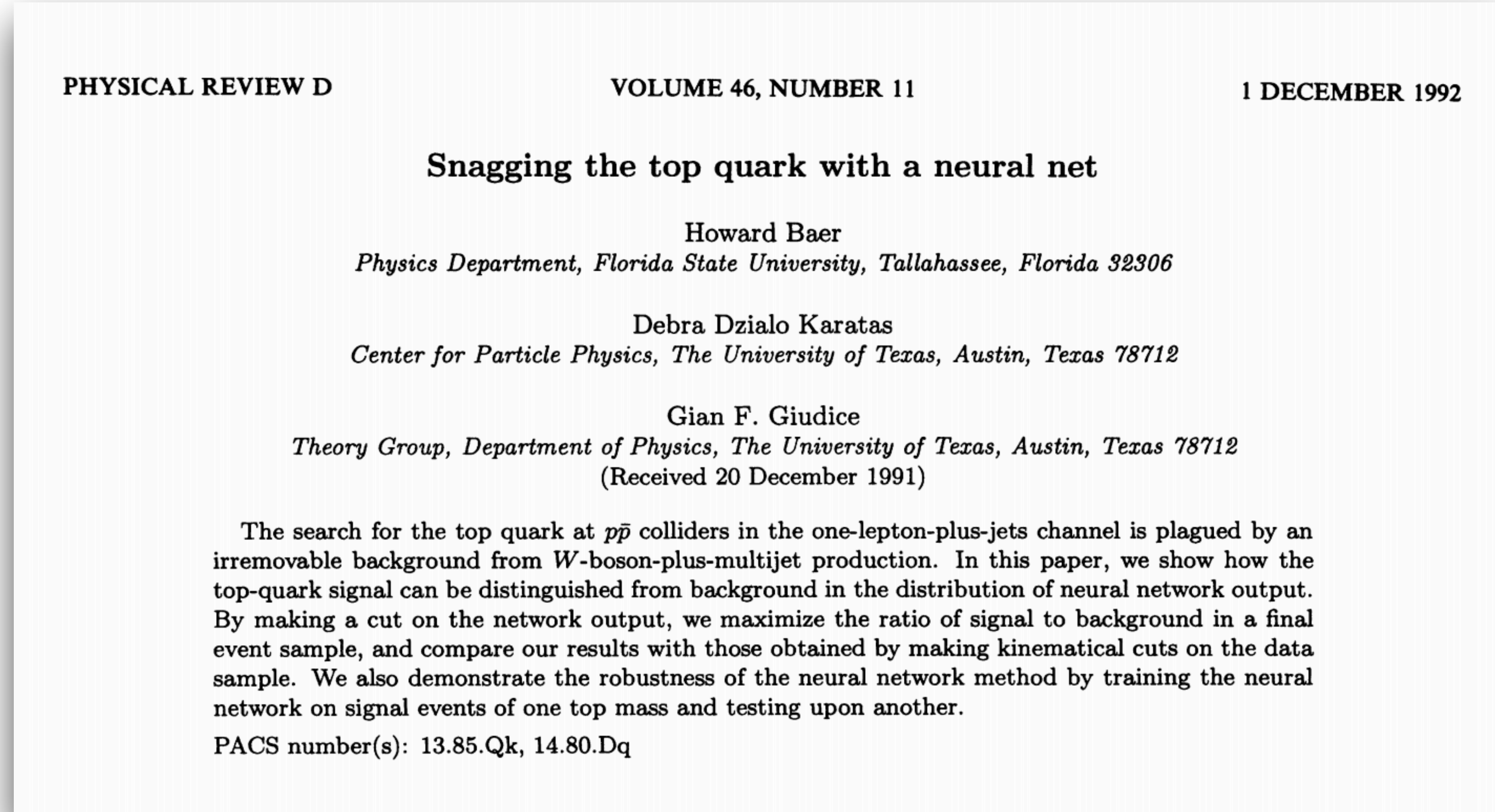
Hybrid

- Weak supervision
- Classification without labels (CWoLa)

Machine learning in High Energy Physics

Great collection of HEP machine learning resources
https://github.com/iml-wg/HEP-ML-Resources

Machine learning in High Energy Physics



PHYSICAL REVIEW D VOLUME 46, NUMBER 11 1 DECEMBER 1992

Snagging the top quark with a neural net

Howard Baer
Physics Department, Florida State University, Tallahassee, Florida 32306

Debra Dzialo Karatas
Center for Particle Physics, The University of Texas, Austin, Texas 78712

Gian F. Giudice
Theory Group, Department of Physics, The University of Texas, Austin, Texas 78712

(Received 20 December 1991)

The search for the top quark at $p\bar{p}$ colliders in the one-lepton-plus-jets channel is plagued by an irremovable background from W-boson-plus-multijet production. In this paper, we show how the top-quark signal can be distinguished from background in the distribution of neural network output. By making a cut on the network output, we maximize the ratio of signal to background in a final event sample, and compare our results with those obtained by making kinematical cuts on the data sample. We also demonstrate the robustness of the neural network method by training the neural network on signal events of one top mass and testing upon another.

PACS number(s): 13.85.Qk, 14.80.Dq

Great collection of HEP machine learning resources
https://github.com/iml-wg/HEP-ML-Resources

Machine learning in High Energy Physics

PHYSICAL REVIEW D VOLUME 46, NUMBER 11 1 DECEMBER 1992

Snagging the top quark with a neural net

Howard Baer
Physics Department, Florida State University, Tallahassee, Florida 32306

Debra Dzialo Karatas
Center for Particle Physics, The University of Texas, Austin, Texas 78712

arXiv.org > hep-ex > arXiv:0707.1712

Search...
Help | Advan

High Energy Physics – Experiment

B-Tagging at CDF and DO, Lessons for LHC

T. Wright, for the CDF, DØ Collaborations

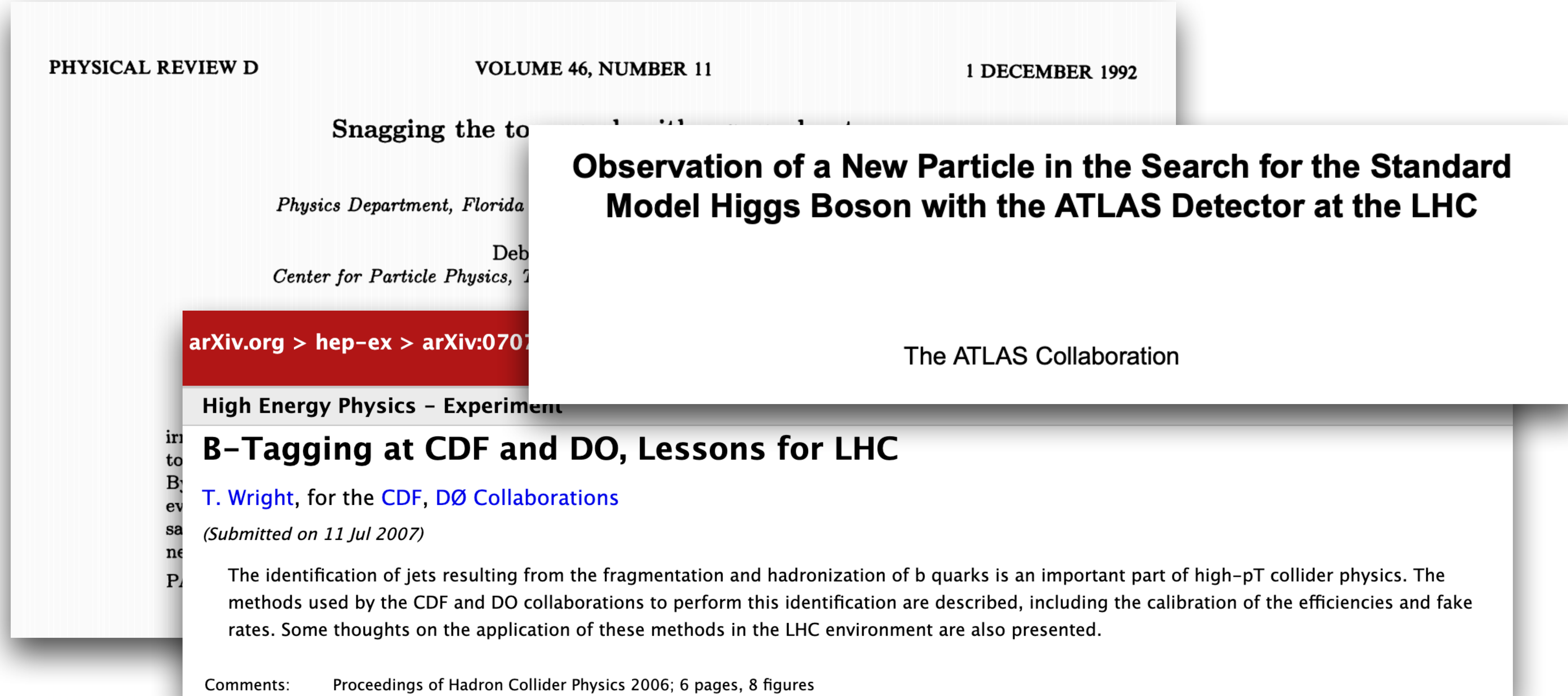
(Submitted on 11 Jul 2007)

The identification of jets resulting from the fragmentation and hadronization of b quarks is an important part of high-pT collider physics. The methods used by the CDF and DO collaborations to perform this identification are described, including the calibration of the efficiencies and fake rates. Some thoughts on the application of these methods in the LHC environment are also presented.

Comments: Proceedings of Hadron Collider Physics 2006; 6 pages, 8 figures

Great collection of HEP machine learning resources
https://github.com/iml-wg/HEP-ML-Resources

Machine learning in High Energy Physics



PHYSICAL REVIEW D VOLUME 46, NUMBER 11 1 DECEMBER 1992

Snagging the to

Physics Department, Florida

Deb

Center for Particle Physics,

Observation of a New Particle in the Search for the Standard Model Higgs Boson with the ATLAS Detector at the LHC

The ATLAS Collaboration

arXiv.org > hep-ex > arXiv:070

High Energy Physics – Experiment

B-Tagging at CDF and DO, Lessons for LHC

T. Wright, for the CDF, DØ Collaborations

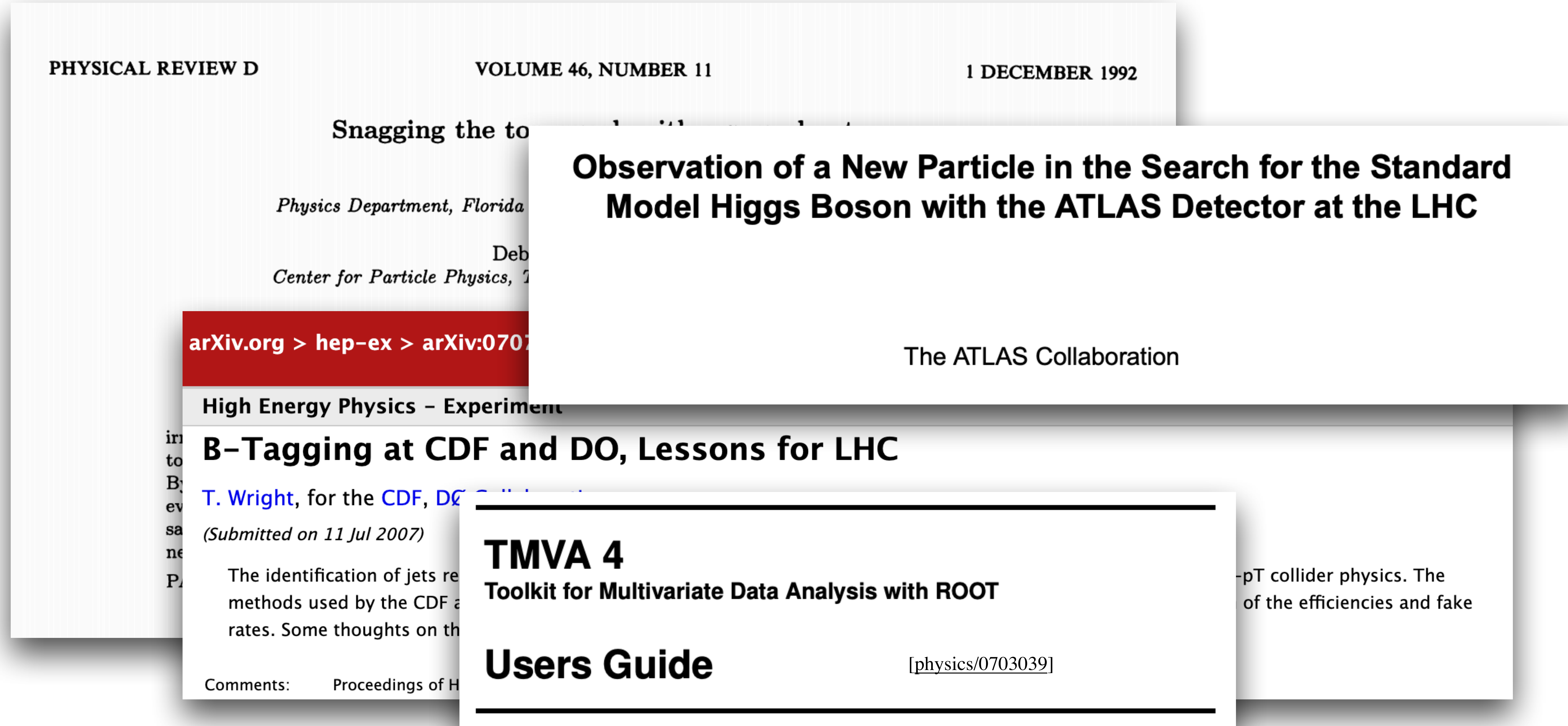
(Submitted on 11 Jul 2007)

The identification of jets resulting from the fragmentation and hadronization of b quarks is an important part of high-pT collider physics. The methods used by the CDF and DO collaborations to perform this identification are described, including the calibration of the efficiencies and fake rates. Some thoughts on the application of these methods in the LHC environment are also presented.

Comments: Proceedings of Hadron Collider Physics 2006; 6 pages, 8 figures

Great collection of HEP machine learning resources
https://github.com/iml-wg/HEP-ML-Resources

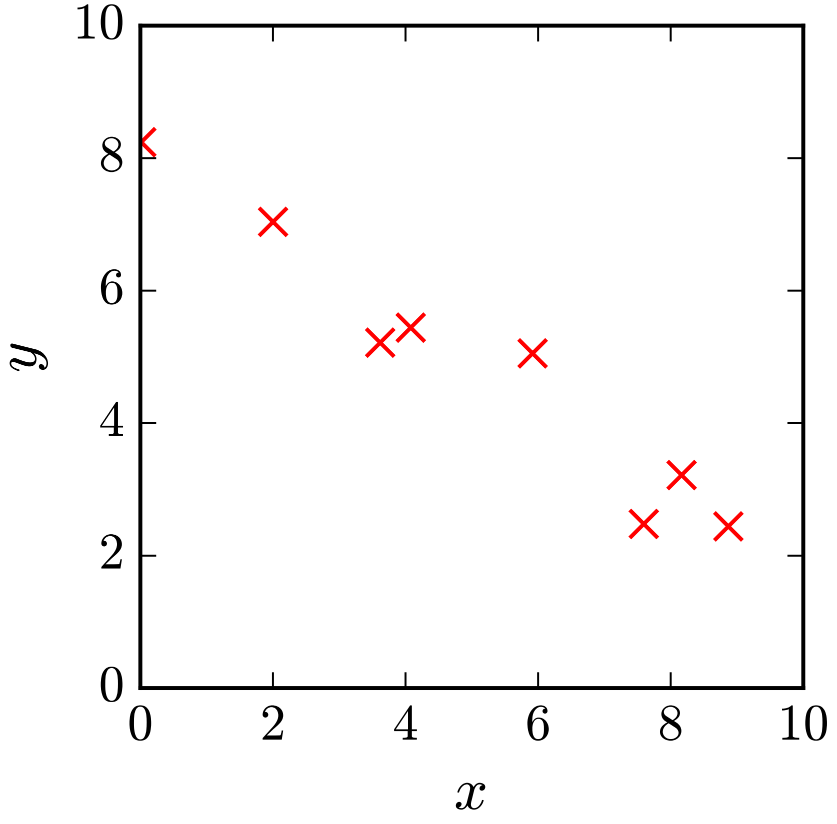
Machine learning in High Energy Physics



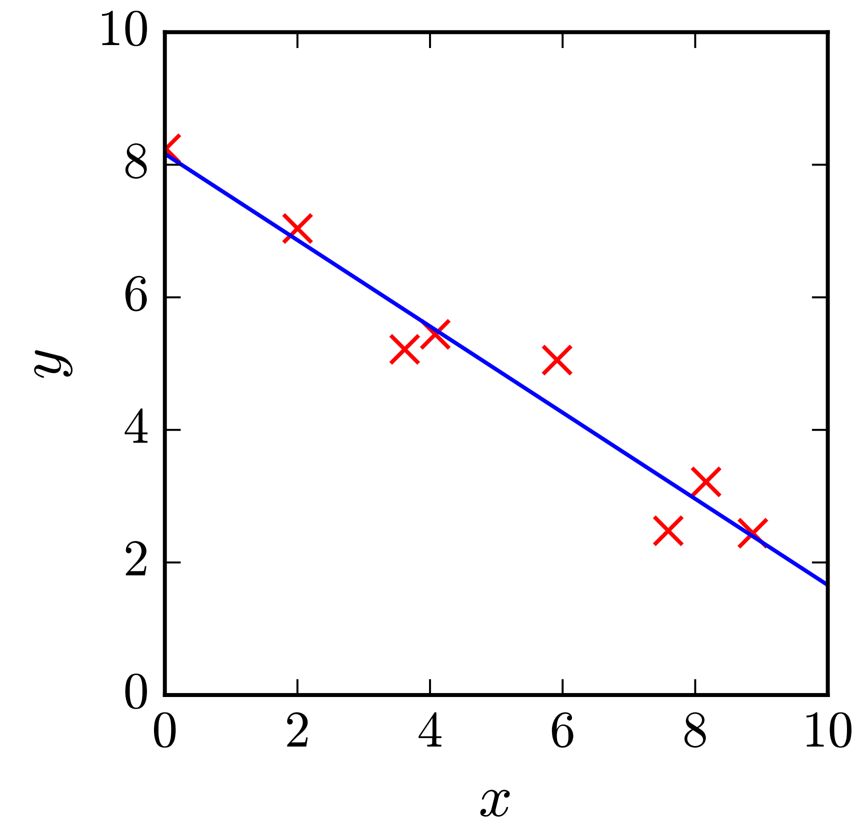
Object tagging, calibrations (systematic regression), event-level analysis

Great collection of HEP machine learning resources
https://github.com/iml-wg/HEP-ML-Resources

Review: Linear Regression



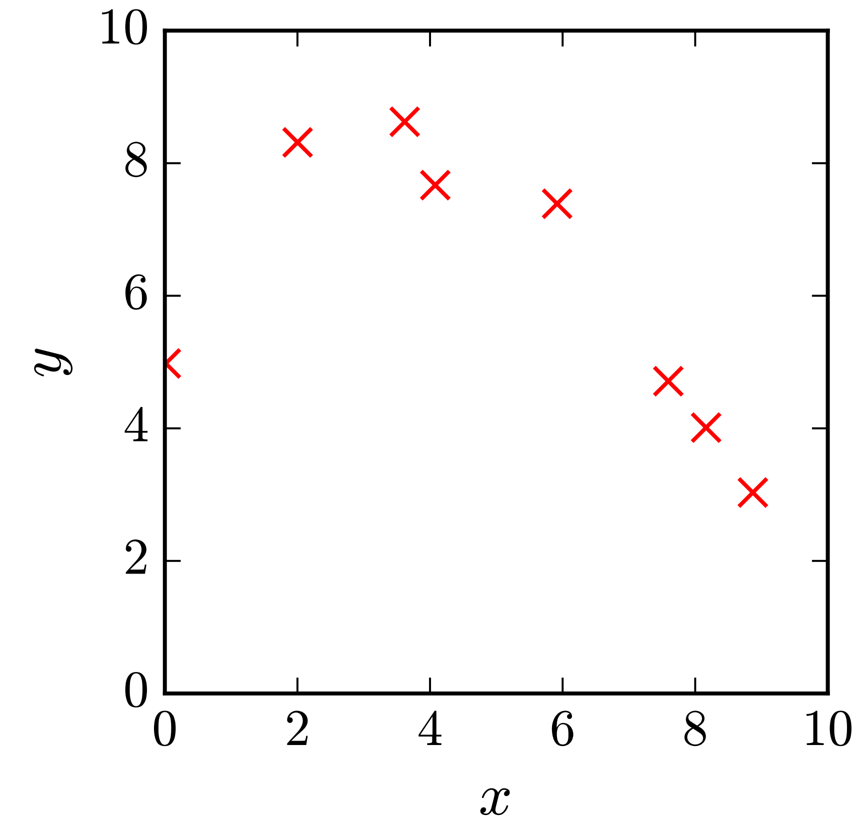
Review: Linear Regression



How to fit data

1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N} \sum_{i=1}^{N} \left(f(x_i, \vec{a}) - y_i\right)^2$
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a \text{ when } \left(\frac{\partial L(x, y, \vec{a})}{\partial \vec{a}} \Big|_{x,y} = 0 \right)$

Review: Linear Regression



How to fit data

1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x + a_2 x^2$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i, \vec{a}) - y_i)^2$
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a \text{ when } \left(\frac{\partial L(x, y, \vec{a})}{\partial \vec{a}} \bigg|_{x,y} = 0 \right)$

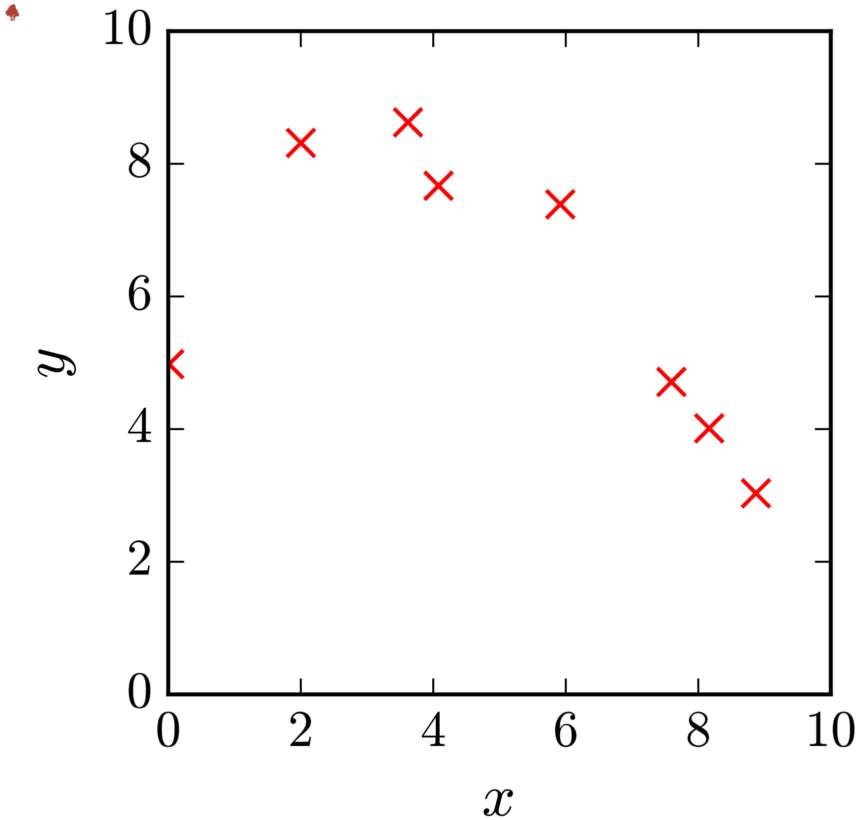
Review: ~~Linear~~ Regression

Quadratic?

<u>How to fit data</u>

1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x + a_2 x^2$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i, \vec{a}) - y_i)^2$

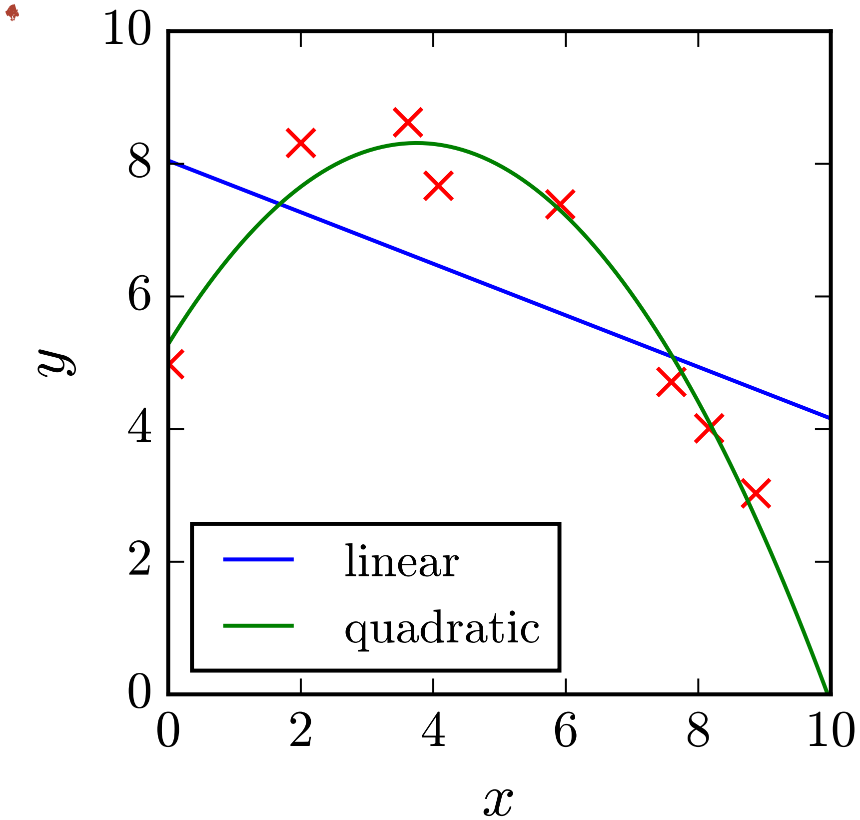
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a \text{ when } \left(\frac{\partial L(x, y, \vec{a})}{\partial \vec{a}} \bigg|_{x,y} = 0 \right)$



Review: ~~Linear~~ Regression

Quadratic?

<u>How to fit data</u>



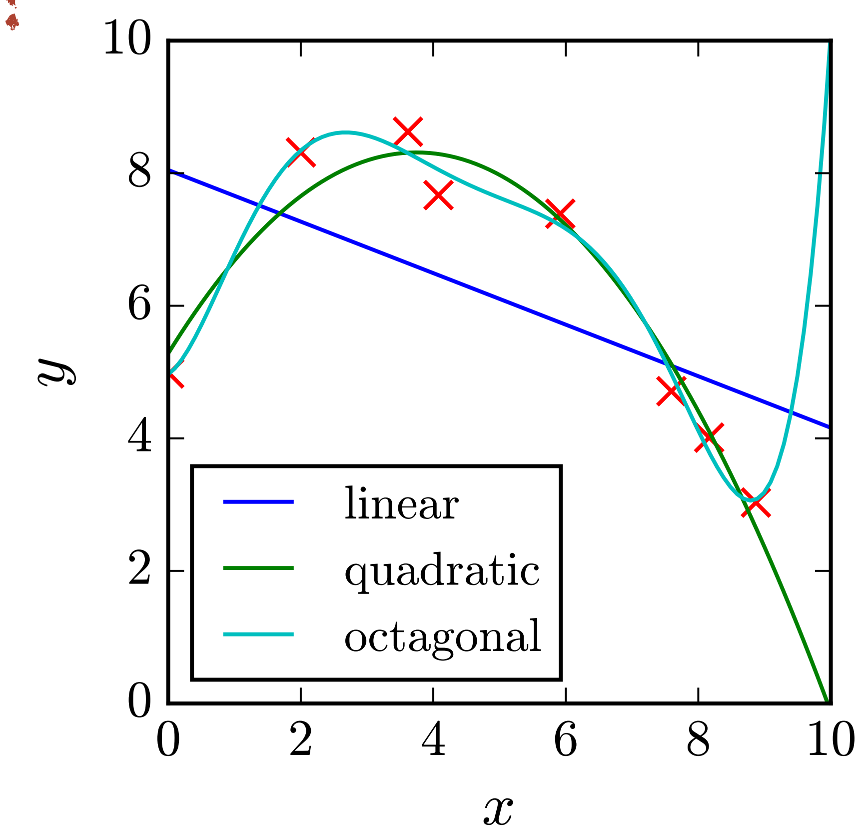
1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x + a_2 x^2$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i, \vec{a}) - y_i)^2$
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a \text{ when } \left(\left.\frac{\partial L(x, y, \vec{a})}{\partial \vec{a}}\right|_{x,y} = 0 \right)$

Review: ~~Linear~~ Regression

Quadratic?

<u>How to fit data</u>

1. Plot the data
2. Define the function
   - $f(x, \vec{a}) = a_0 + a_1 x + a_2 x^2$
3. Choose how to know what fits best
   - a.k.a. *Loss Function*
   - MSE: $L(x, y, \vec{a}) = \frac{1}{N} \sum_{i=1}^{N} (f(x_i, \vec{a}) - y_i)^2$
5. Find the minimum error (loss) (cost)
   - $a_{\text{best}} = a \text{ when } \left(\left. \frac{\partial L(x, y, \vec{a})}{\partial \vec{a}} \right|_{x,y} = 0 \right)$



Is that good enough?
How many parameters can we add?

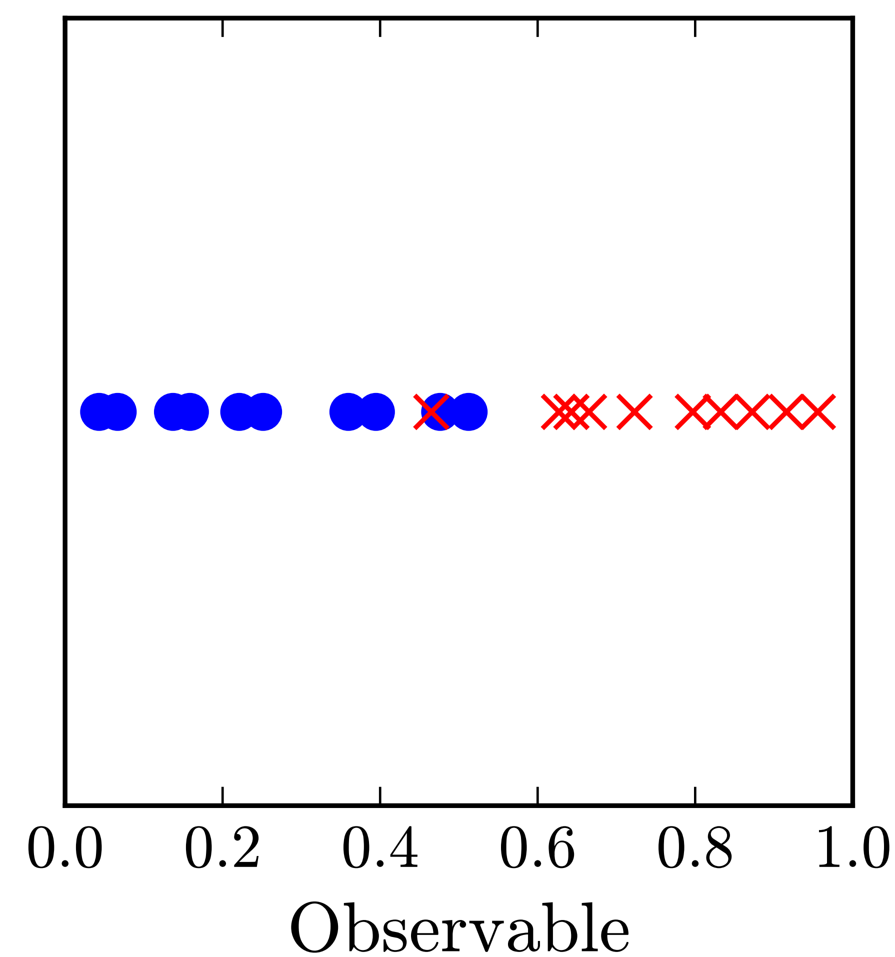
Logistic Regression

What if we want to predict a class, not a number?

Logistic Regression

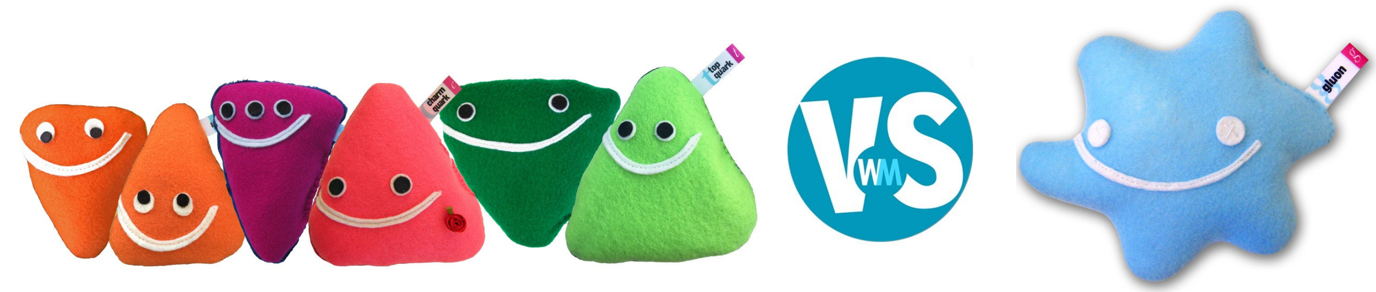
What if we want to predict a class, not a number?

What is the y-value we are trying to fit/predict?



Logistic Regression

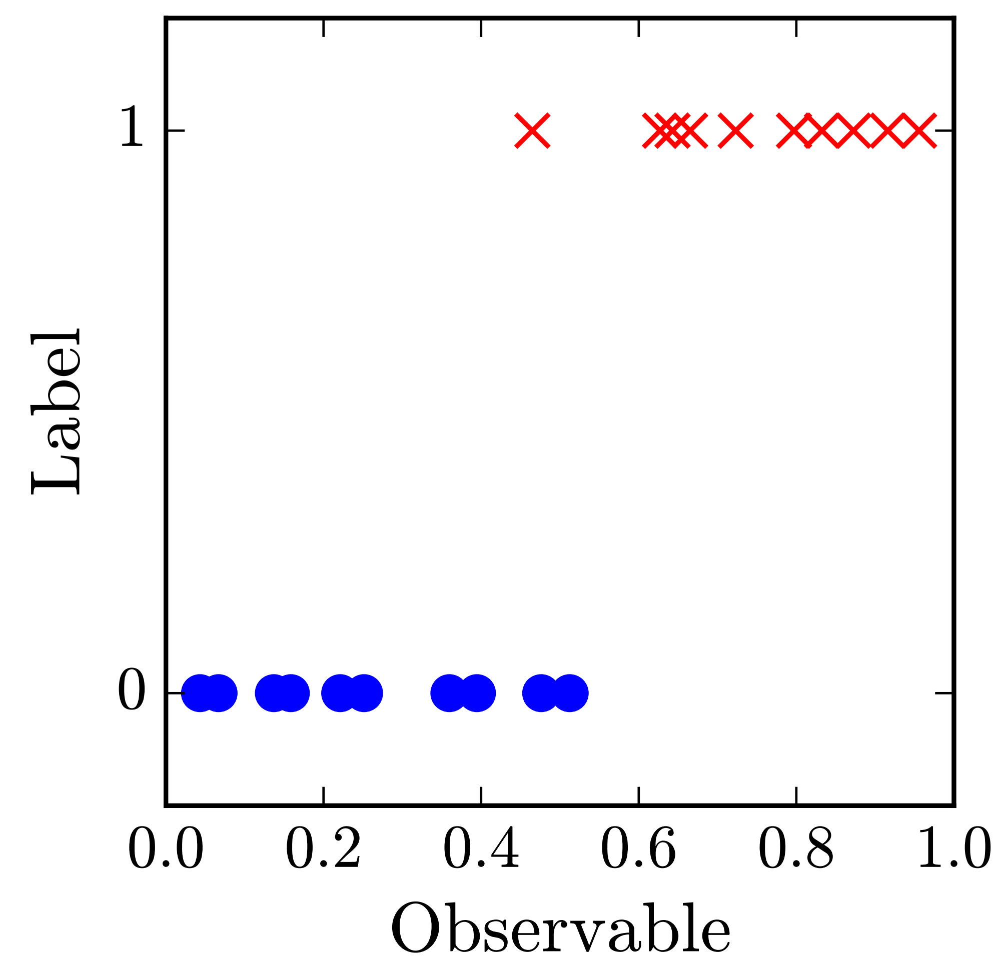
What if we want to predict a class, not a number?



What is the y-value we are trying to fit/predict?

Define one class as 1 (Signal)

Other class as 0 (Background)



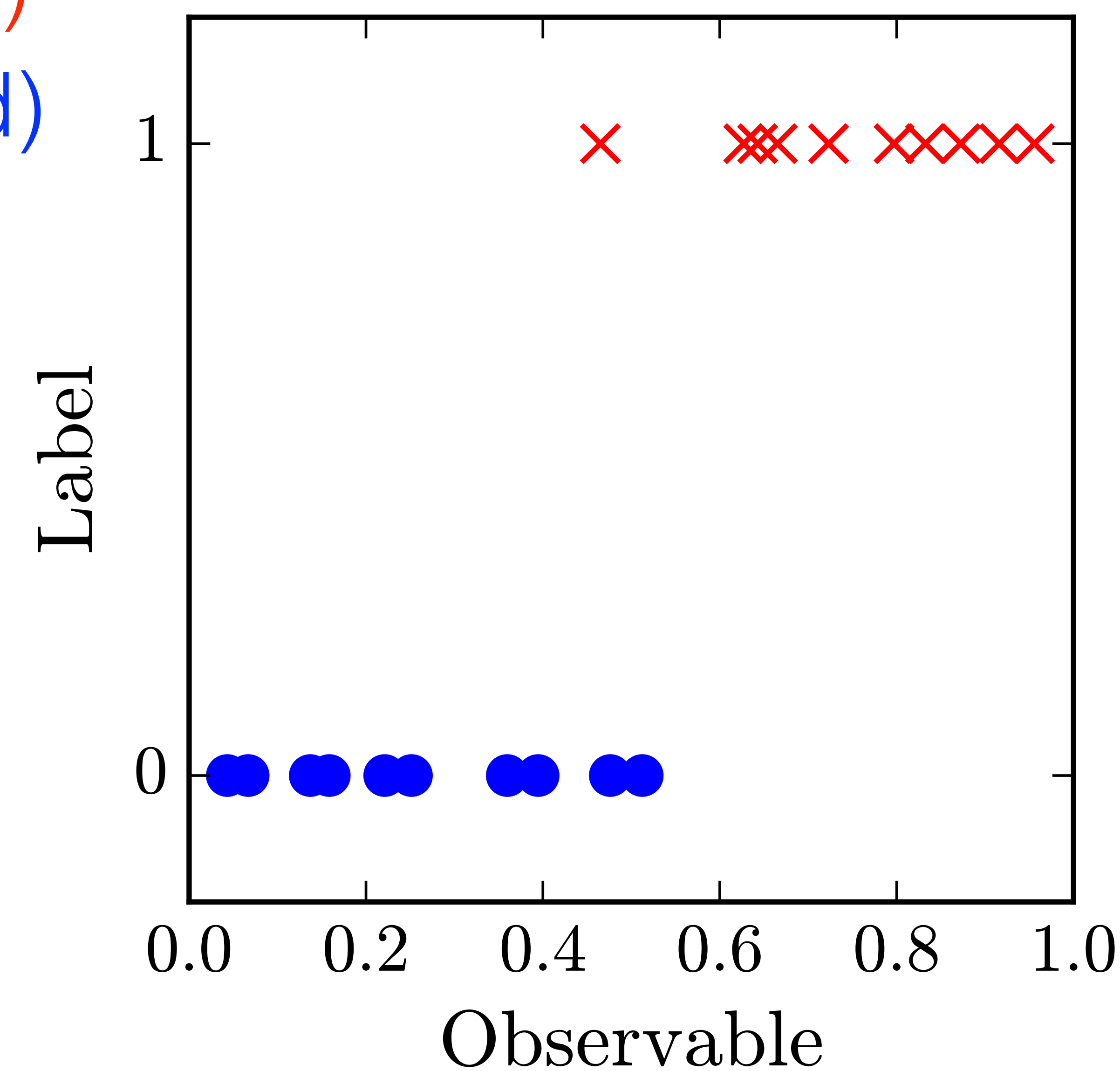
Logistic Regression

What if we want to predict a class, not a number?

What is the y-value we are trying to fit/predict?

Define one class as 1 (Signal)

Other class as 0 (Background)



Logistic Regression

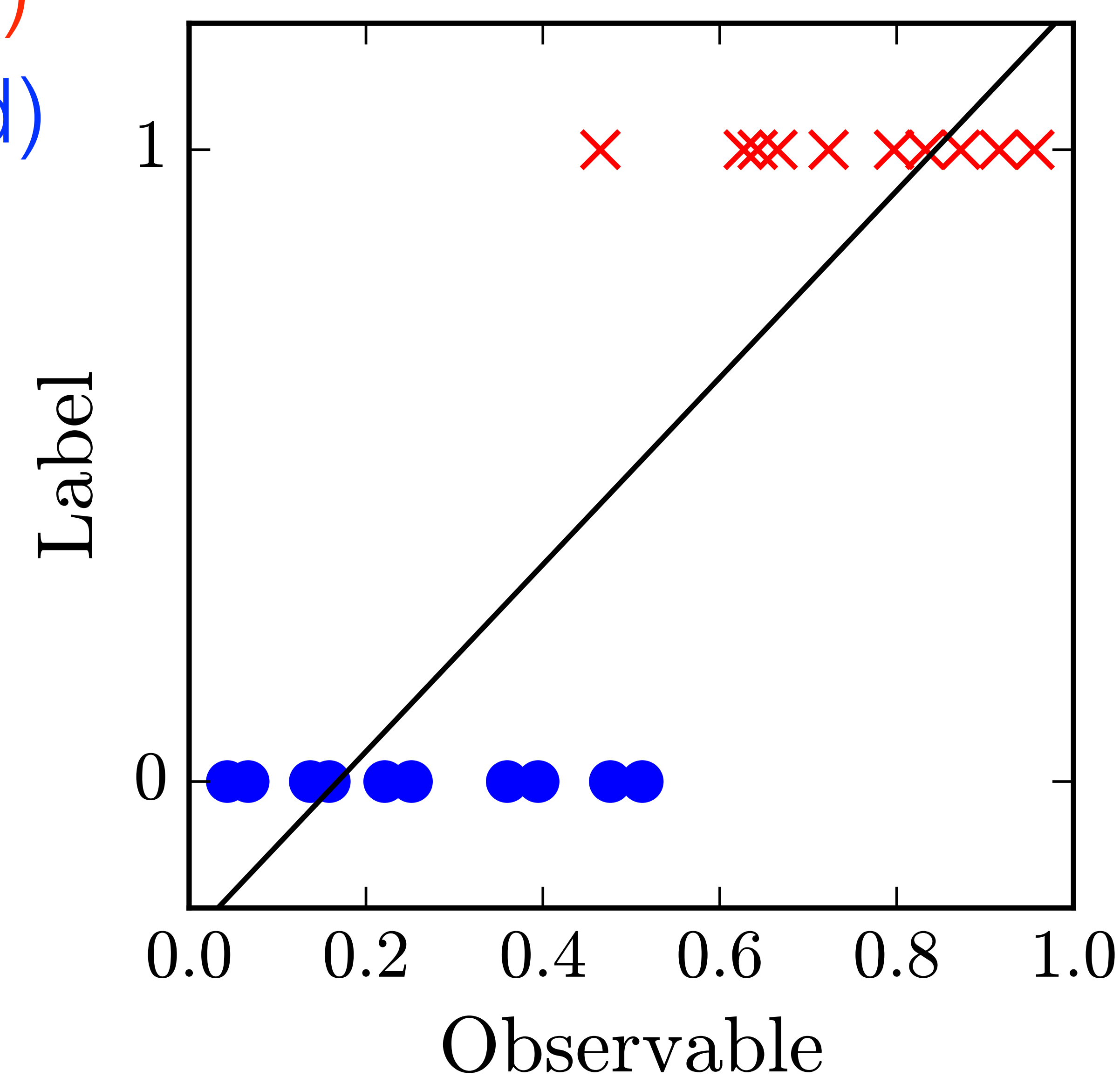
What if we want to predict a class, not a number?

What is the y-value we are trying to fit/predict?

Define one class as 1 (Signal)

Other class as 0 (Background)

- Linear Fit?



Logistic Regression

What if we want to predict a class, not a number?

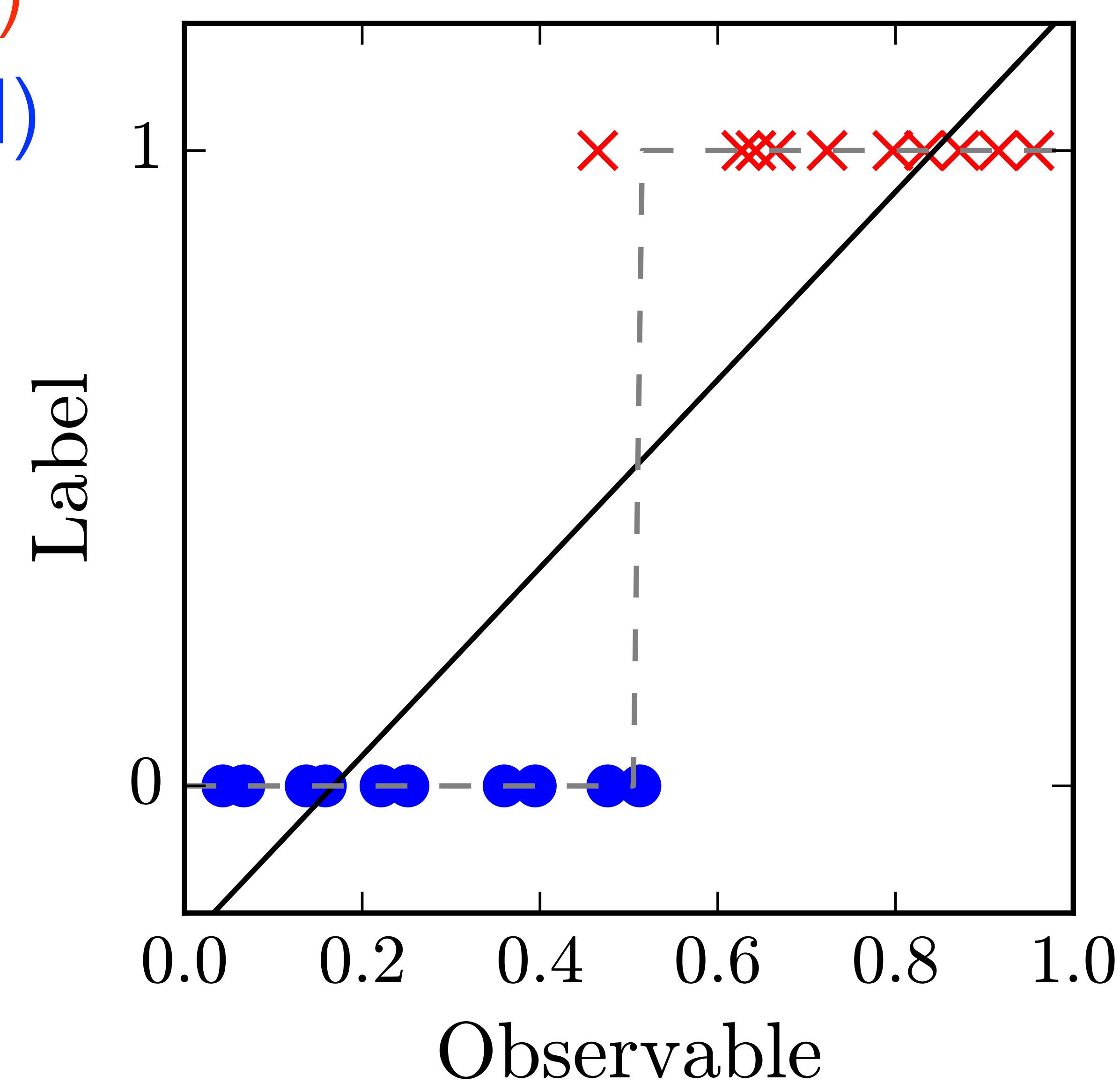
What is the y-value we are trying to fit/predict?

Define one class as 1 (Signal)

Other class as 0 (Background)

- Linear Fit?

- Round to nearest number?



Logistic Regression

What if we want to predict a class, not a number?

What is the y-value we are trying to fit/predict?

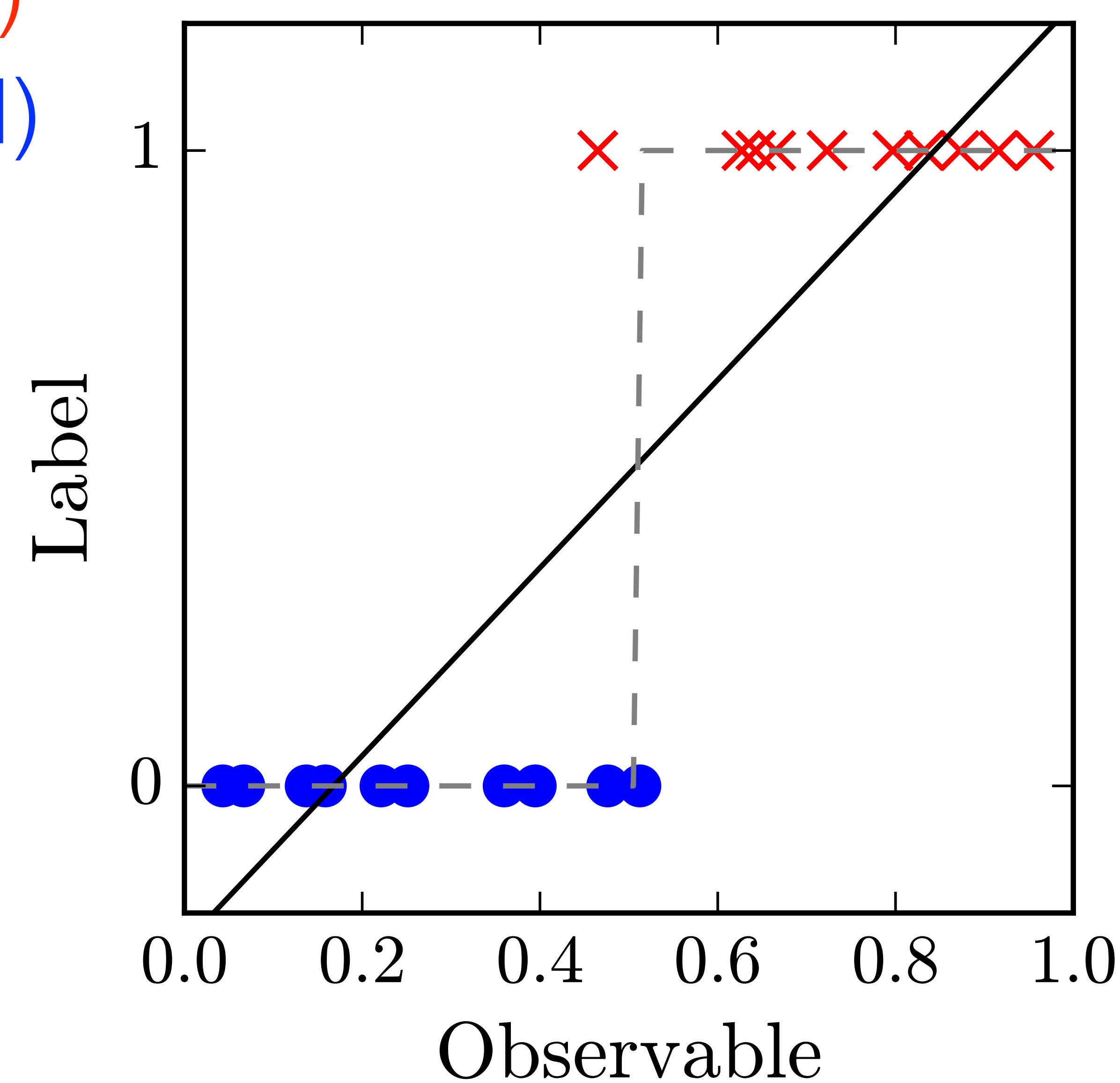
Define one class as 1 (Signal)

Other class as 0 (Background)

- Linear Fit?

- Round to nearest number?

- How does it generalize to more data?



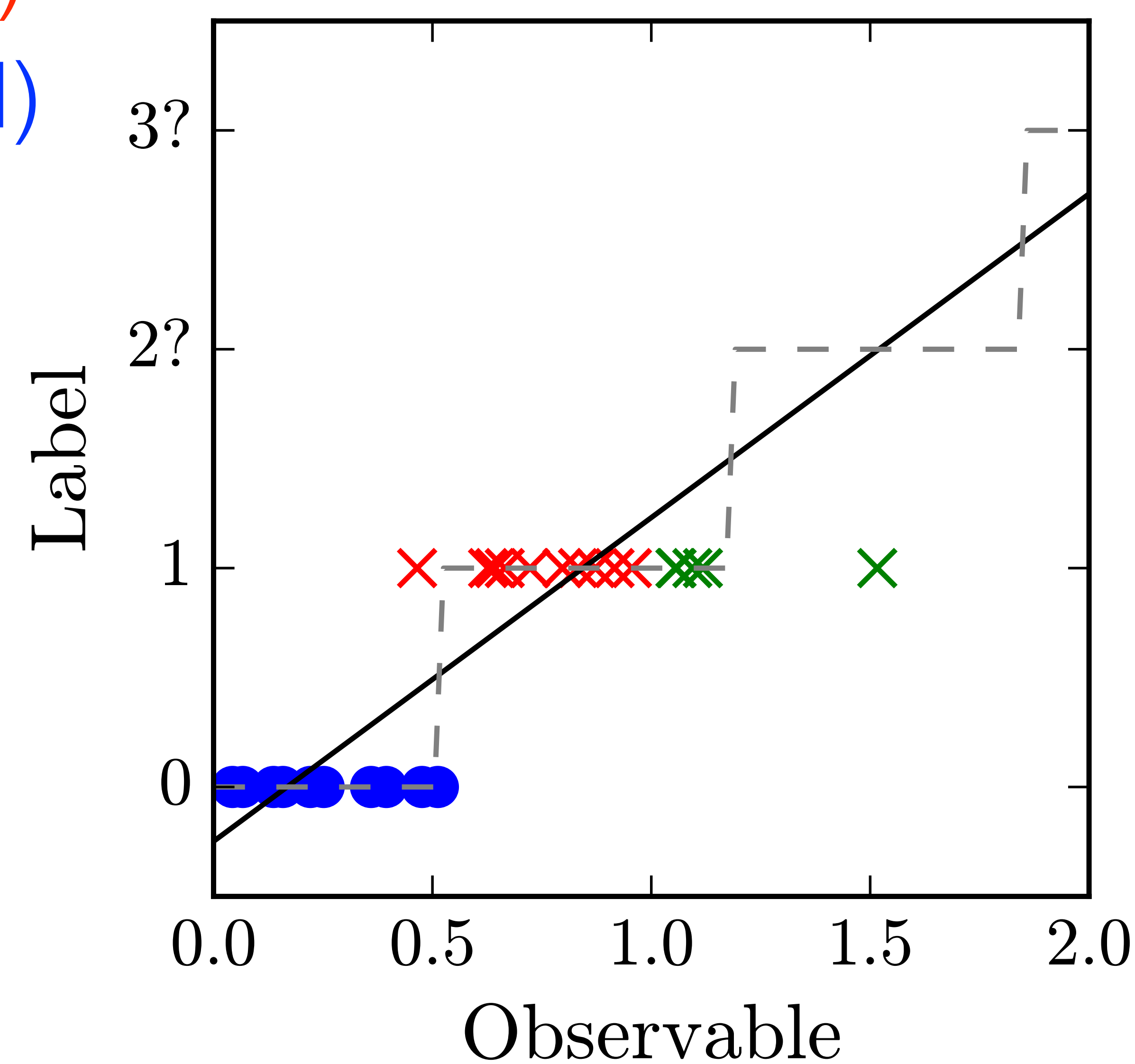
Logistic Regression

What if we want to predict a class, not a number?

What is the y-value we are trying to fit/predict?

Define one class as 1 (Signal)
Other class as 0 (Background)

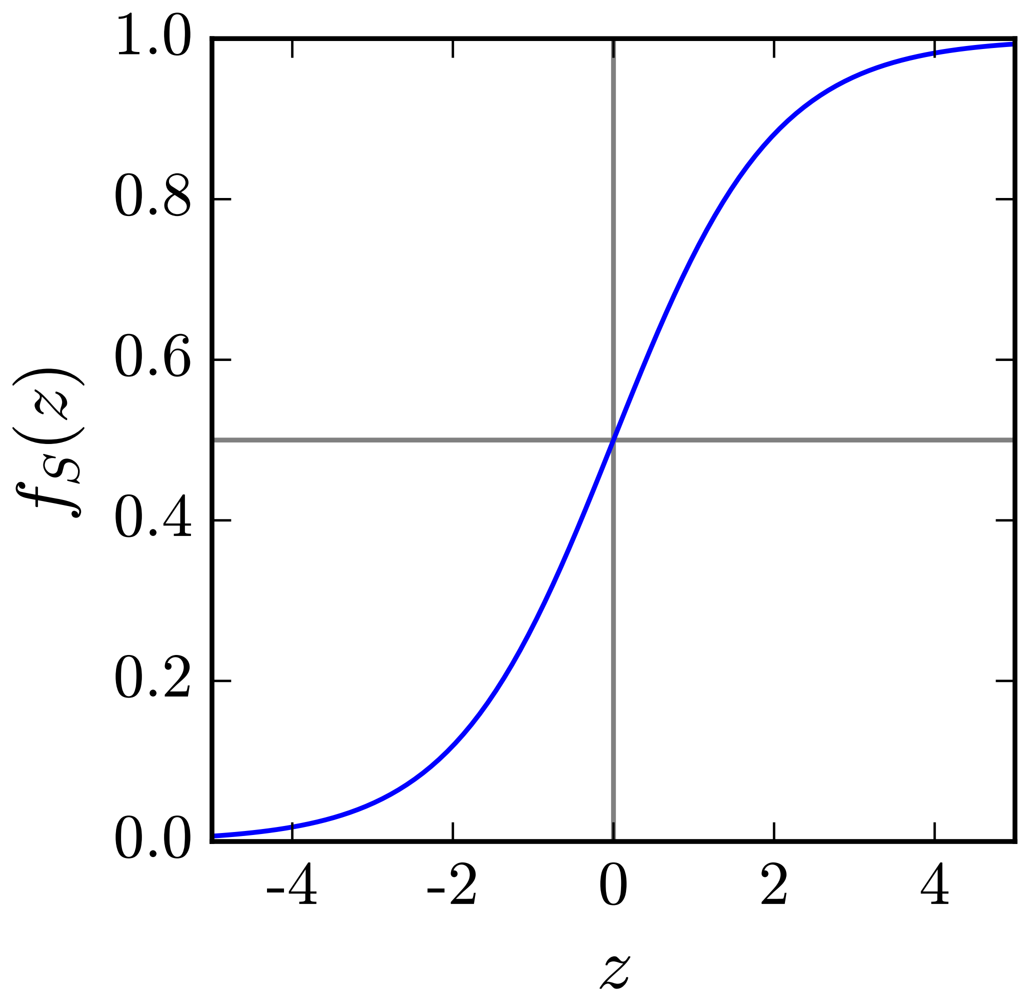
- Linear Fit?
- Round to nearest number?
- How does it generalize to more data?



Logistic Regression

What if we want to predict a class, not a number?

- Change the shape of function: Logistic/Sigmoid function



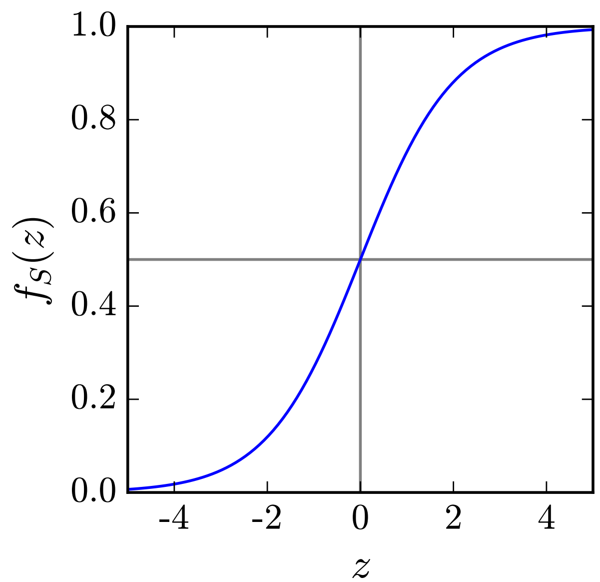
$$f_S(z) = \frac{1}{1 + \mathrm{e}^{-z}}$$

Does not add parameters

Logistic Regression

What if we want to predict a class, not a number?

- Change the shape of function: Logistic/Sigmoid function



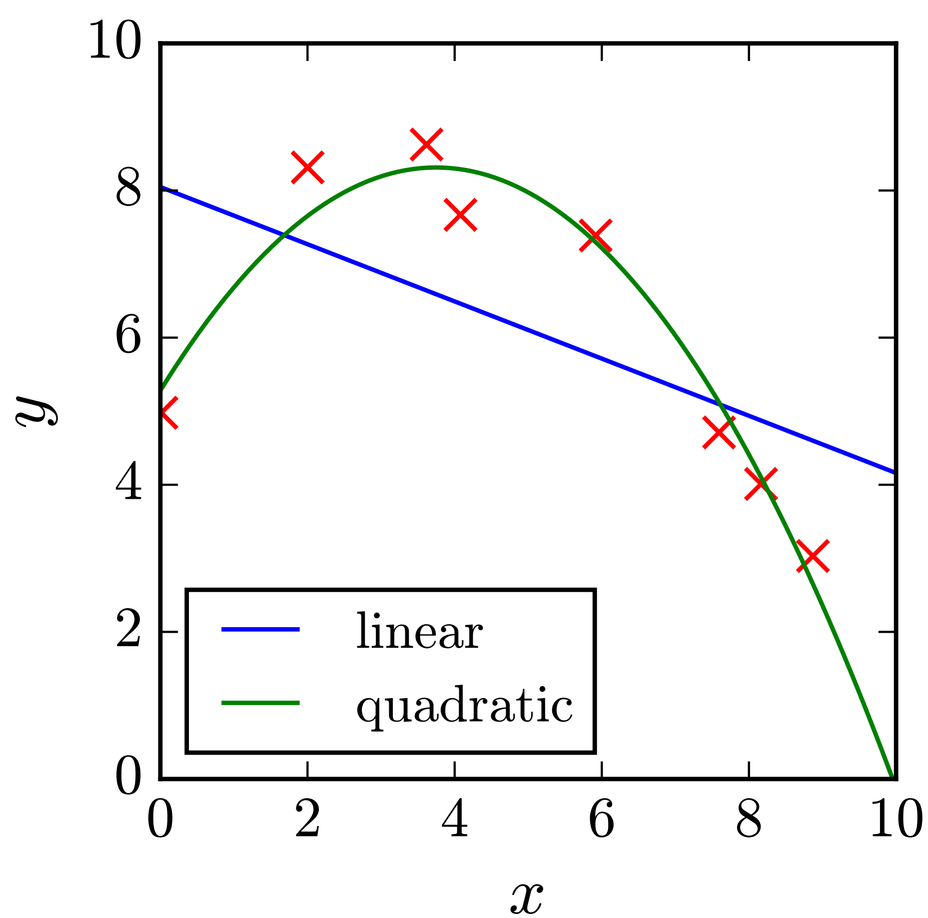
$$f_S(z) = \frac{1}{1 + e^{-z}}$$

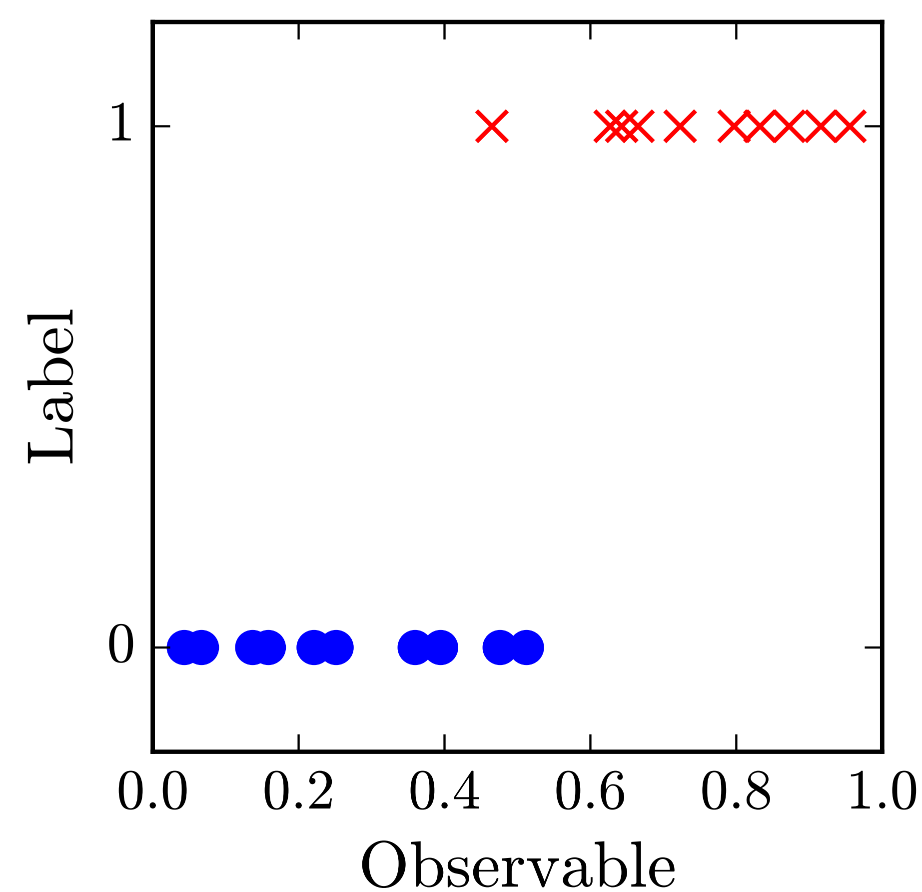
- Is the mean squared error still a good loss function?
  - What happens if a prediction is very far off?
  - What does it mean to be “far off”

Logistic Regression

What if we want to predict a class, not a number?

- Is the mean squared error still a good loss function?
  - What happens if a prediction is very far off?
  - What does it mean to be "far off"





Logistic Regression

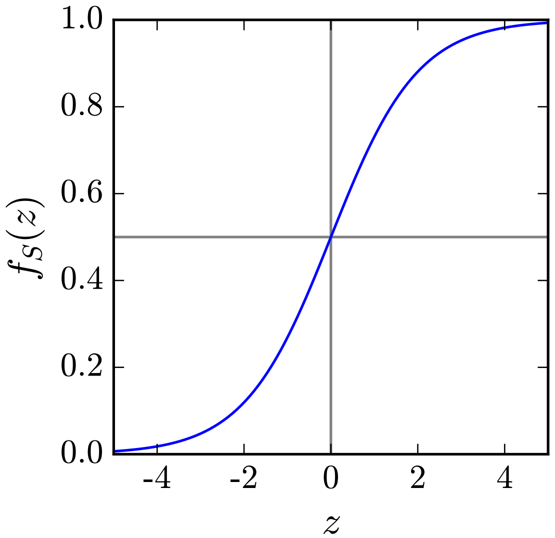
What if we want to predict a class, not a number?

- Is the mean squared error still a good loss function?
  - What happens if a prediction is very far off?
  - What does it mean to be “far off”
- Change the loss function: Binary Cross Entropy
  - Can be penalized for “confident” but wrong predictions

$$L(\vec{\text{pred}}, \vec{y}, \vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \log \text{pred}_\text{i} + (1 - \text{y}_\text{i}) \log\left(1 - \text{pred}_\text{i}\right)\Big)$$

Logistic Regression

What if we are trying to predict a class, not a number?

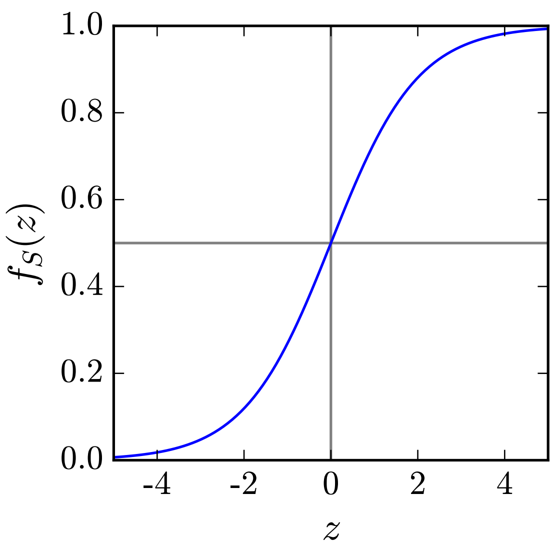


$$L(\vec{x}, \vec{y}, \vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(f_S(p(x,a))\right) + (1-y_i)\log\left(1 - f_S(p(x,a))\right)\right)$$

$$f_S(z) = \frac{1}{1+\mathrm{e}^{-z}} \qquad z = p(x,a)$$

Logistic Regression

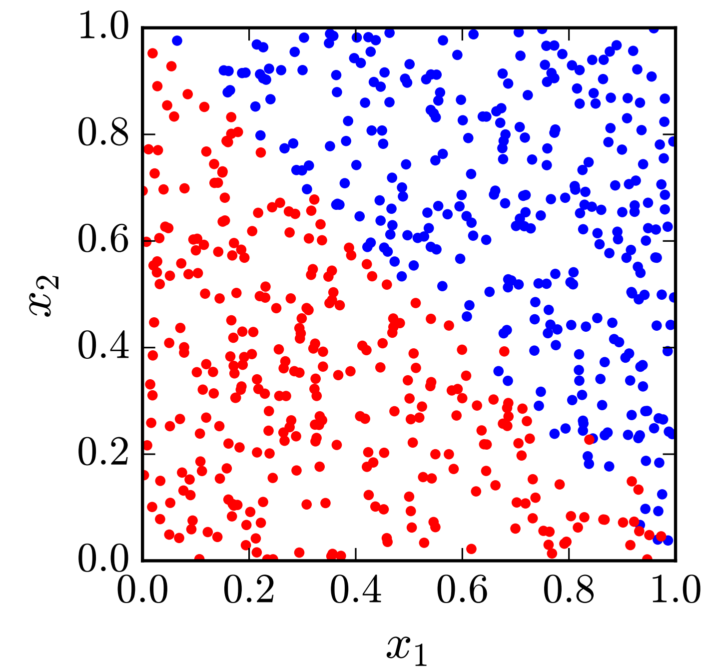
What if we are trying to predict a class, not a number?



$$L(\vec{x},\vec{y},\vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(f_S(p(x,a))\right) + (1-y_i)\log\left(1 - f_S(p(x,a))\right)\right)$$

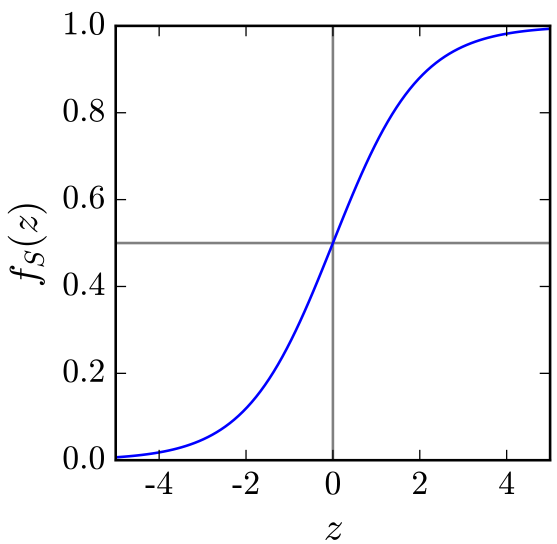
$$f_S(z) = \frac{1}{1+\mathrm{e}^{-z}} \qquad z = p(x,a)$$

What is $p(x,a)$?



Logistic Regression

What if we are trying to predict a class, not a number?

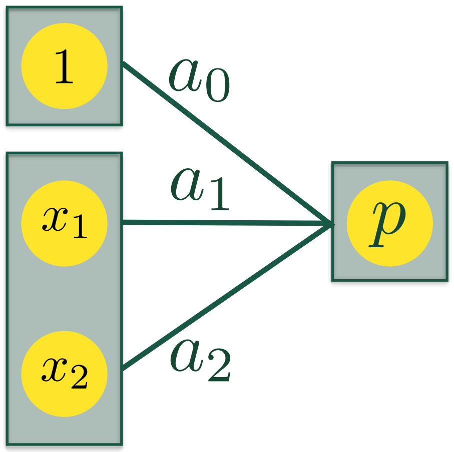


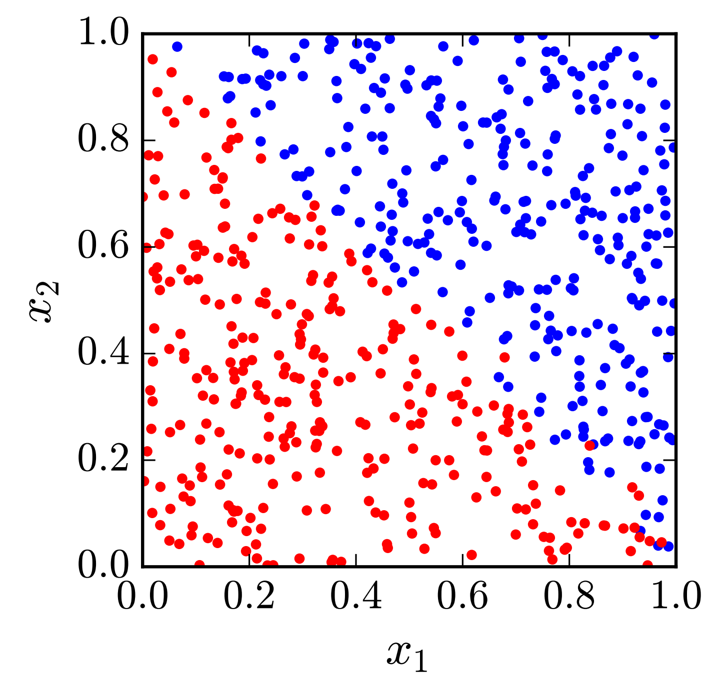
$$L(\vec{x}, \vec{y}, \vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \log\big(f_S(p(x,a))\big) + (1-y_i)\log\big(1 - f_S(p(x,a))\big)\Big)$$

$$f_S(z) = \frac{1}{1+\mathrm{e}^{-z}} \qquad z = p(x,a)$$

What is $p(x,a)$?

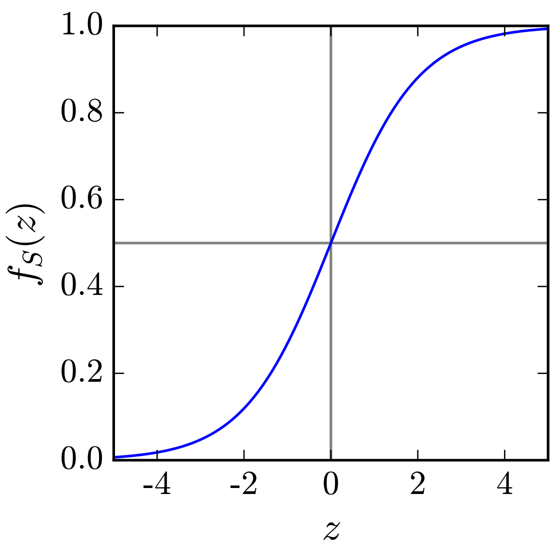
$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$





Logistic Regression

What if we are trying to predict a class, not a number?

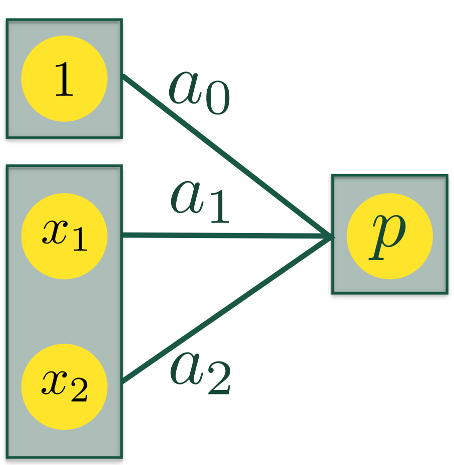


$$L(\vec{x},\vec{y},\vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(f_S(p(x,a))\right) + (1-y_i)\log\left(1 - f_S(p(x,a))\right)\right)$$

$$f_S(z) = \frac{1}{1+\mathrm{e}^{-z}} \qquad z = p(x,a)$$

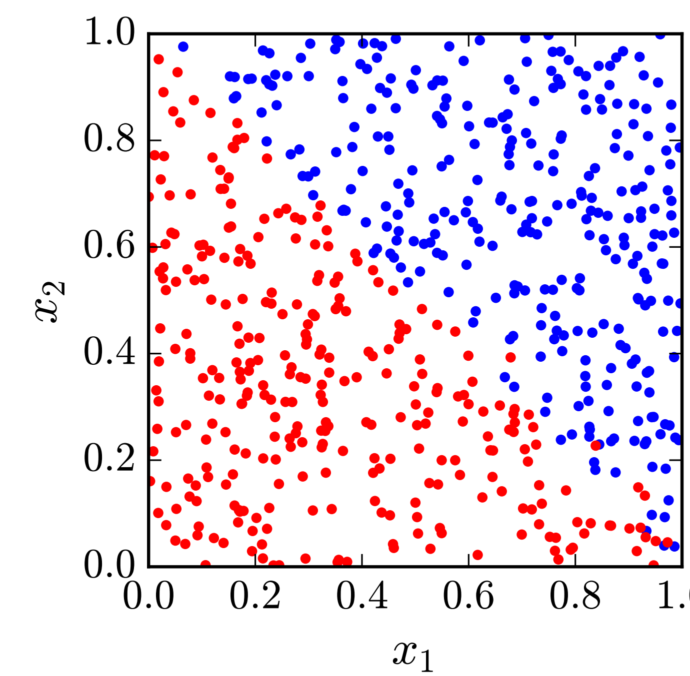
What is $p(x,a)$?

$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$



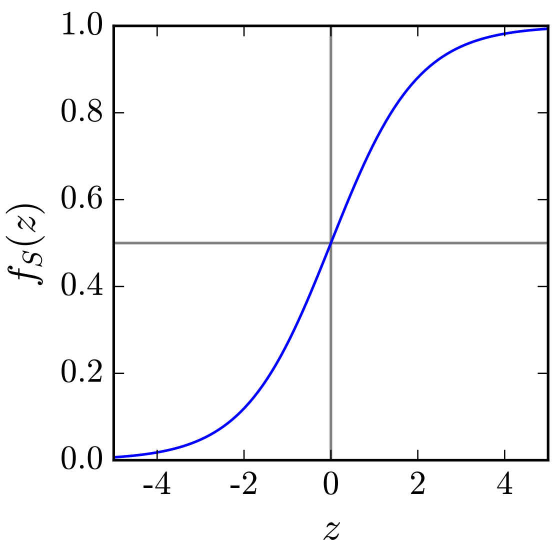
Minimize the loss with respect to $\vec{a}$

Boundary at $p(x,a) = 0$



Logistic Regression

What if we are trying to predict a class, not a number?



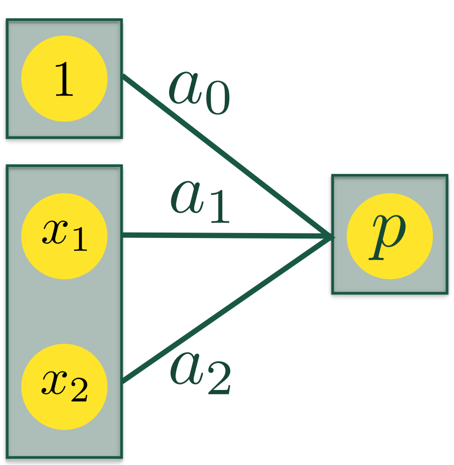
$$L(\vec{x}, \vec{y}, \vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log\left(f_S(p(x,a))\right) + (1-y_i)\log\left(1 - f_S(p(x,a))\right)\right)$$

$$f_S(z) = \frac{1}{1+\mathrm{e}^{-z}} \qquad z = p(x,a)$$

What is $p(x,a)$?

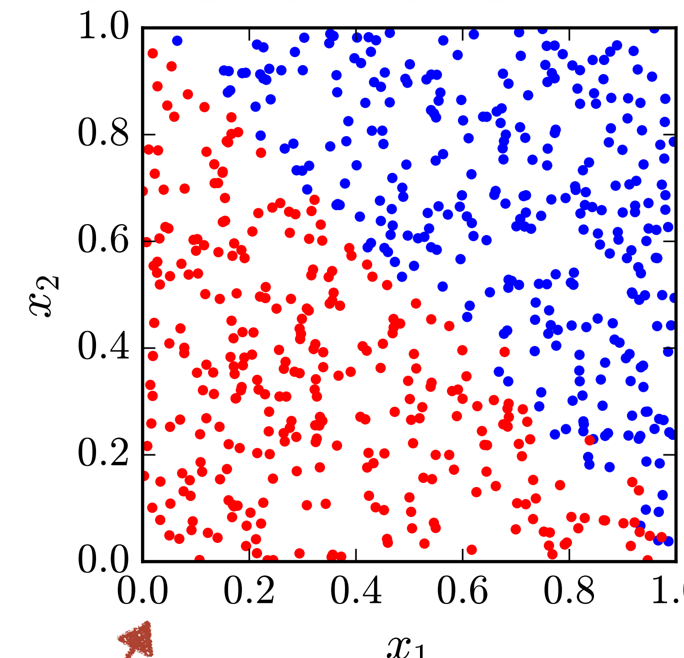
Large + values of p

$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$



Minimize the loss with respect to $\vec{a}$

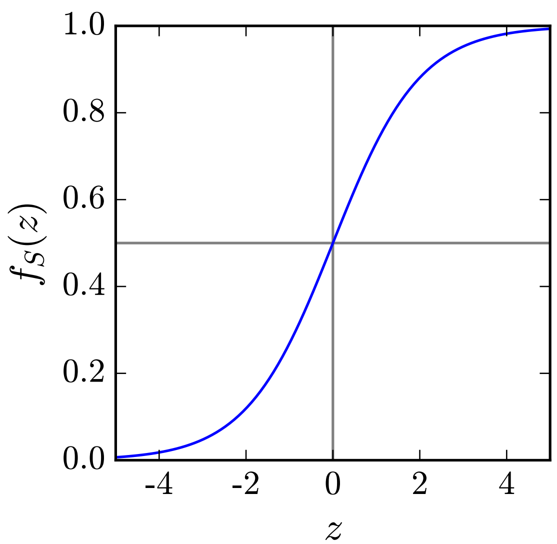
Boundary at $p(x,a) = 0$



Large - values of p

Logistic Regression

What if we are trying to predict a class, not a number?

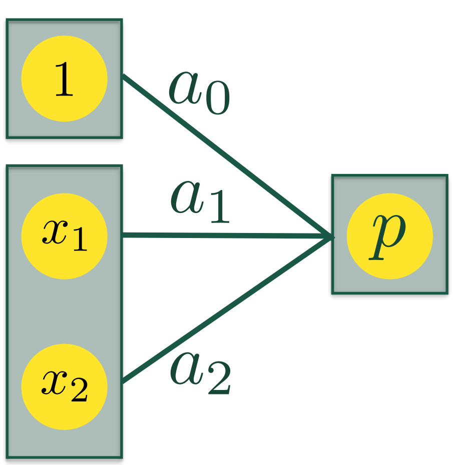


$$L(\vec{x},\vec{y},\vec{a}) = -\frac{1}{N}\sum_{i=1}^{N}\Big(y_i \log\big(f_S(p(x,a))\big) + (1-y_i)\log\big(1-f_S(p(x,a))\big)\Big)$$

$$f_S(z) = \frac{1}{1+\mathrm{e}^{-z}} \qquad z = p(x,a)$$

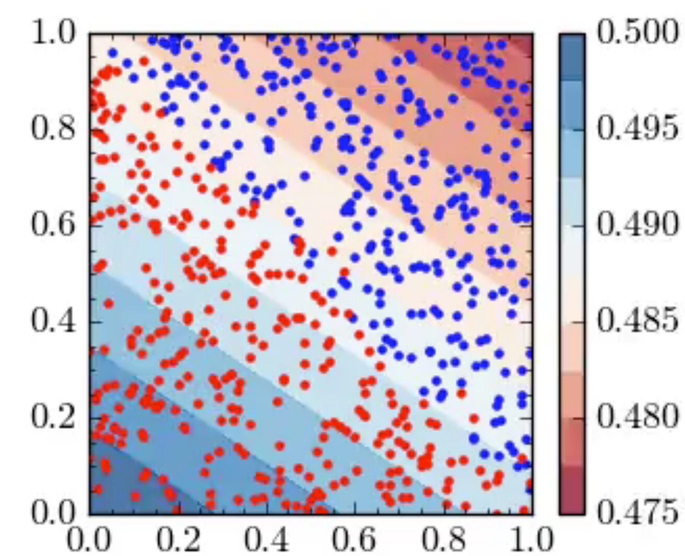
What is $p(x,a)$?

$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$



Minimize the loss with respect to $\vec{a}$

Boundary at $p(x,a) = 0$

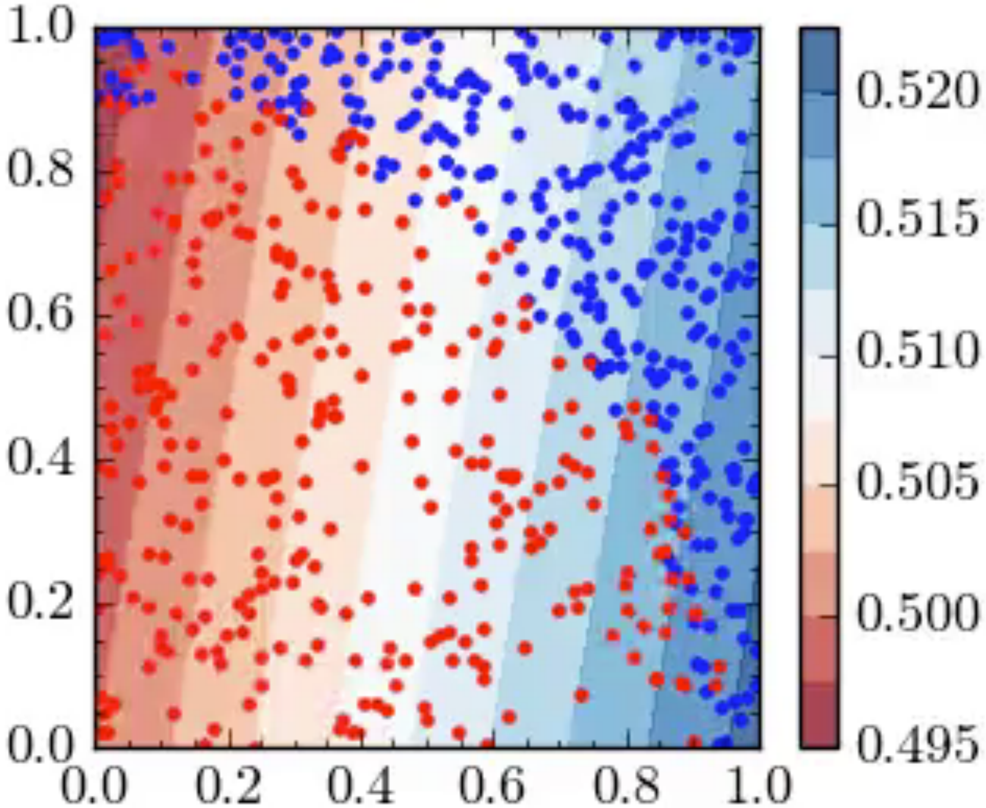


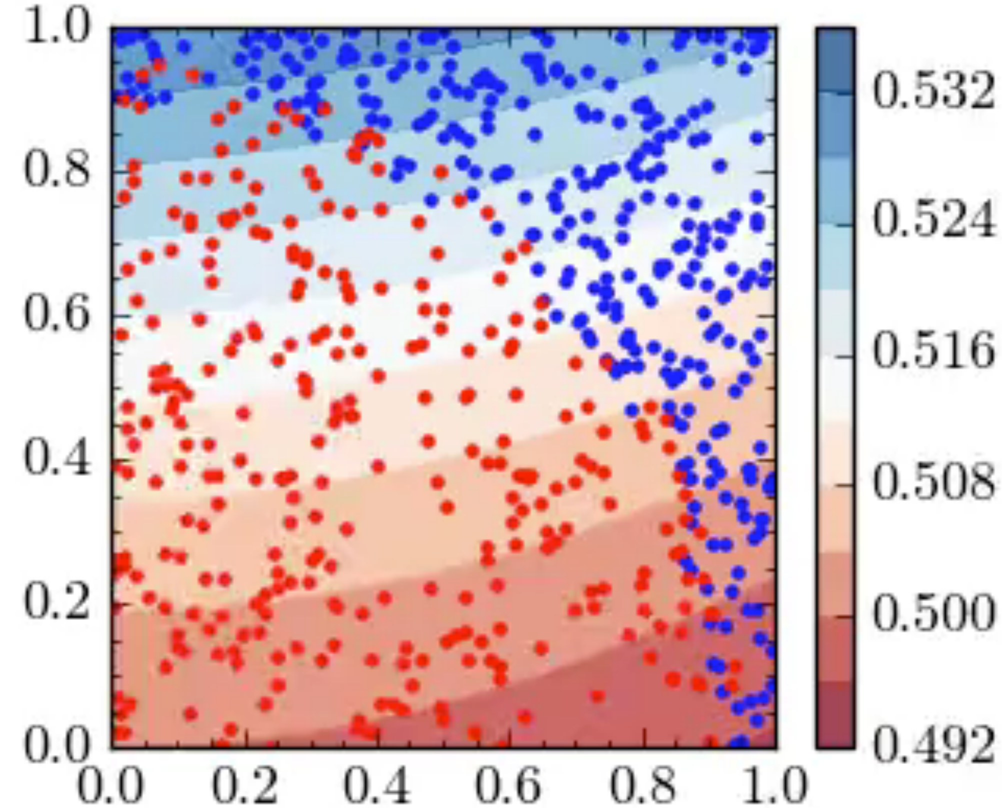
Logistic Regression

What if there is a shape in the data?

$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$

$$p(x,a) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2$$



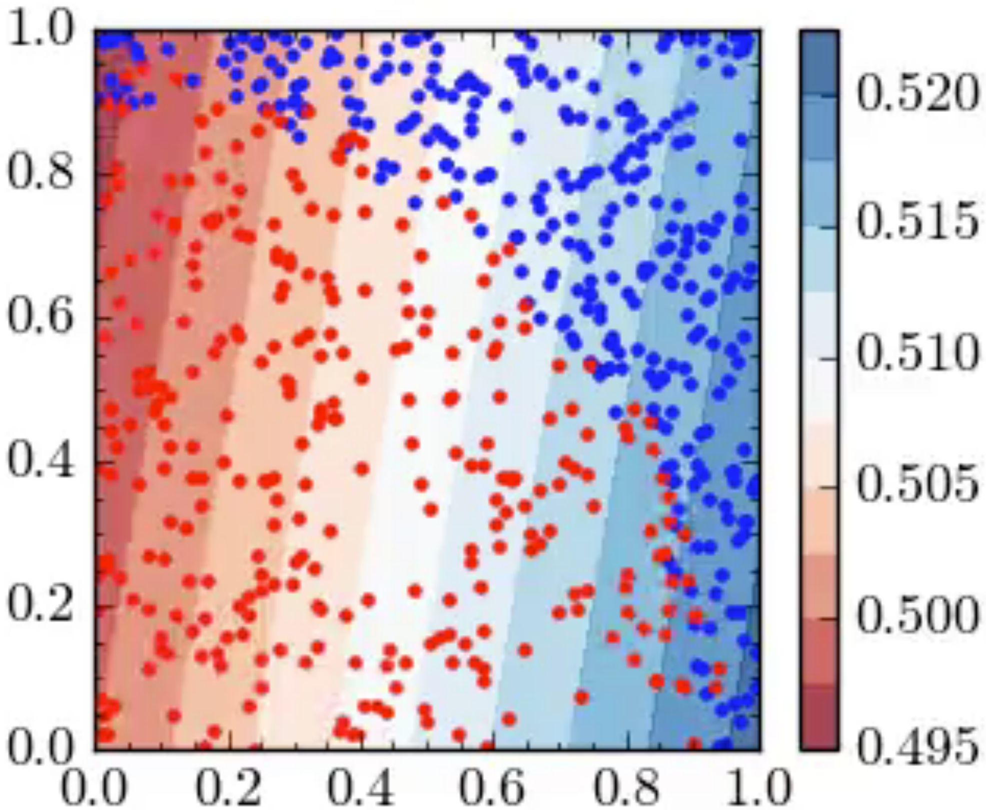


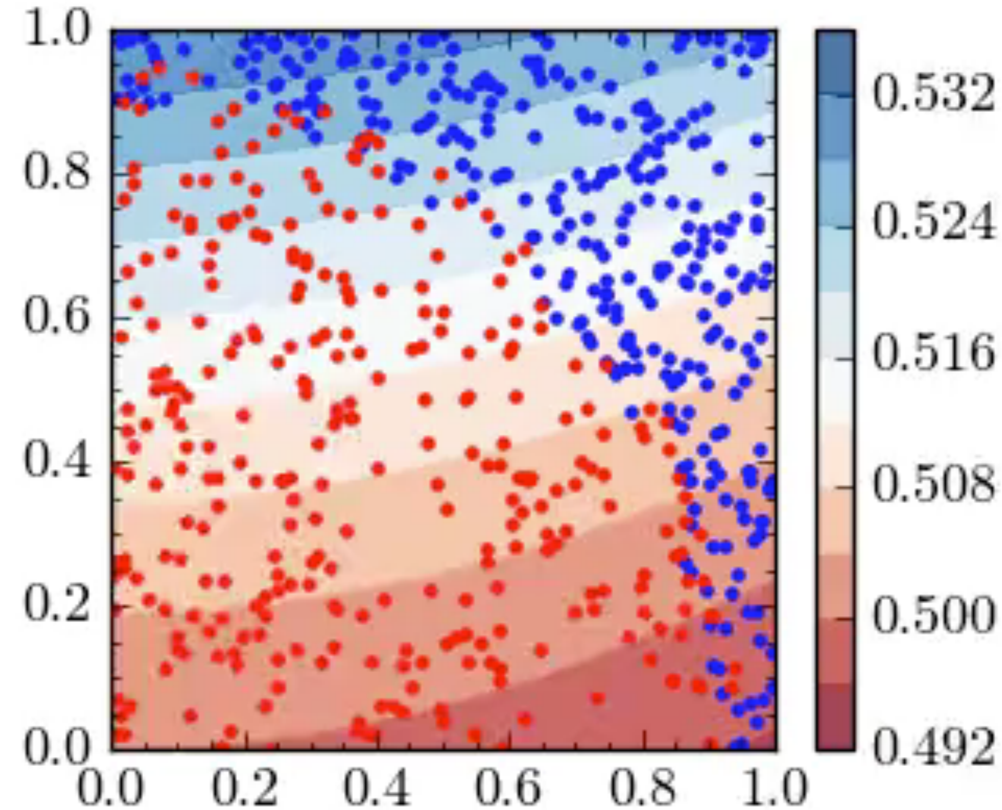
Logistic Regression

What if there is a shape in the data?

$$p(x,a) = a_0 + x_1 a_1 + x_2 a_2$$

$$p(x,a) = a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1^2 + a_4 x_2^2 + a_5 x_1 x_2$$



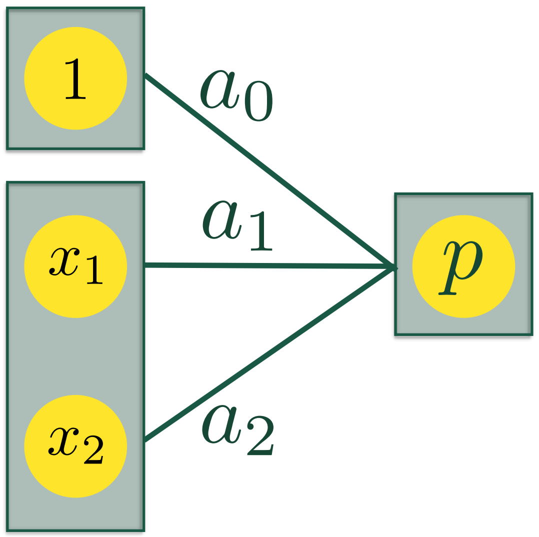


Logistic Regression

1. Can use nearly the same process for fitting a curve (predicting a number) or classification
2. Minimize a defined cost function
3. Easy to add parameters if shape is unknown — worry about over-fitting
4. If many inputs and complicated shapes, number of parameters necessary grows very quickly

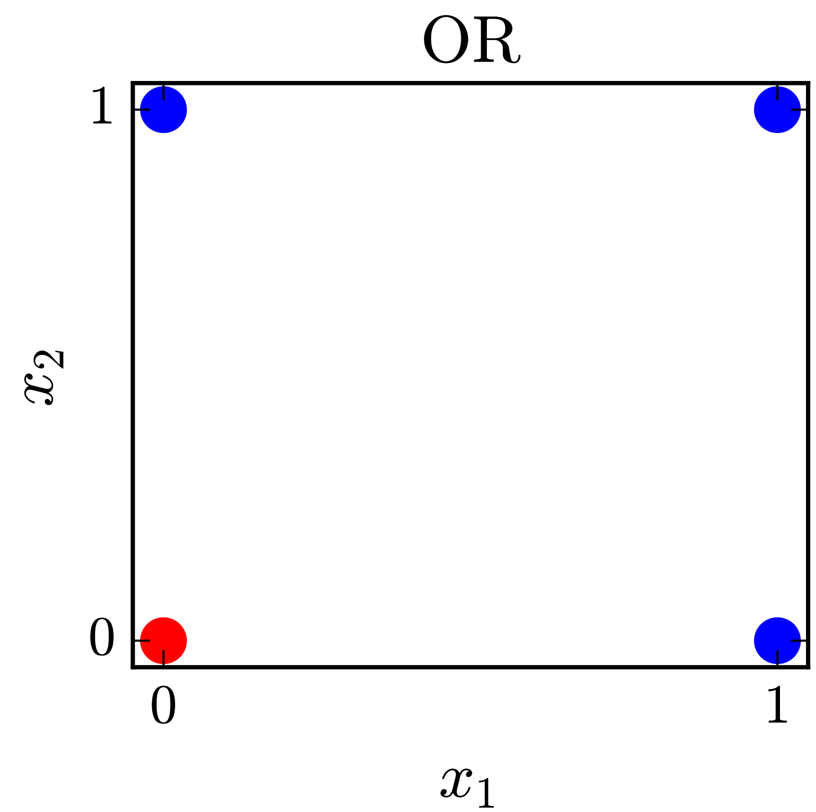
Why use more complicated algorithms?

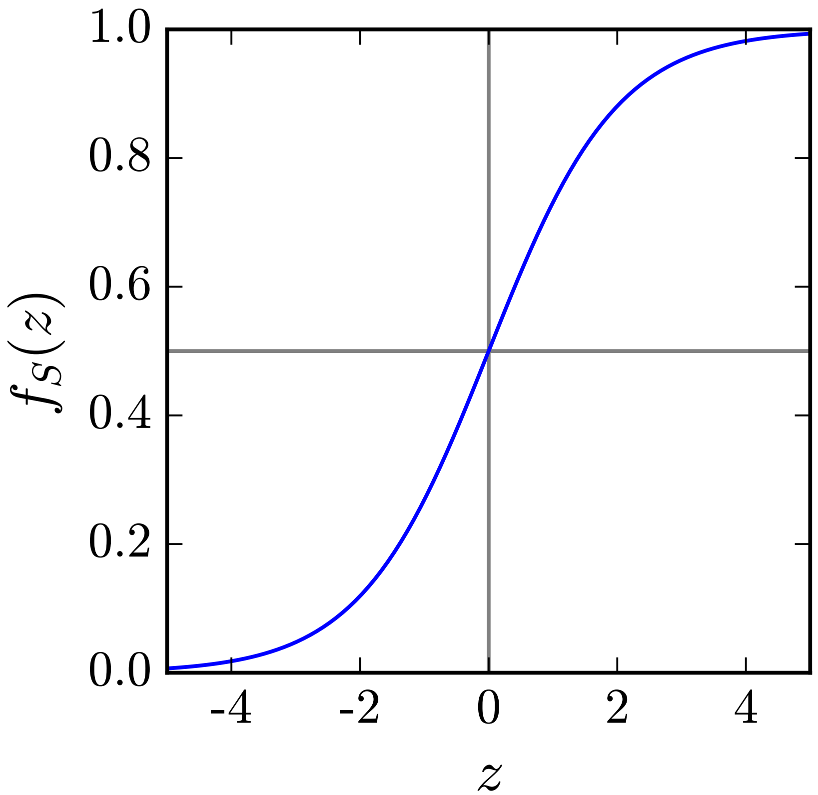
Neural Networks



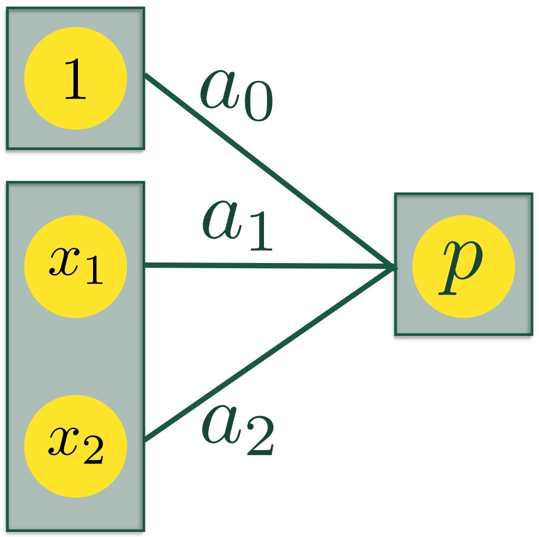
OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



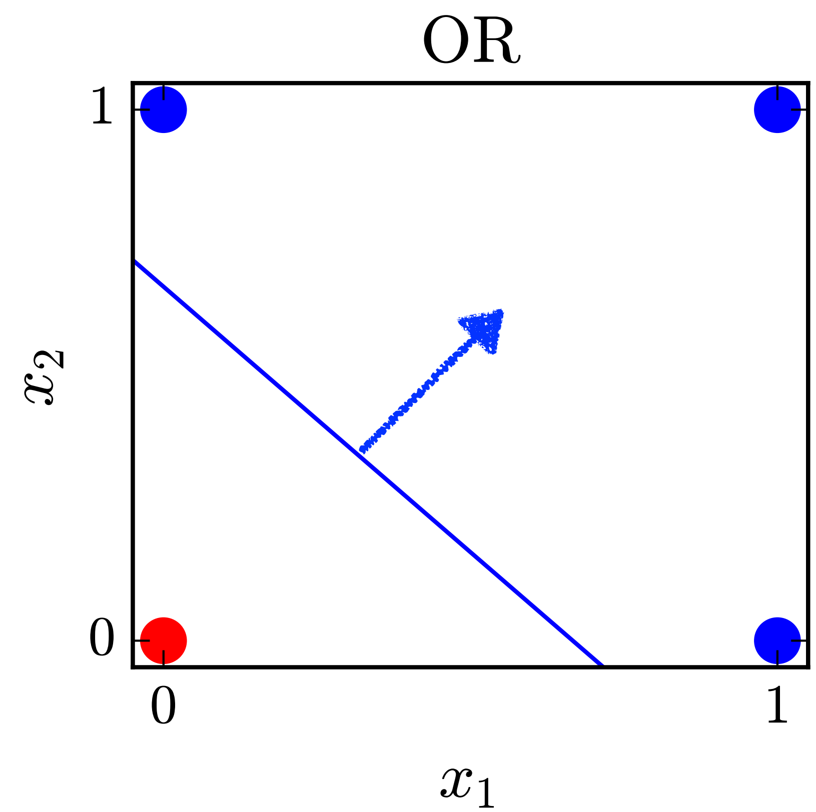


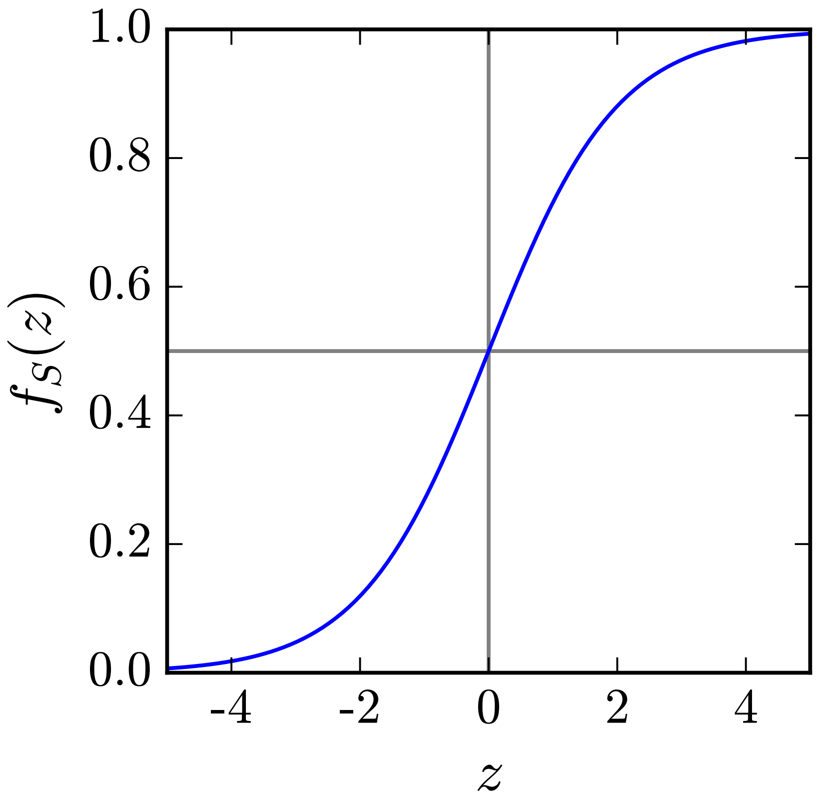
Neural Networks



OR

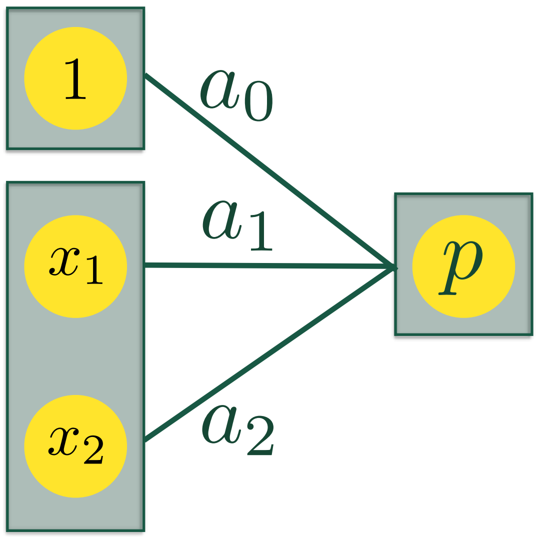
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

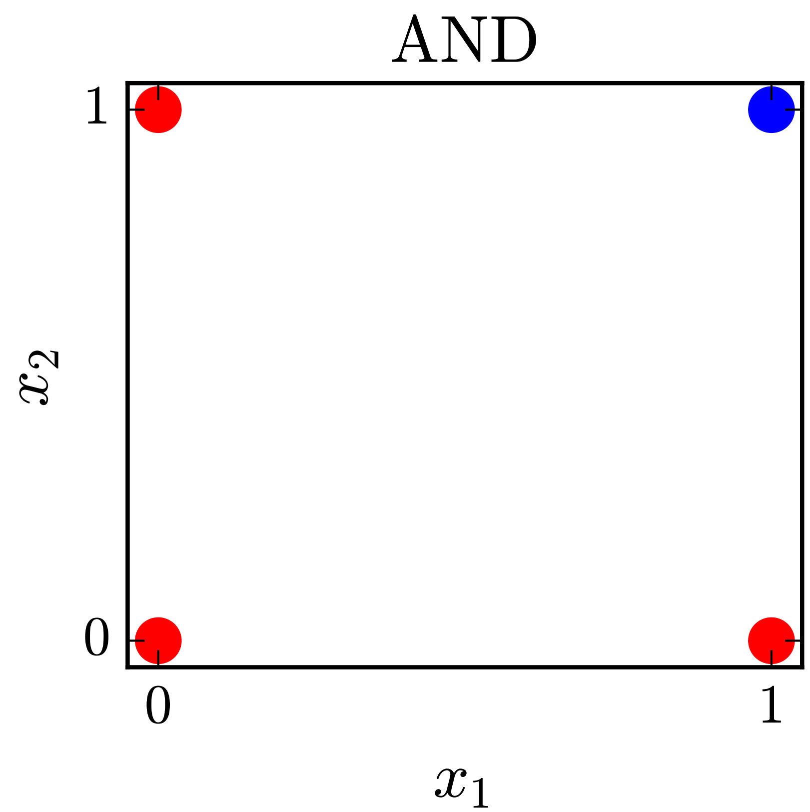




$$a_0 = -10, \quad a_1 = 15, \quad a_2 = 15$$

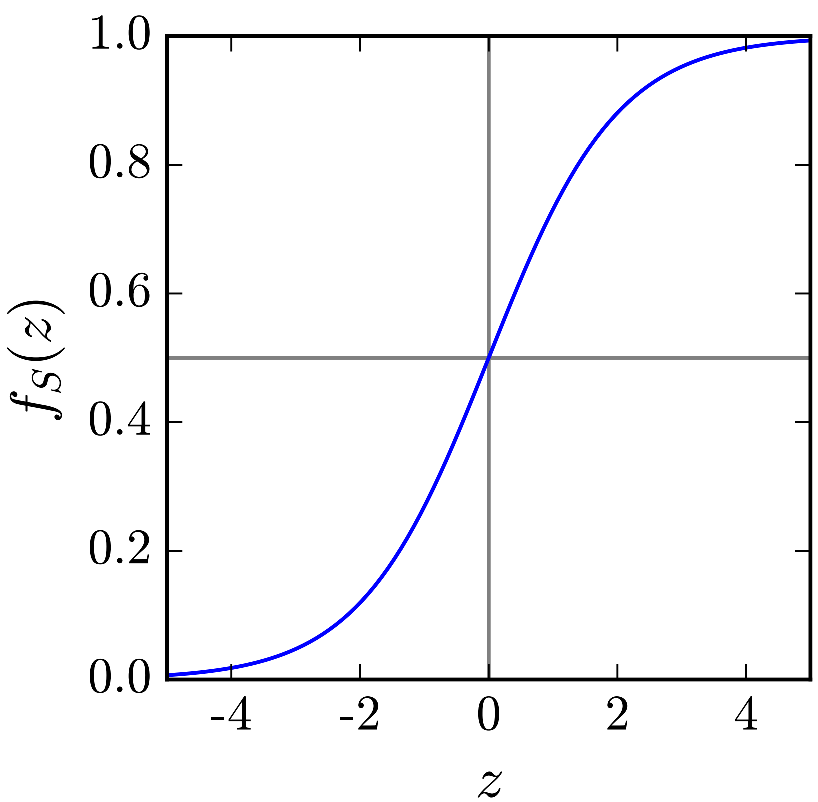
Neural Networks



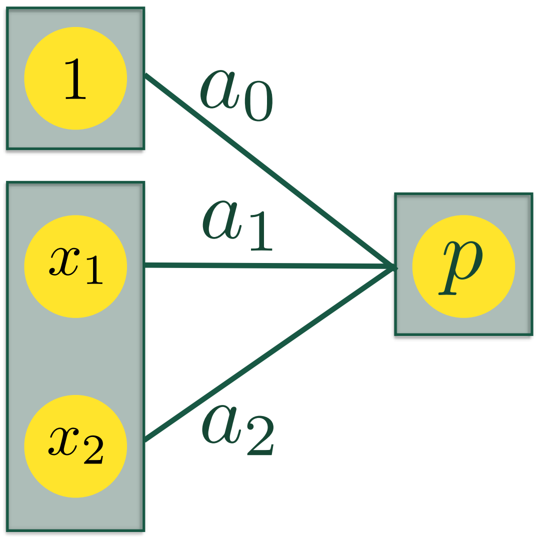


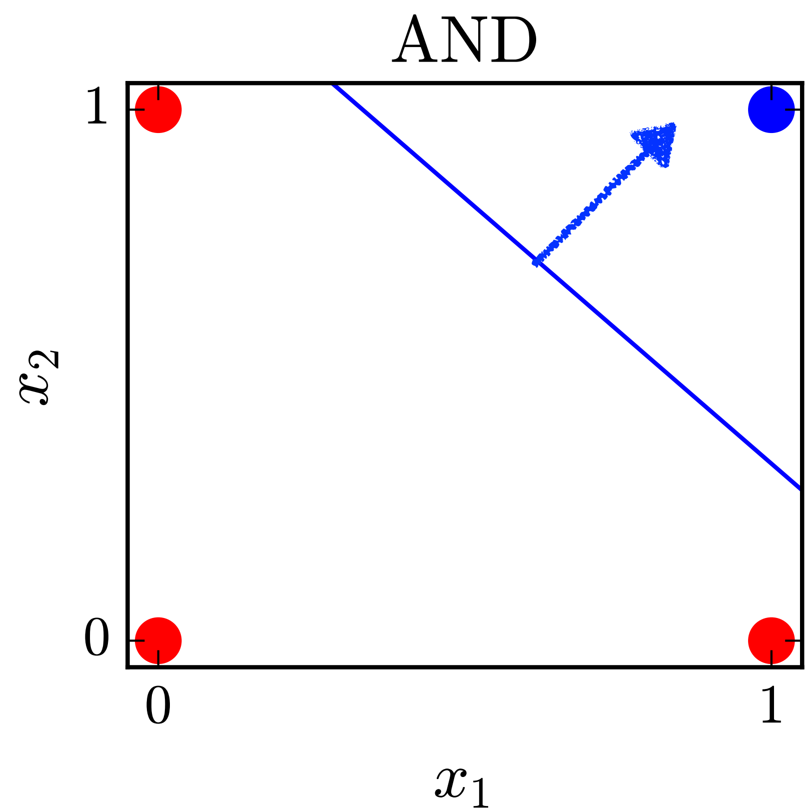
AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



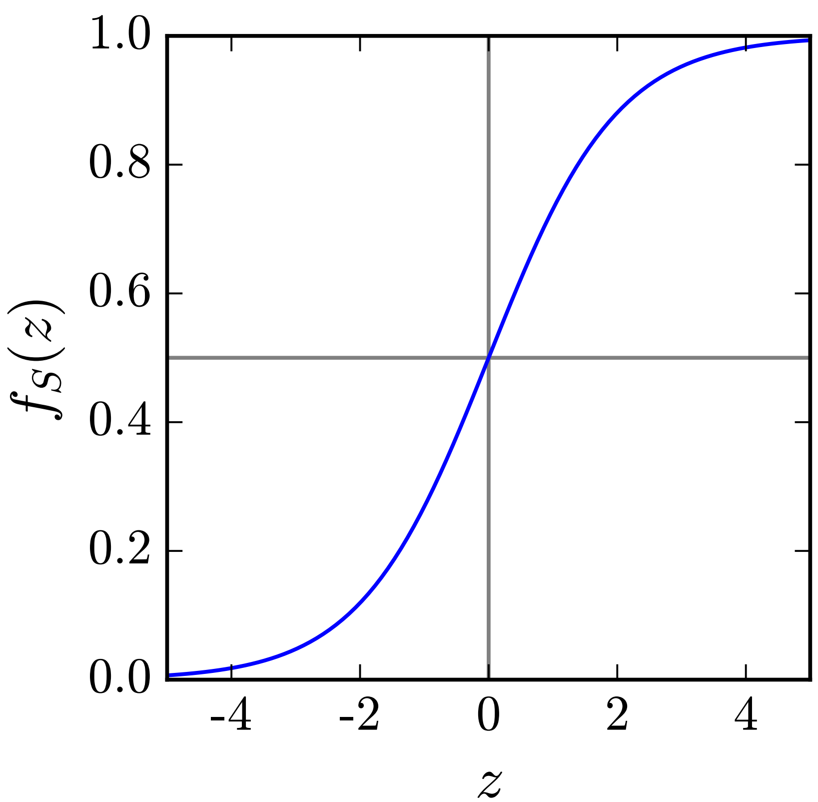
Neural Networks





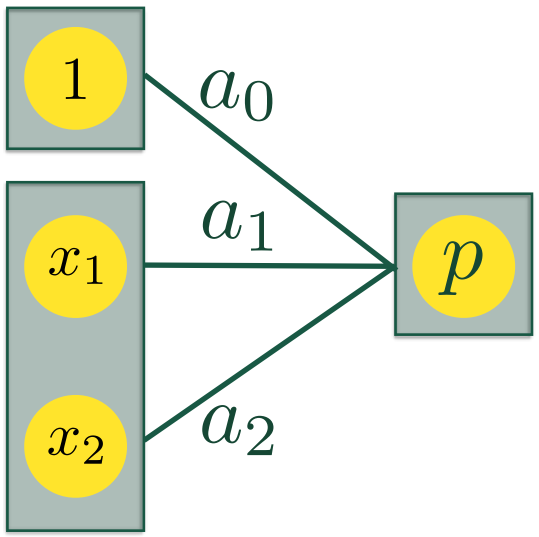
AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



$$a_0 = -20, \quad a_1 = 15, \quad a_2 = 15$$

Neural Networks



OR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	1

$$a_0 = -10, \quad a_1 = 15, \quad a_2 = 15$$

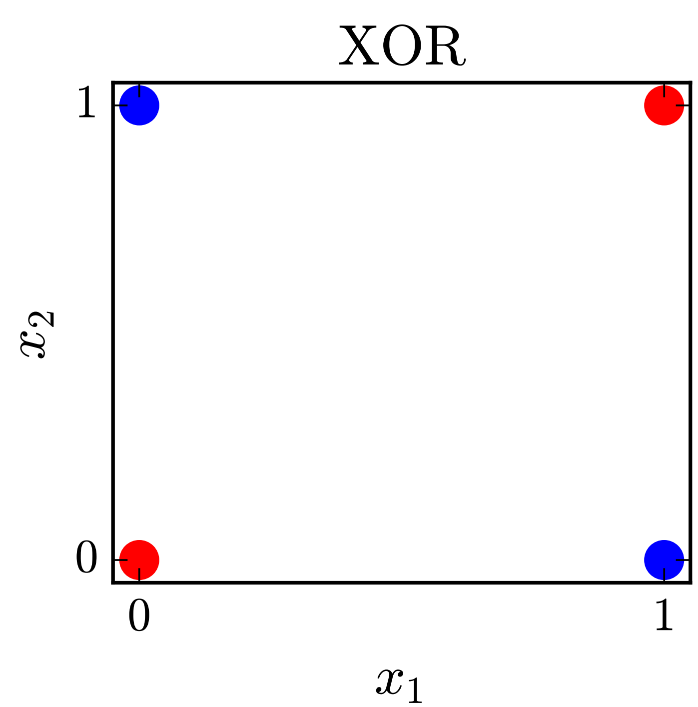
AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

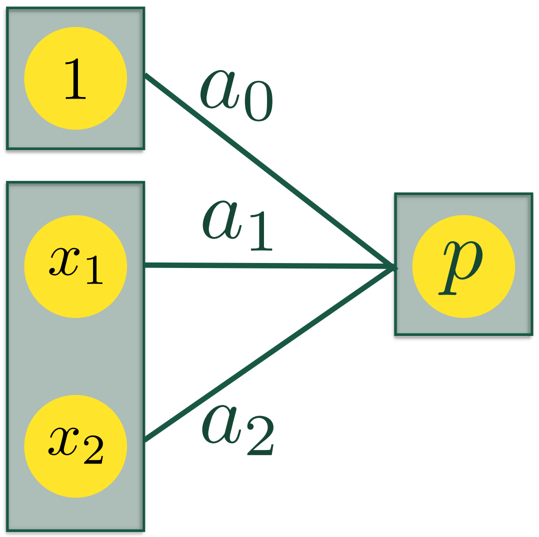
$$a_0 = -20, \quad a_1 = 15, \quad a_2 = 15$$

XOR

x1	x2	y
0	0	0
0	1	1
1	0	1
1	1	0



Neural Networks



OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

$$a_0 = -10, \quad a_1 = 15, \quad a_2 = 15$$

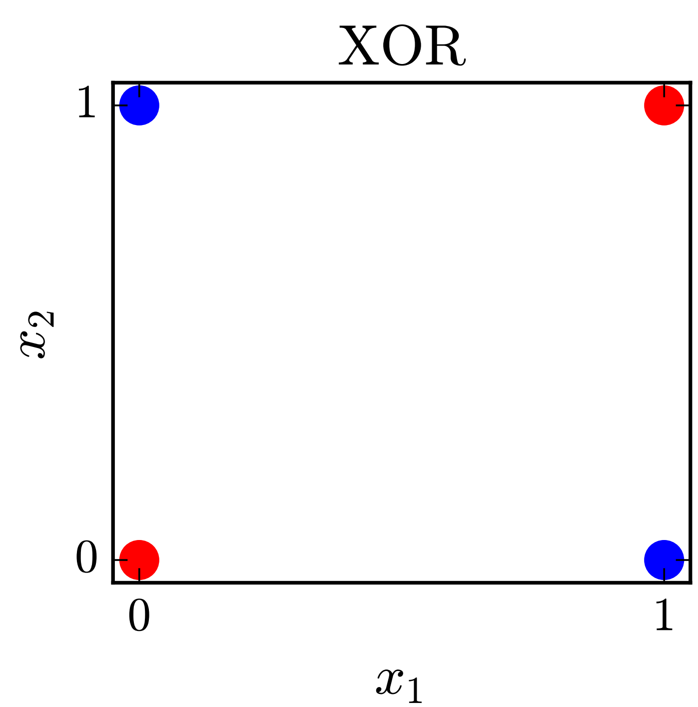
AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$a_0 = -20, \quad a_1 = 15, \quad a_2 = 15$$

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



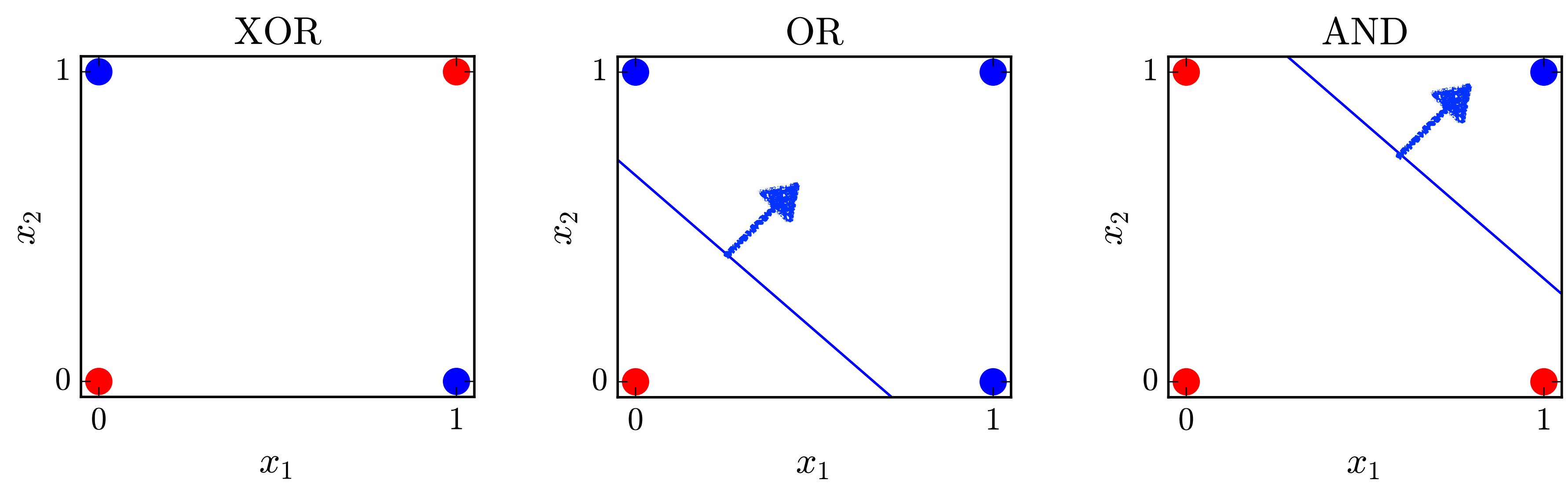
This system cannot produce XOR

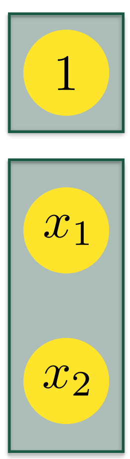
(cannot make a two sided cut)

Neural Networks

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

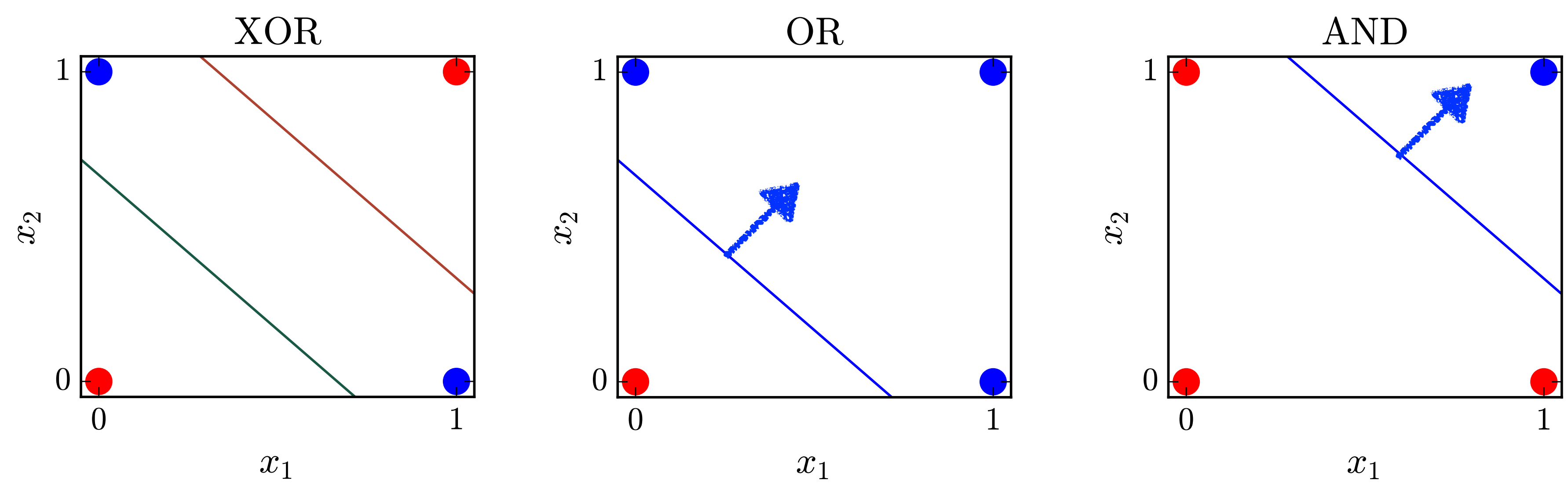


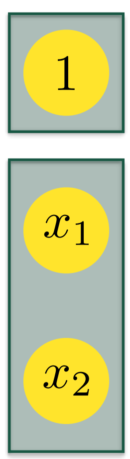


Neural Networks

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

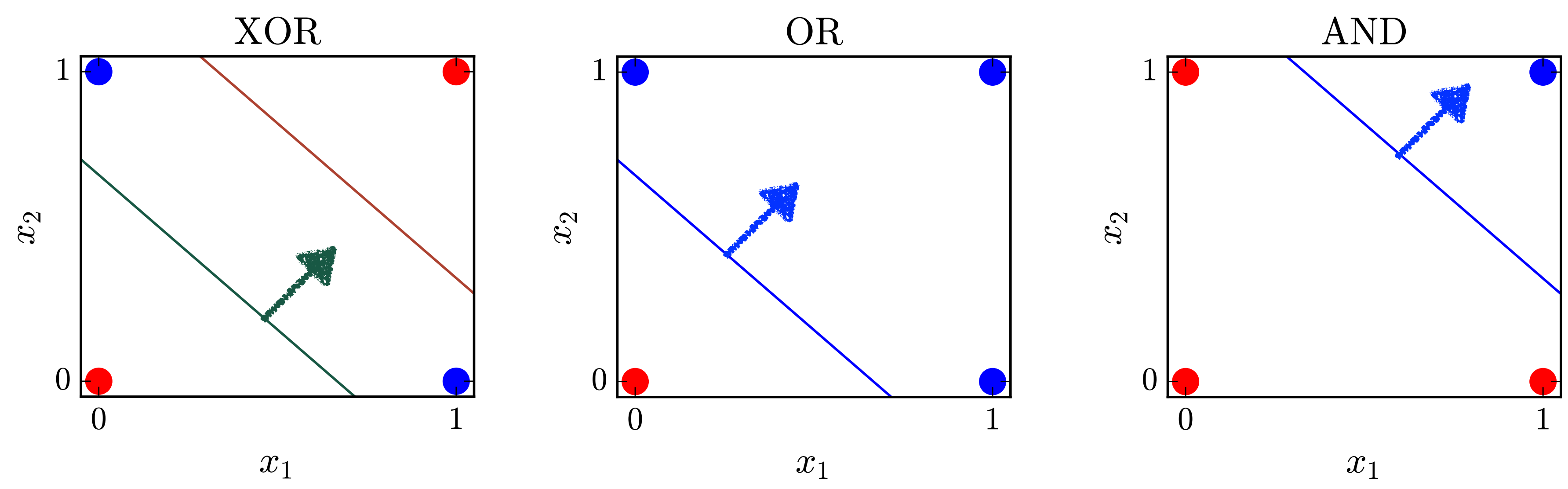


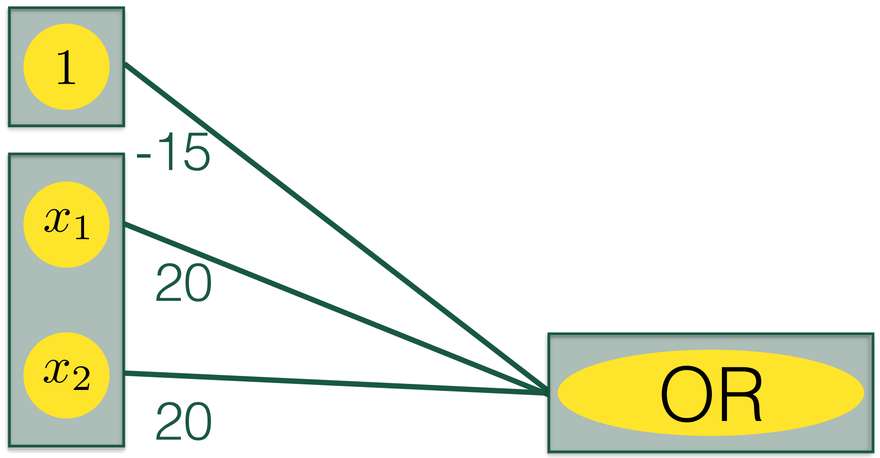


Neural Networks

XOR

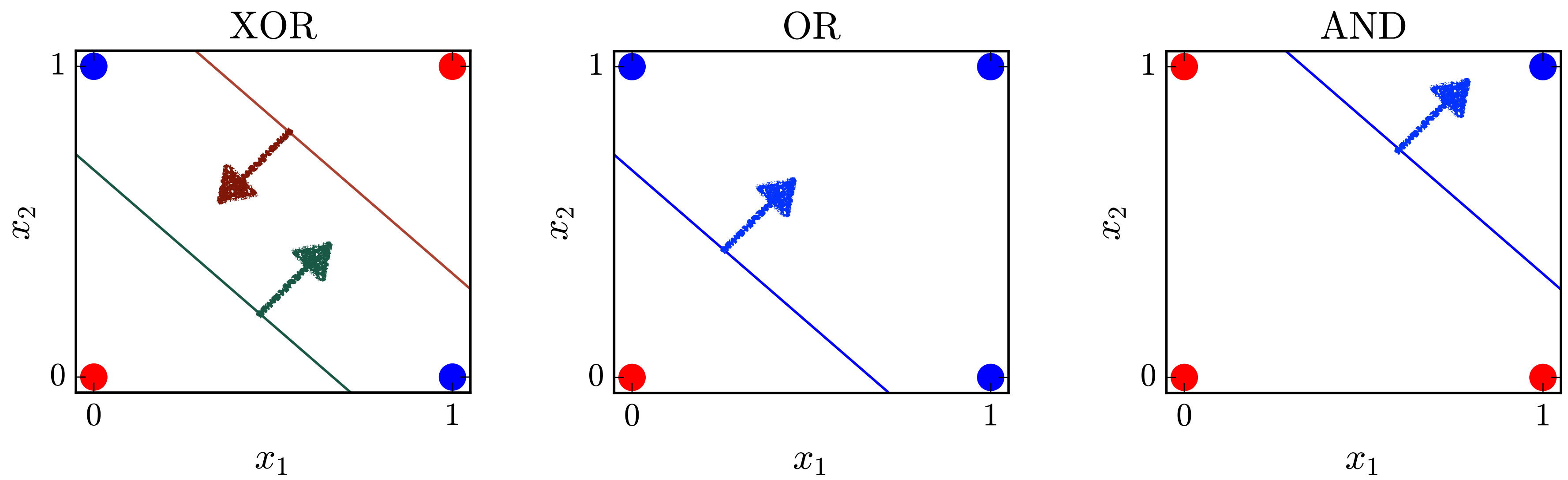
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

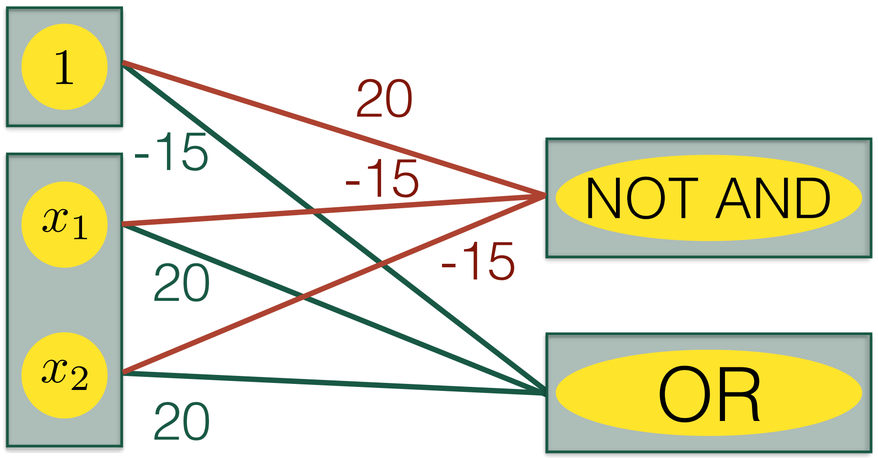




Neural Networks

XOR		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

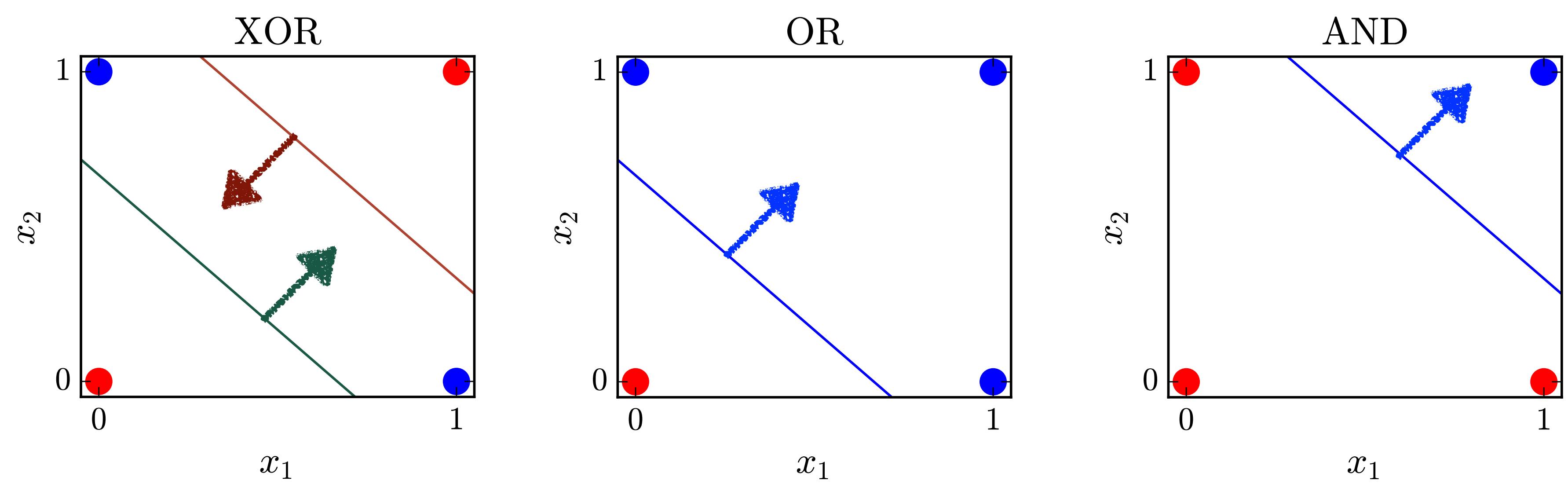


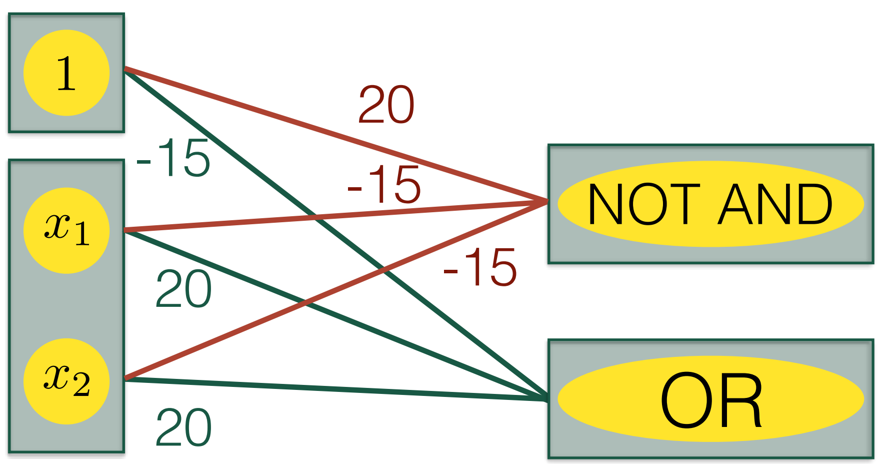


Neural Networks

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

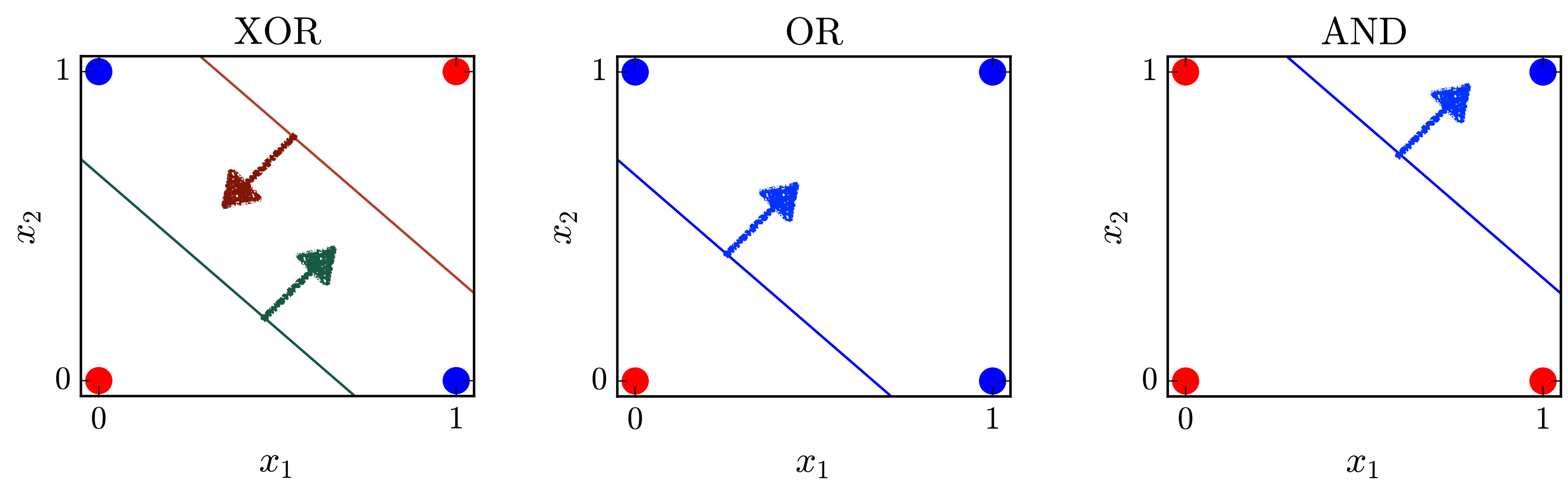


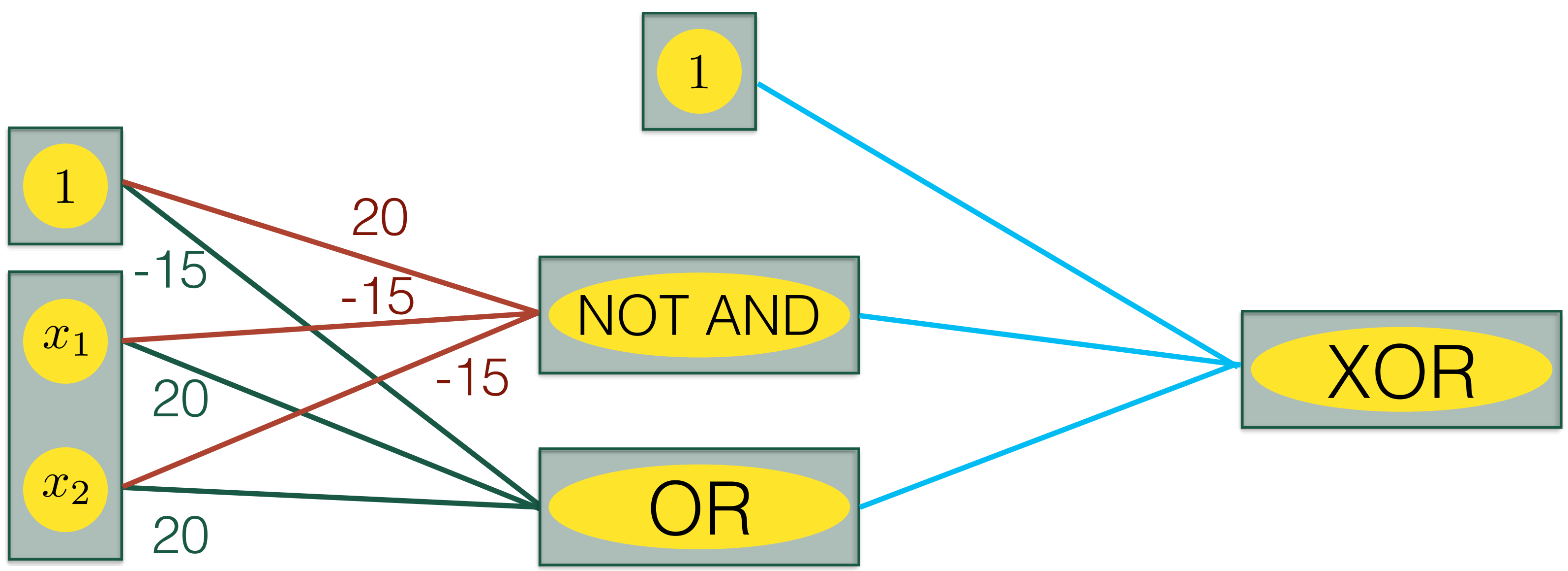


x_1	x_2	OR	NOT AND	XOR
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Neural Networks

XOR		
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



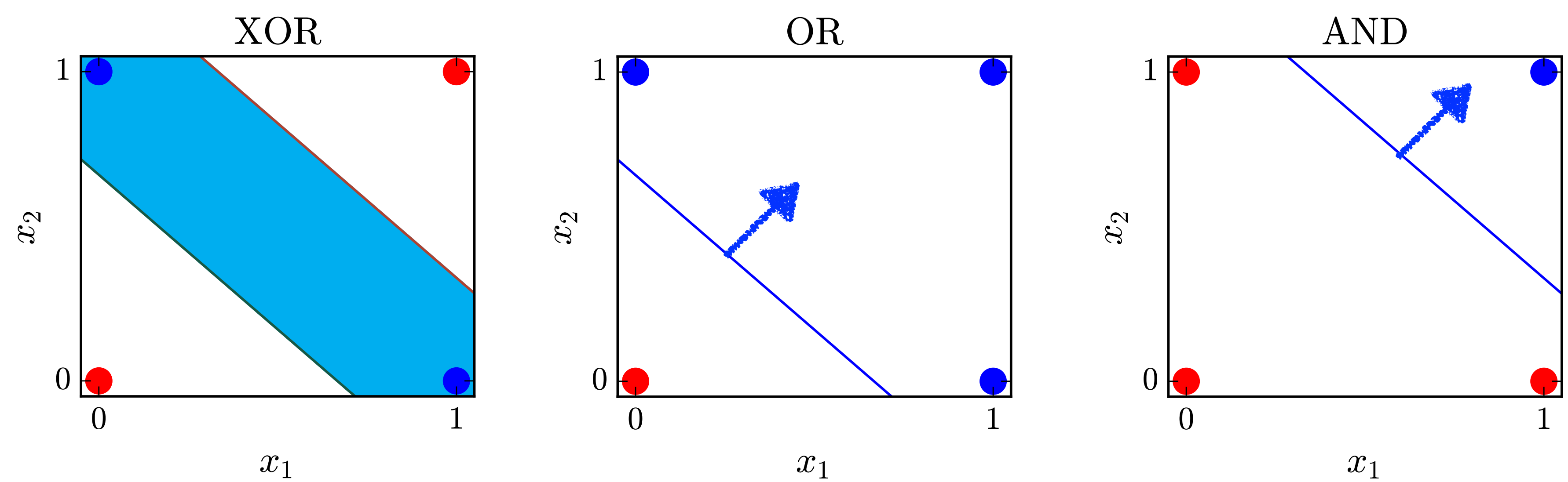


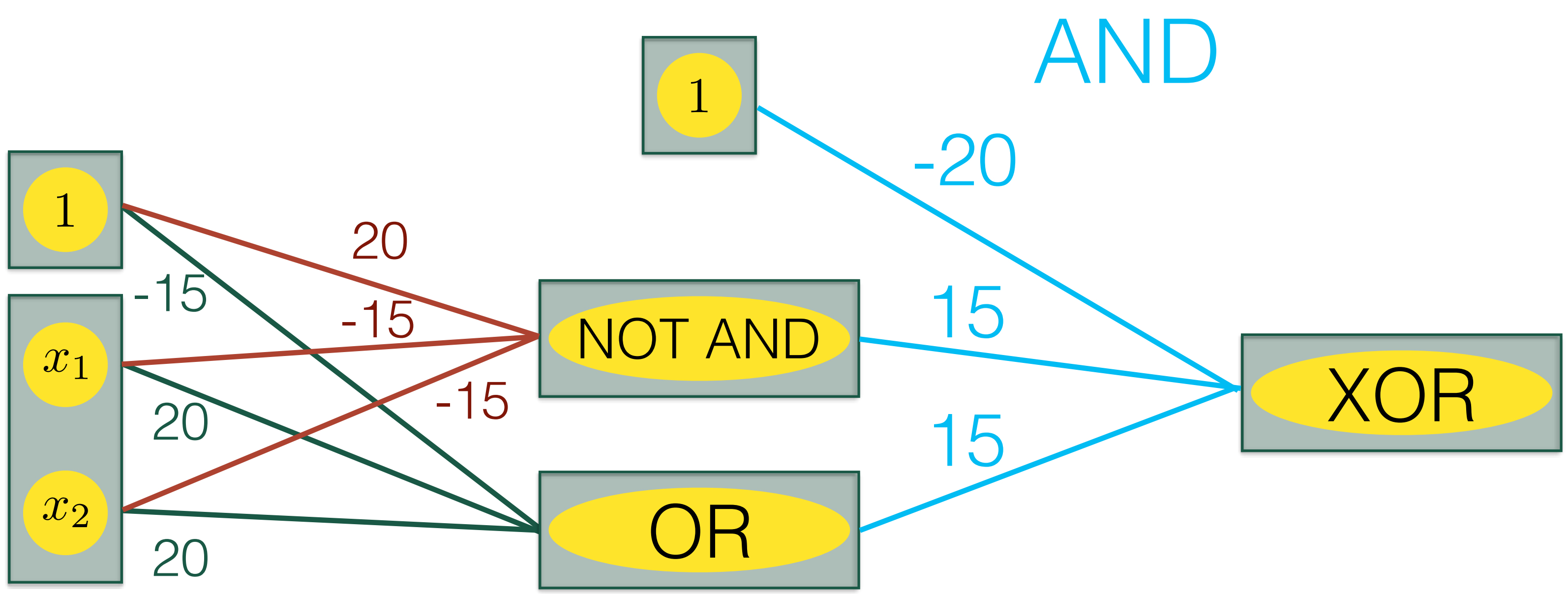
x_1	x_2	OR	NOT AND	XOR
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Neural Networks

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



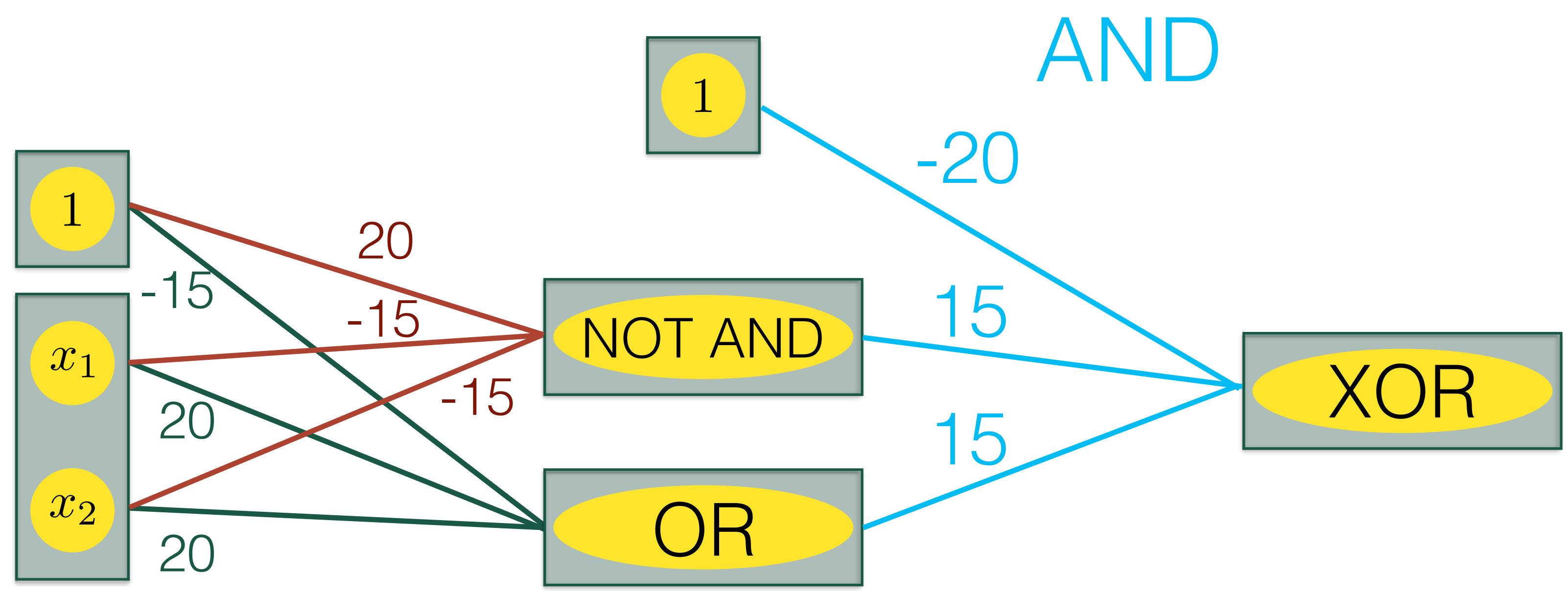


x_1	x_2	OR	NOT AND	XOR
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Neural Networks

XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

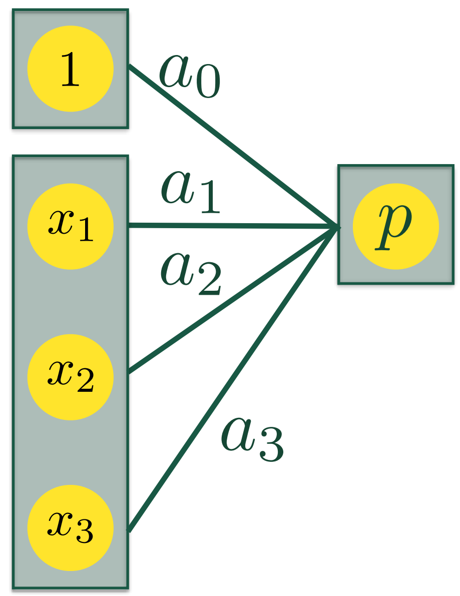


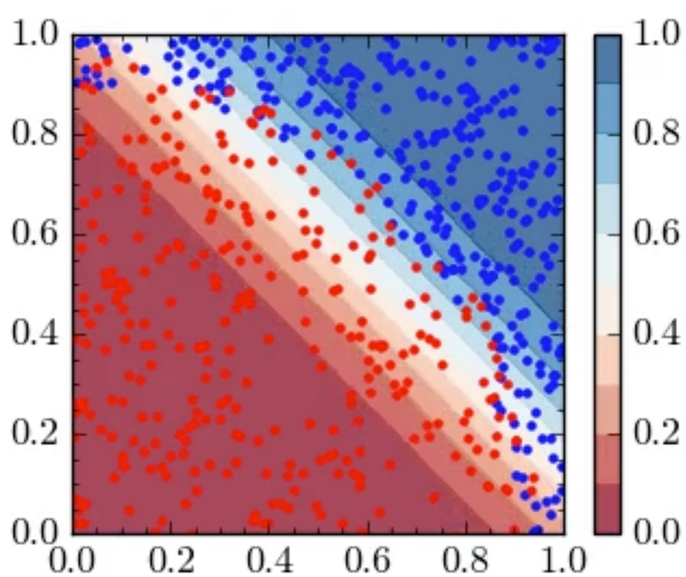
Simple example showing that neural network can access ‘high-level’ functions

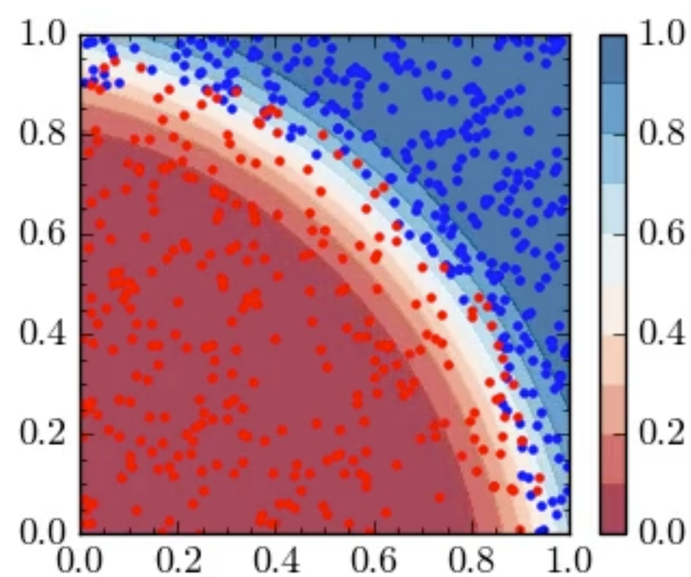
To learn weights, need large training set and CPU time

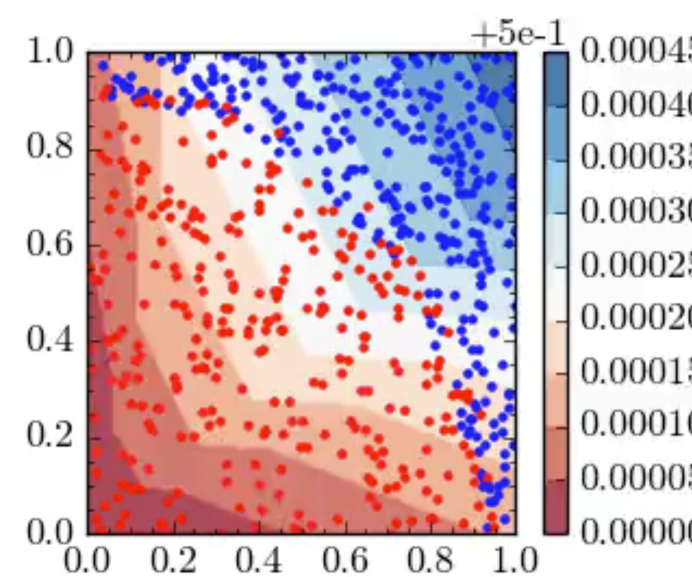
Neural Networks

- Don't add more inputs, let machine find own shape
- Ability to learn 'any' function
- More nodes/hidden layers allows for more complex features



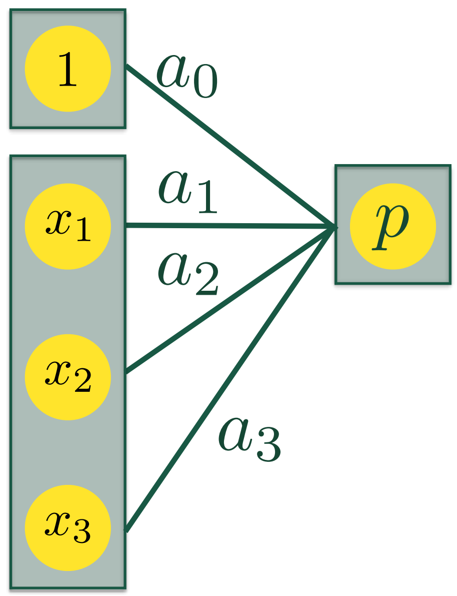


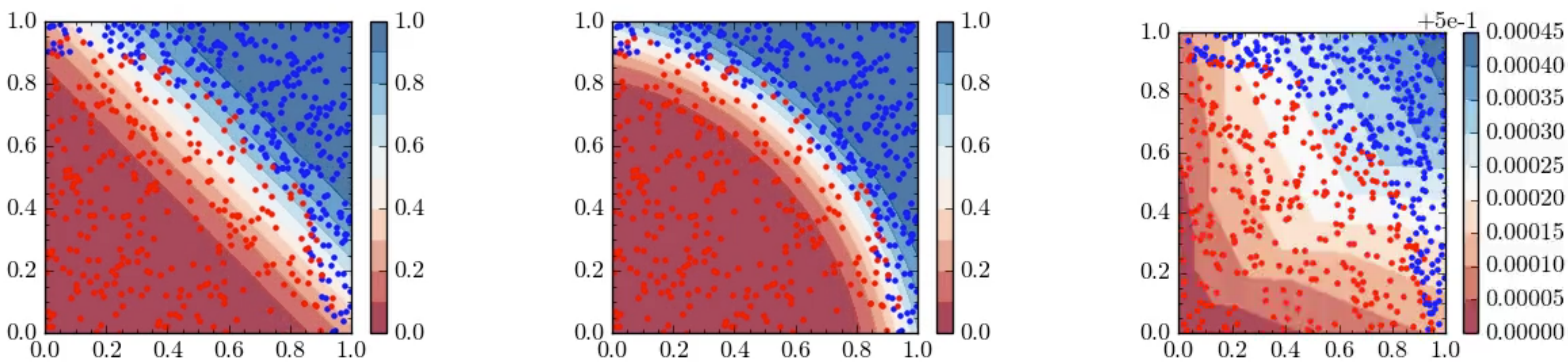




Neural Networks

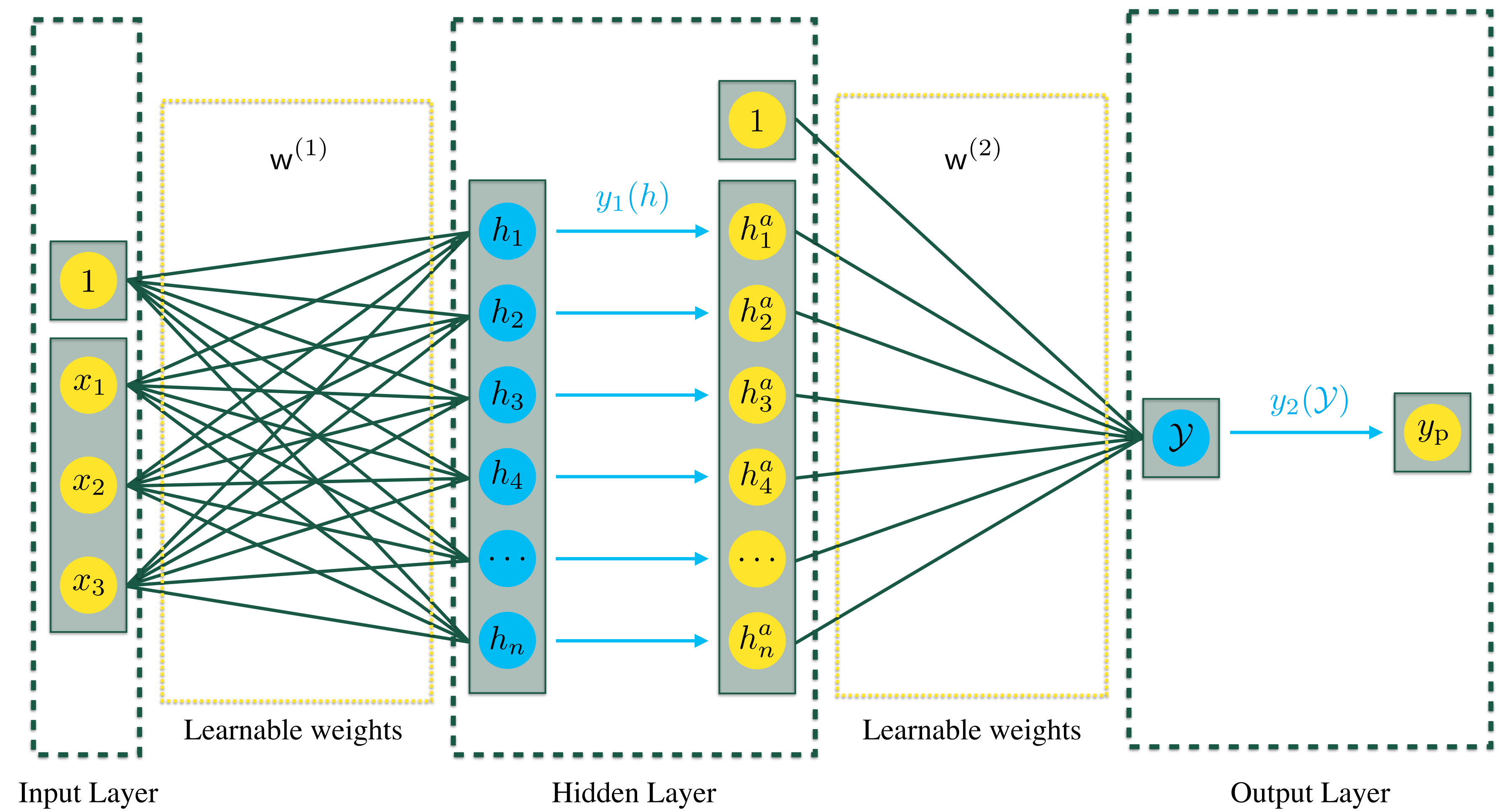
- Don't add more inputs, let machine find own shape
- Ability to learn 'any' function
- More nodes/hidden layers allows for more complex features

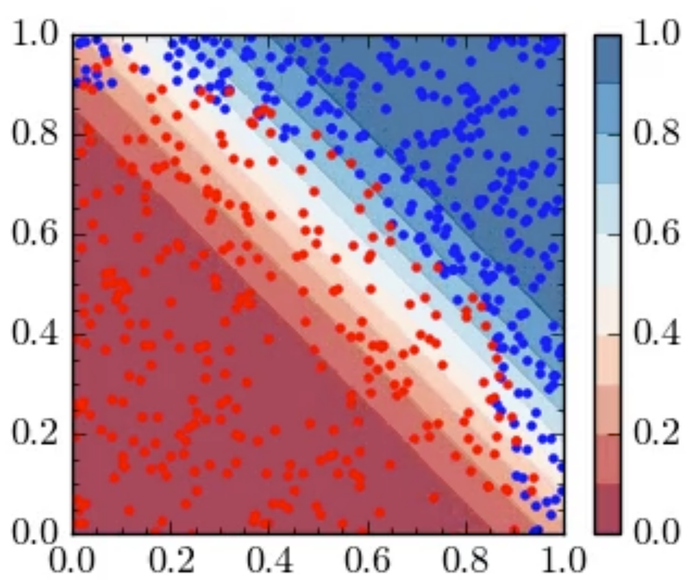


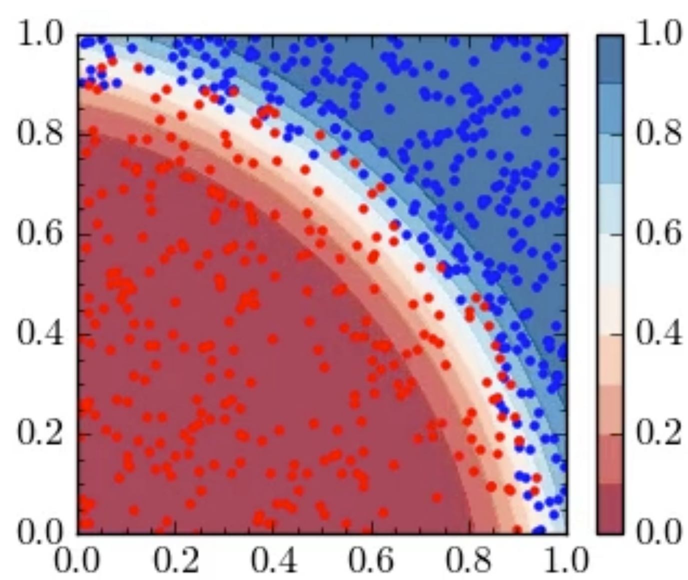


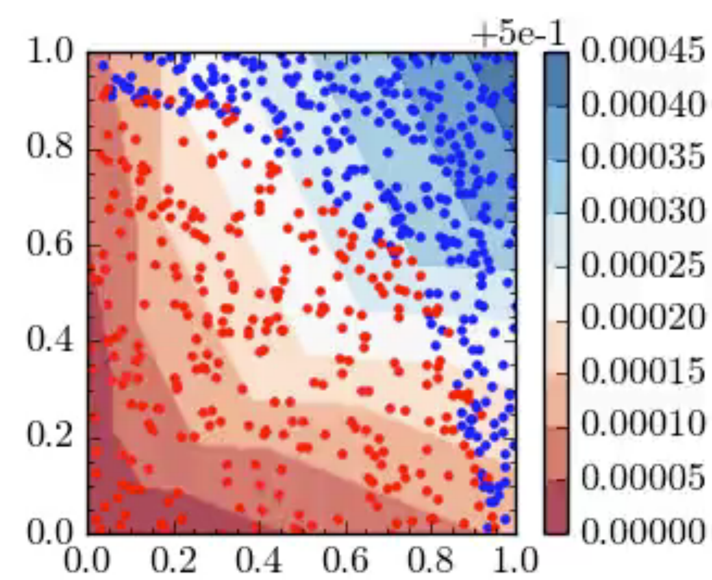
Neural Networks

- Don't add more inputs, let machine find own shape
- Ability to learn 'any' function
- More nodes/hidden layers allows for more complex features



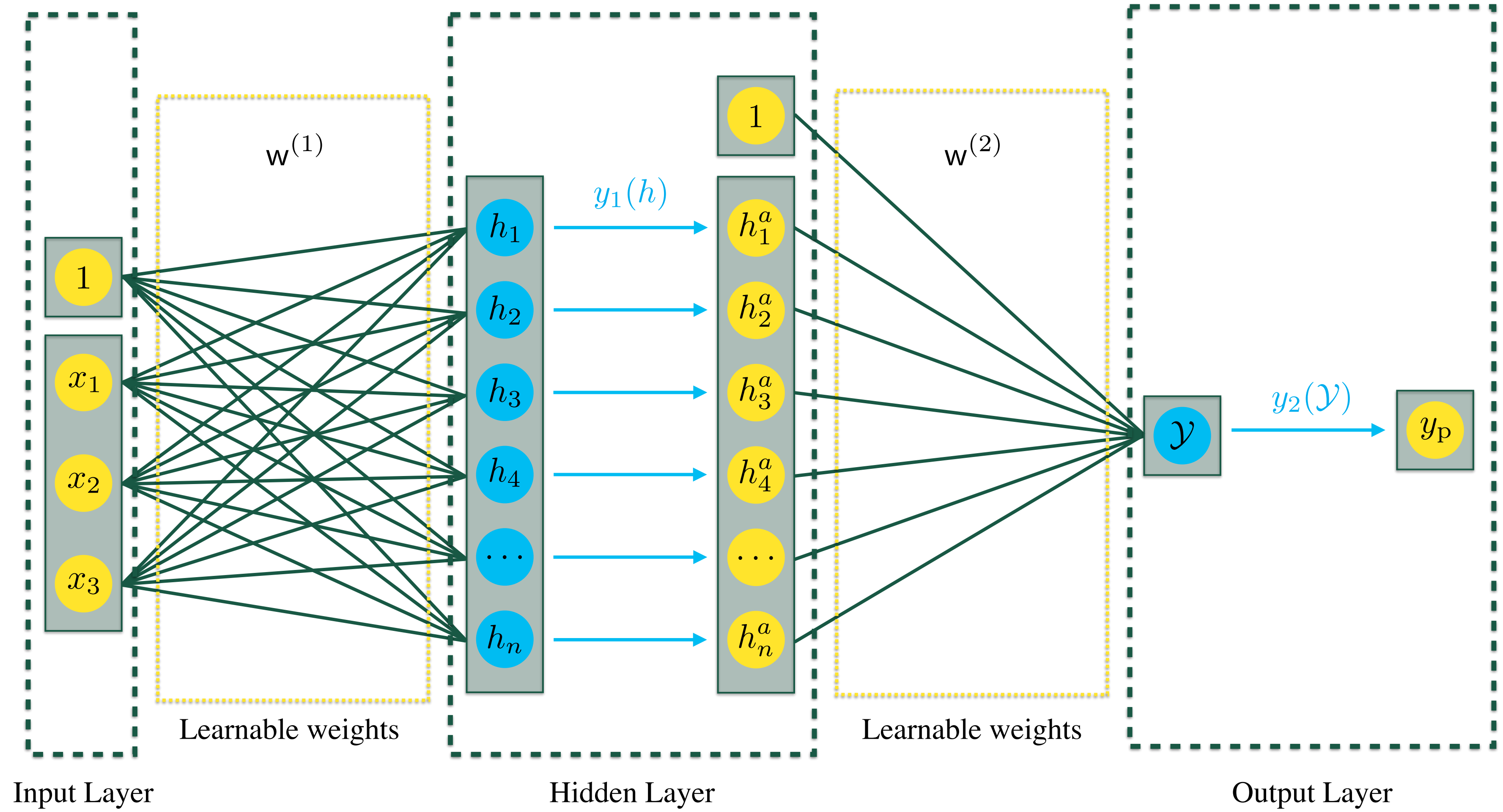


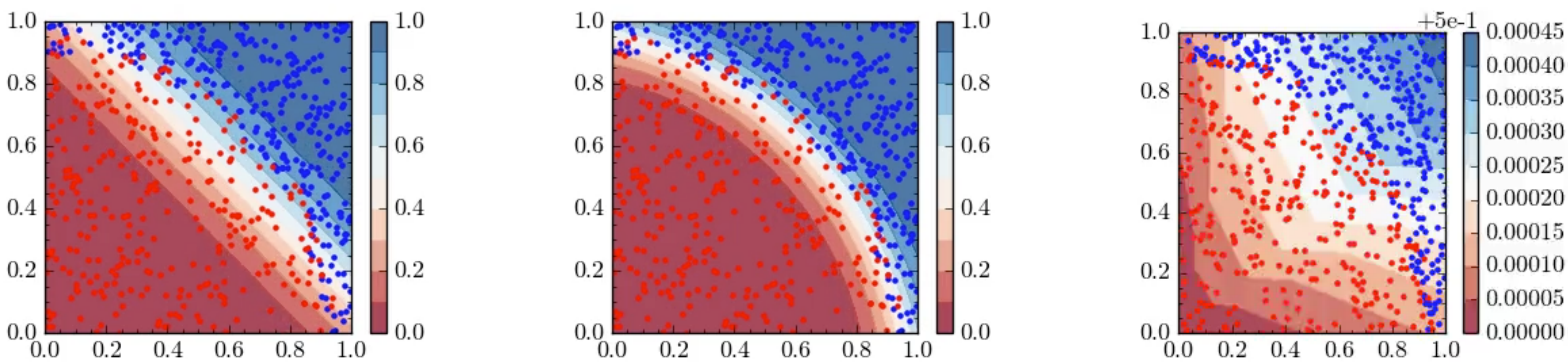




Neural Networks

- Don't add more inputs, let machine find own shape
- Ability to learn 'any' function
- More nodes/hidden layers allows for more complex features

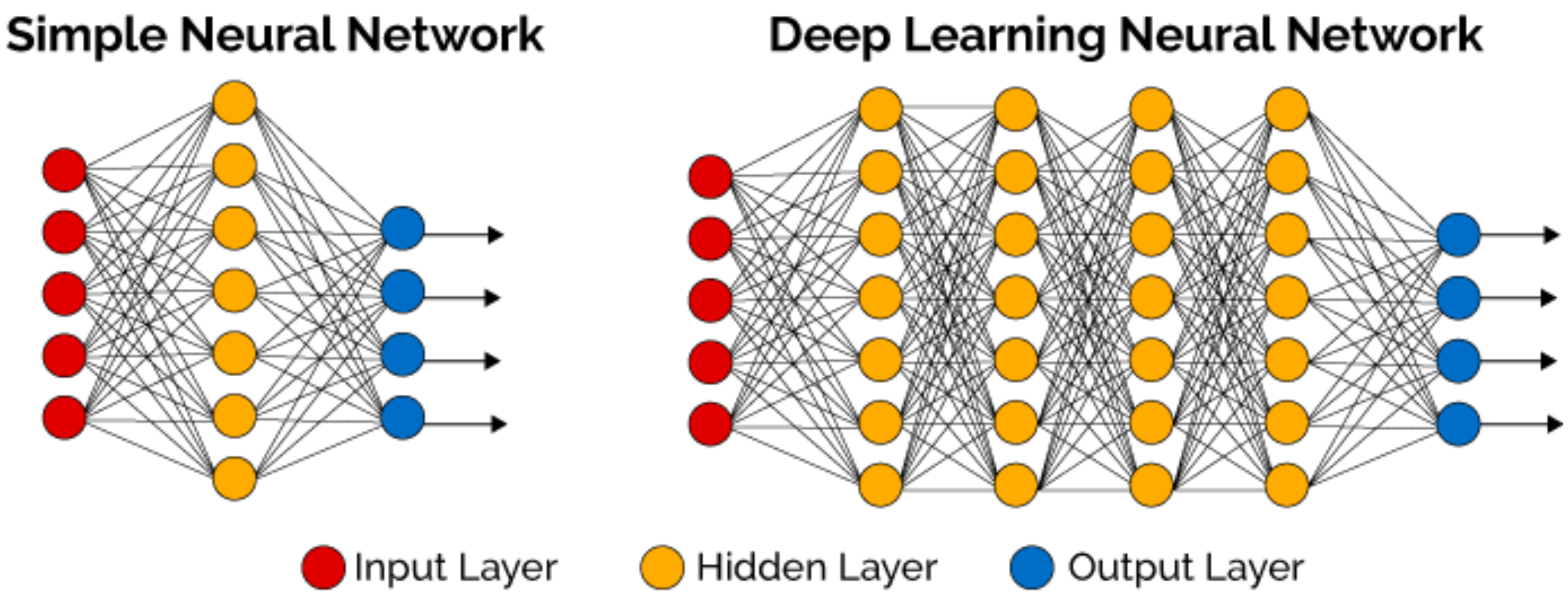




Neural Network Review

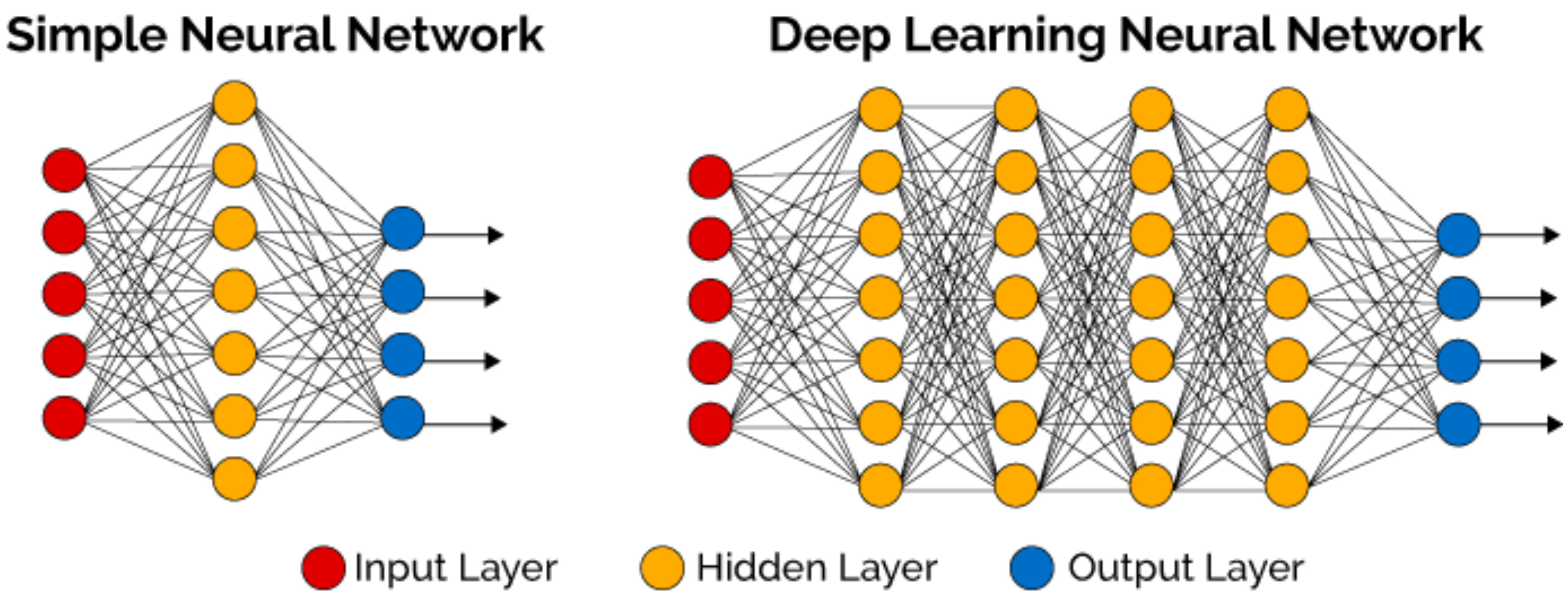
- Neural networks act as universal function fitter
- Deep networks (many hidden layers) allow the network to pick its own features

Deep Learning

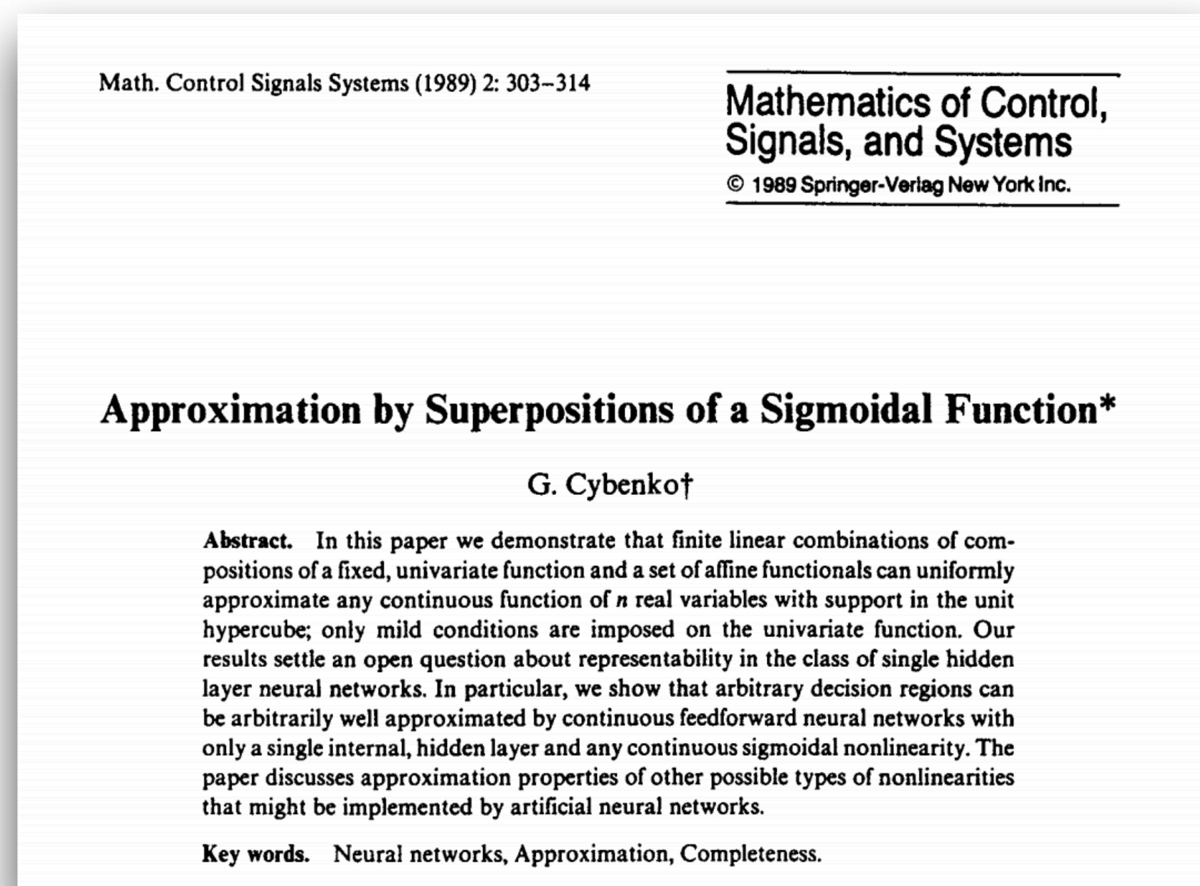


Pic Credit: Xenonstack | Simple Neural Network and Deep Neural Network

Deep Learning



Pic Credit: Xenonstack | Simple Neural Network and Deep Neural Network



Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

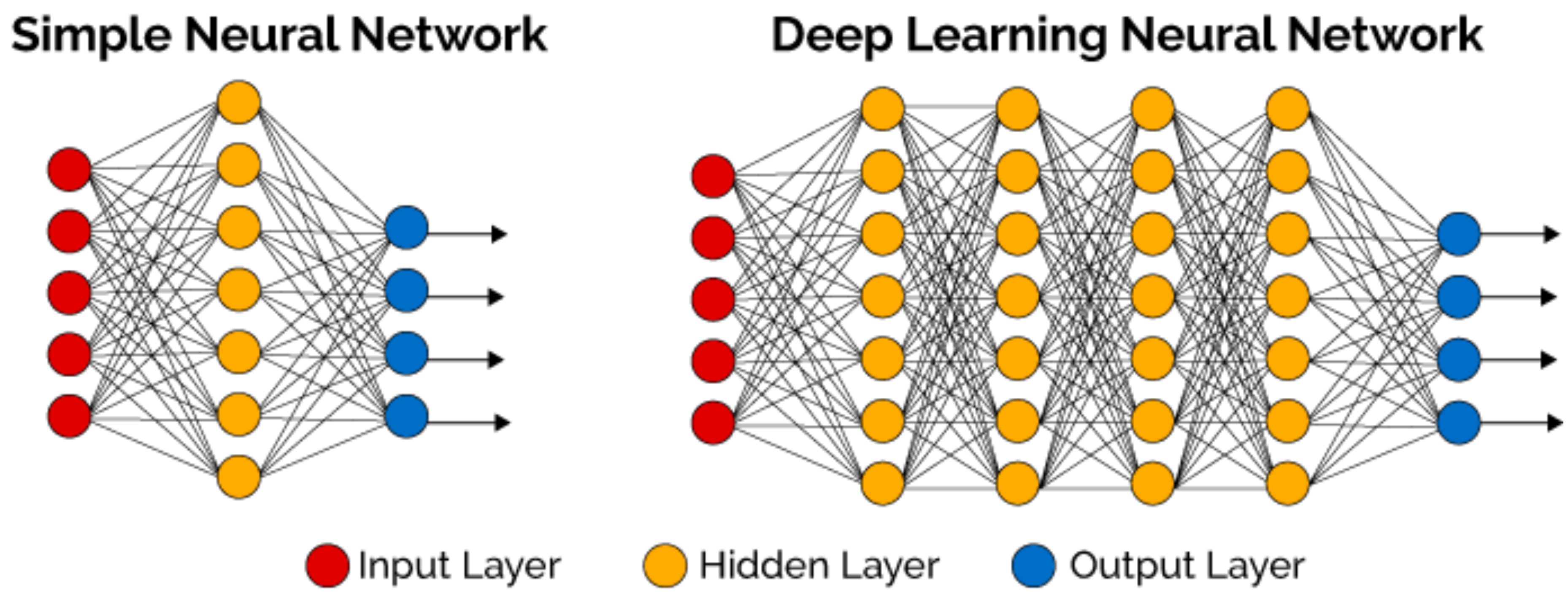
Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of n real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

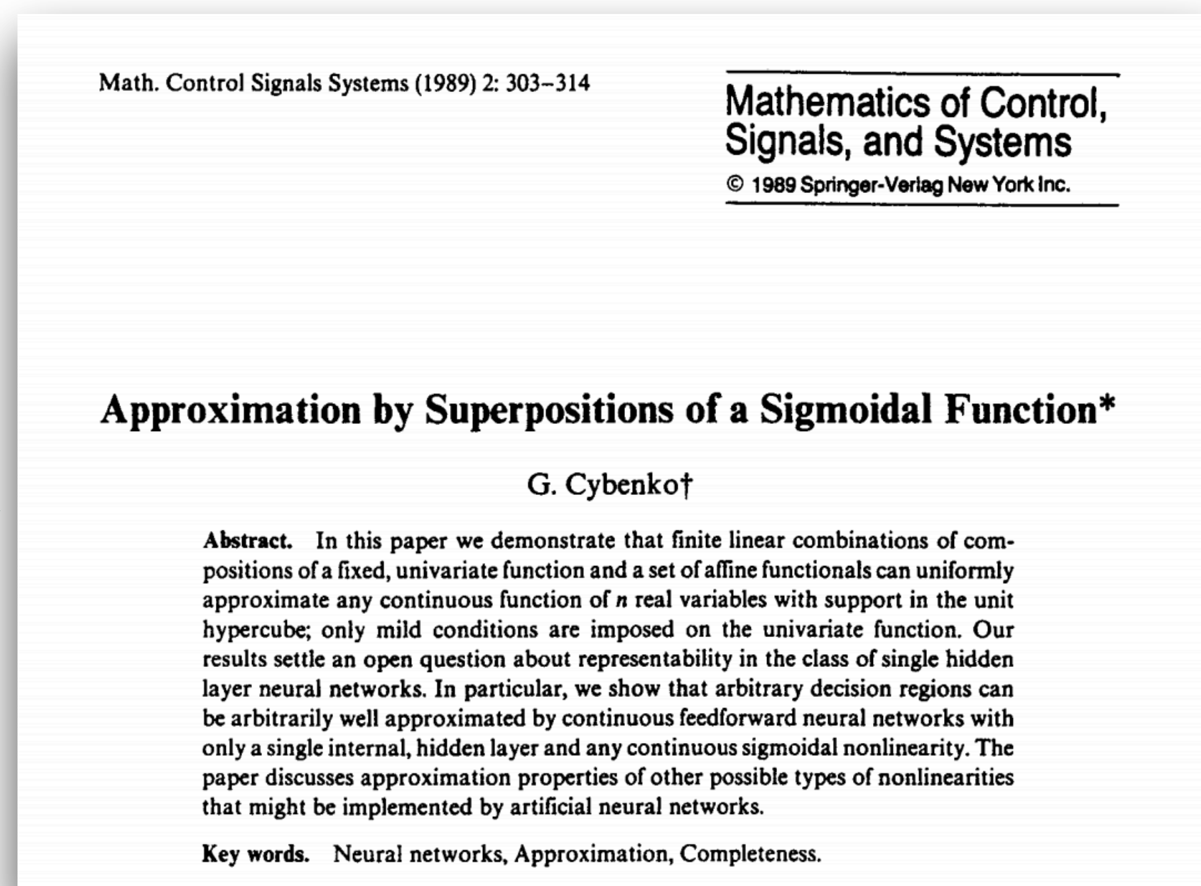
Key words. Neural networks, Approximation, Completeness.

Deep Learning



Pic Credit: Xenonstack | Simple Neural Network and Deep Neural Network

Can fit any function with infinite data and infinite nodes (1 hidden layer)



Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

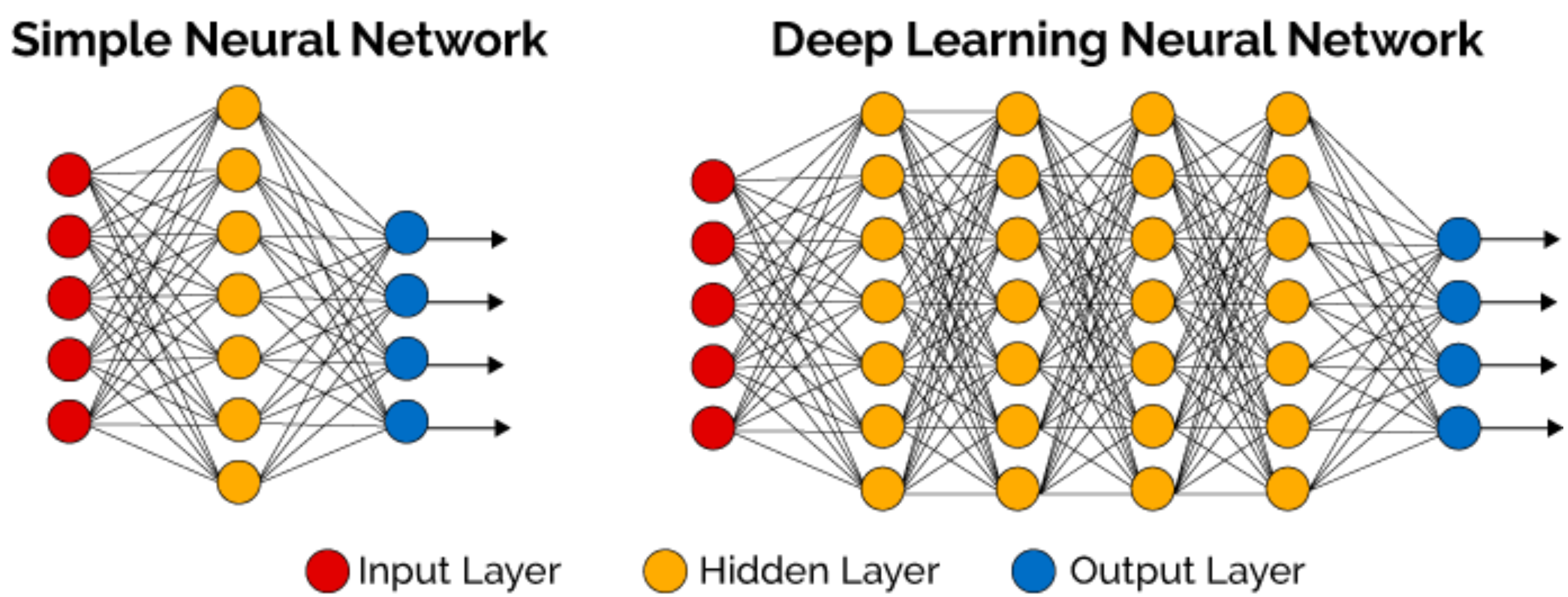
Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of *n* real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

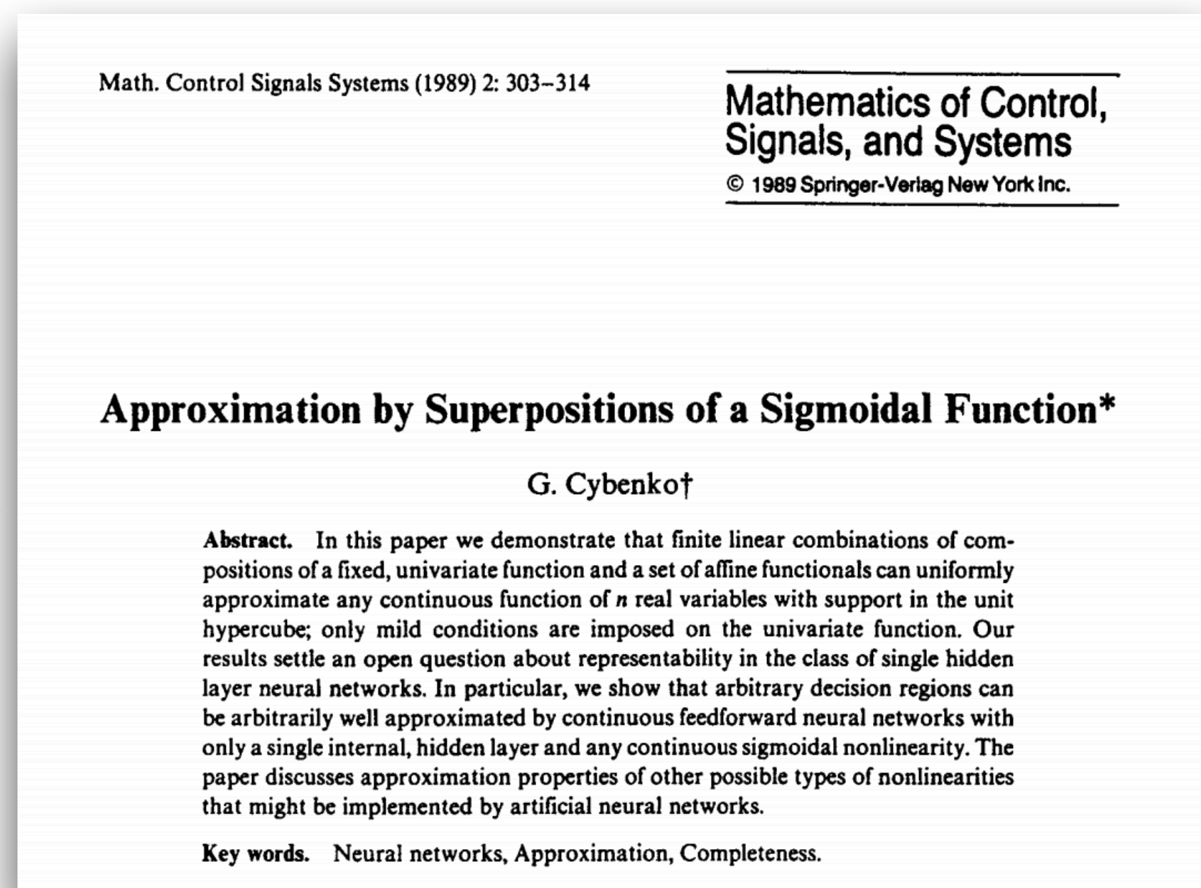
Key words. Neural networks, Approximation, Completeness.

Deep Learning



Pic Credit: Xenonstack | Simple Neural Network and Deep Neural Network

Can fit any function with infinite data and infinite nodes (1 hidden layer)



Math. Control Signals Systems (1989) 2: 303–314

Mathematics of Control, Signals, and Systems

© 1989 Springer-Verlag New York Inc.

Approximation by Superpositions of a Sigmoidal Function*

G. Cybenko†

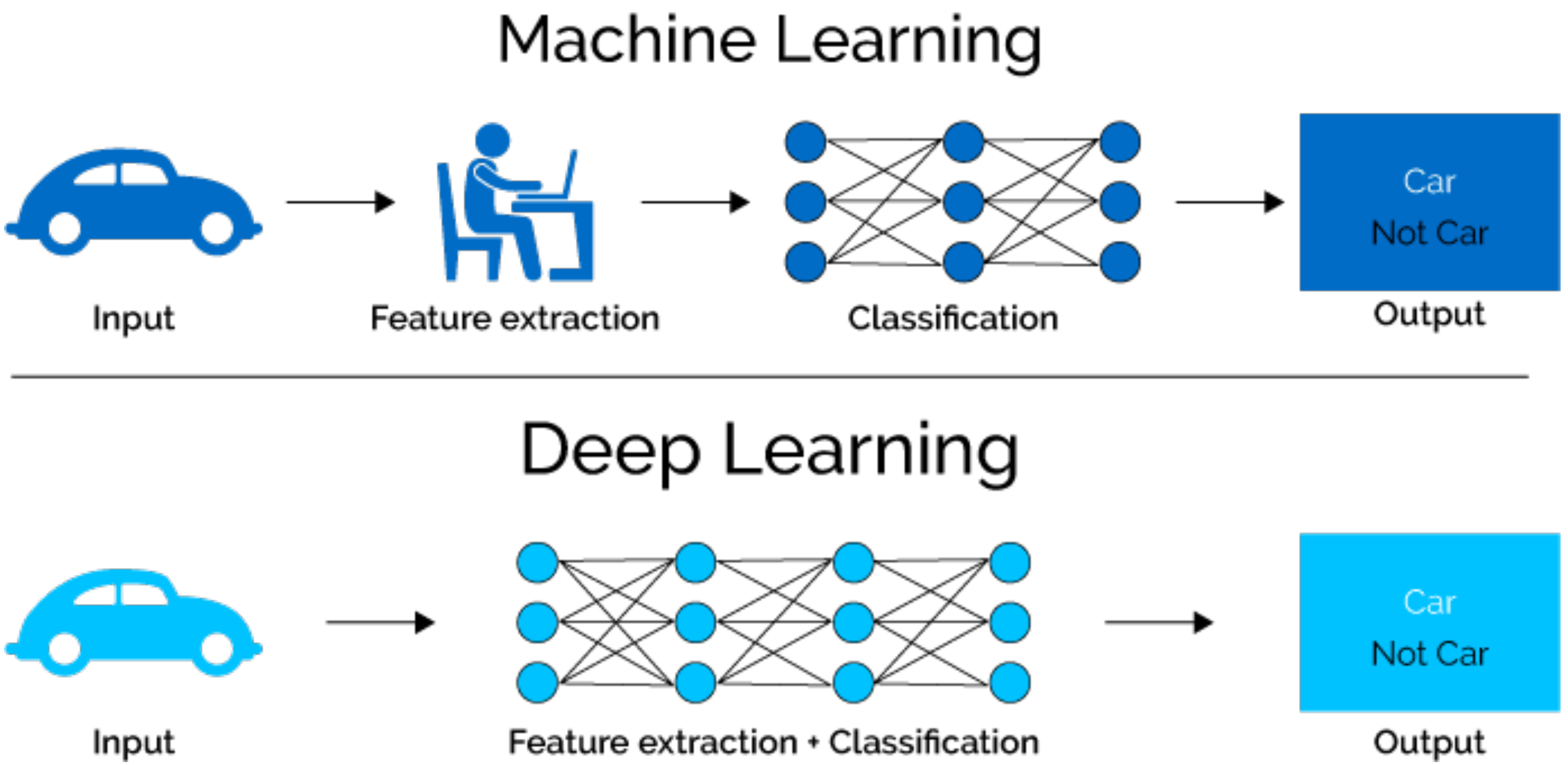
Abstract. In this paper we demonstrate that finite linear combinations of compositions of a fixed, univariate function and a set of affine functionals can uniformly approximate any continuous function of *n* real variables with support in the unit hypercube; only mild conditions are imposed on the univariate function. Our results settle an open question about representability in the class of single hidden layer neural networks. In particular, we show that arbitrary decision regions can be arbitrarily well approximated by continuous feedforward neural networks with only a single internal, hidden layer and any continuous sigmoidal nonlinearity. The paper discusses approximation properties of other possible types of nonlinearities that might be implemented by artificial neural networks.

Key words. Neural networks, Approximation, Completeness.

Going deeper rather than wider learns non-linearities with fewer parameters

Deep Learning

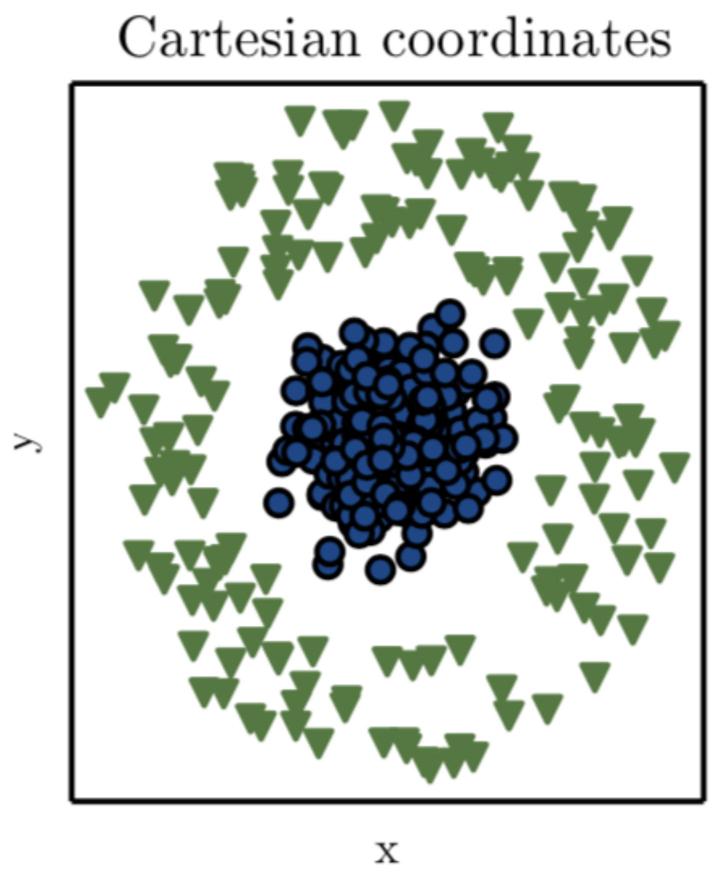
Part of the deep learning revolution is end-to-end learning



Pic Credit: Xenonstack | Machine Learning vs Deep Learning

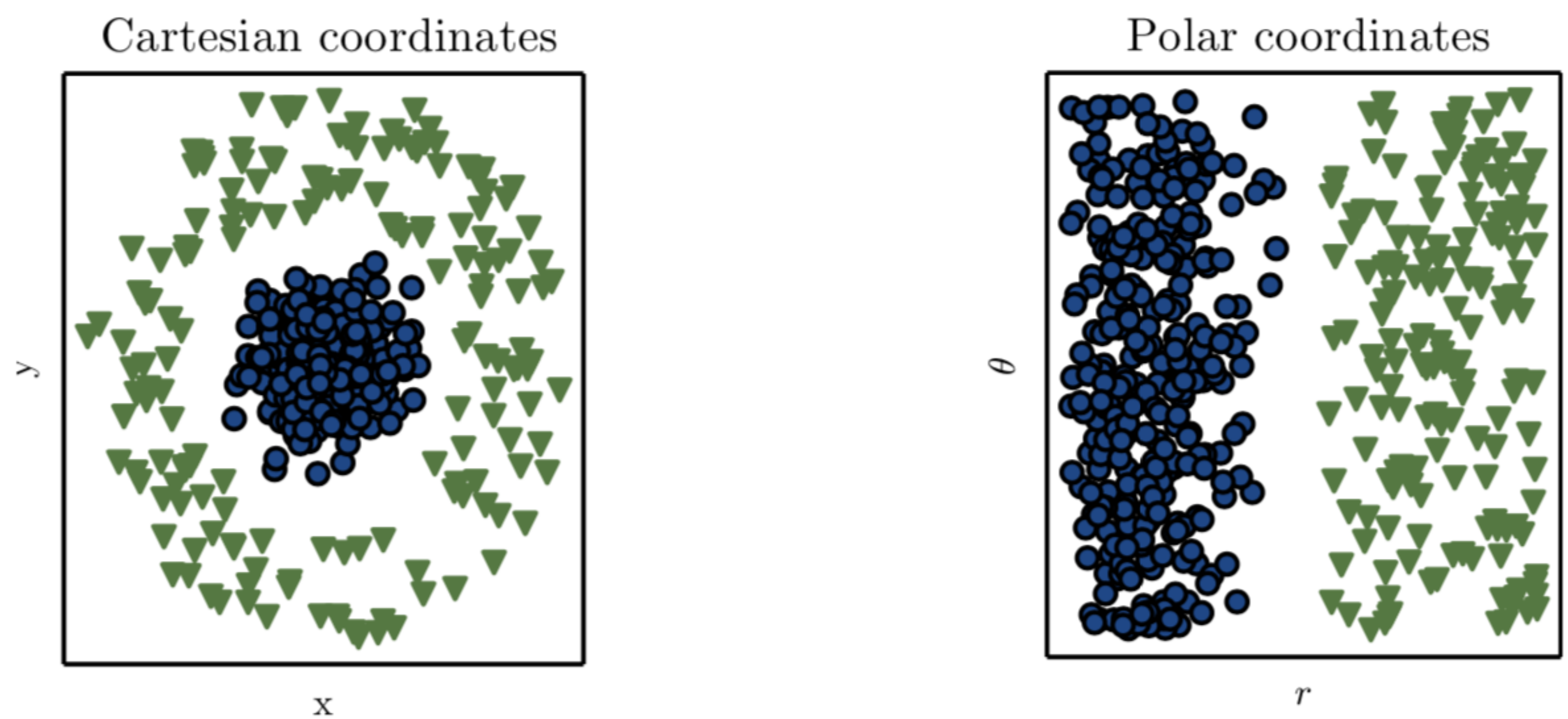
End-To-End Learning

This data is not currently linearly separable



End-To-End Learning

This data is not currently linearly separable



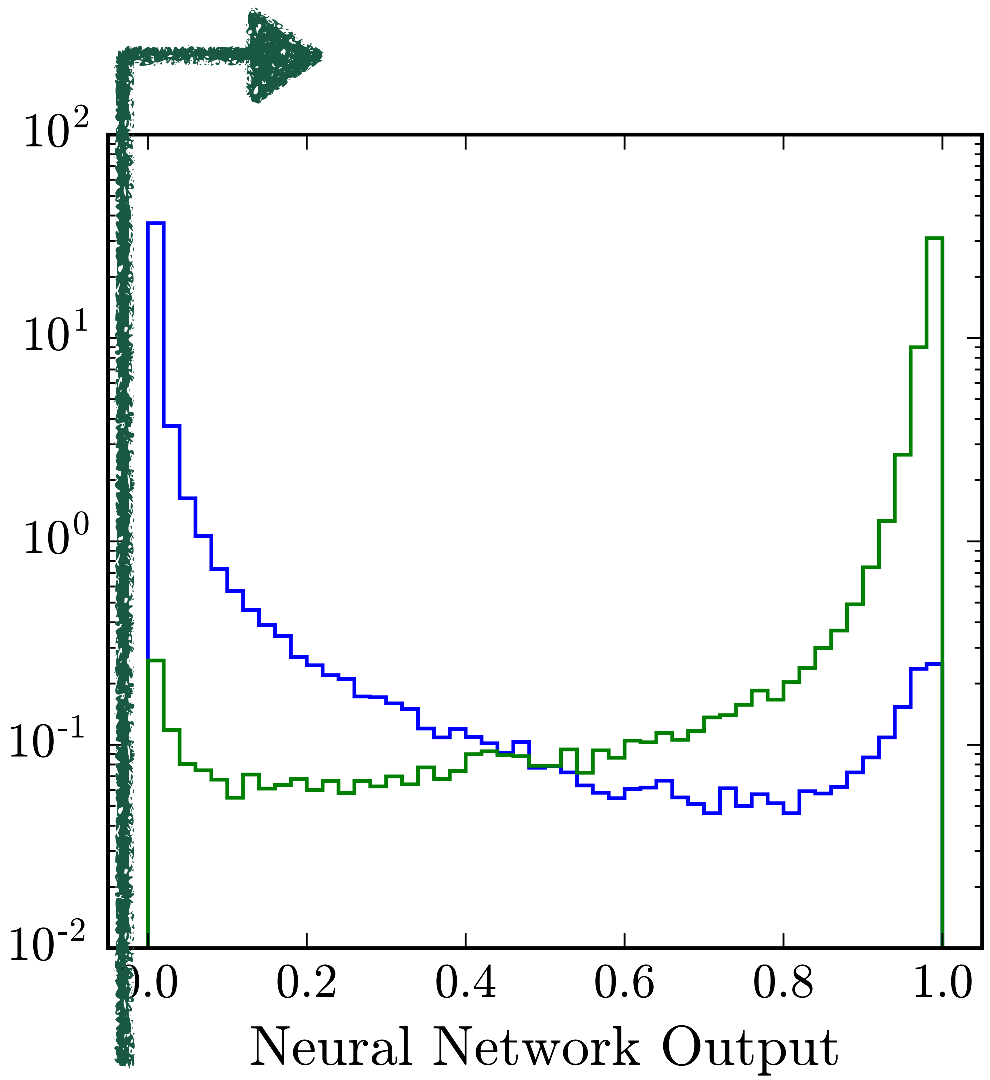
- A simple coordinate transformation makes this a linear separable problem
- Hard to come up with transformations in high dimensions
- Use physics insights for collider variables

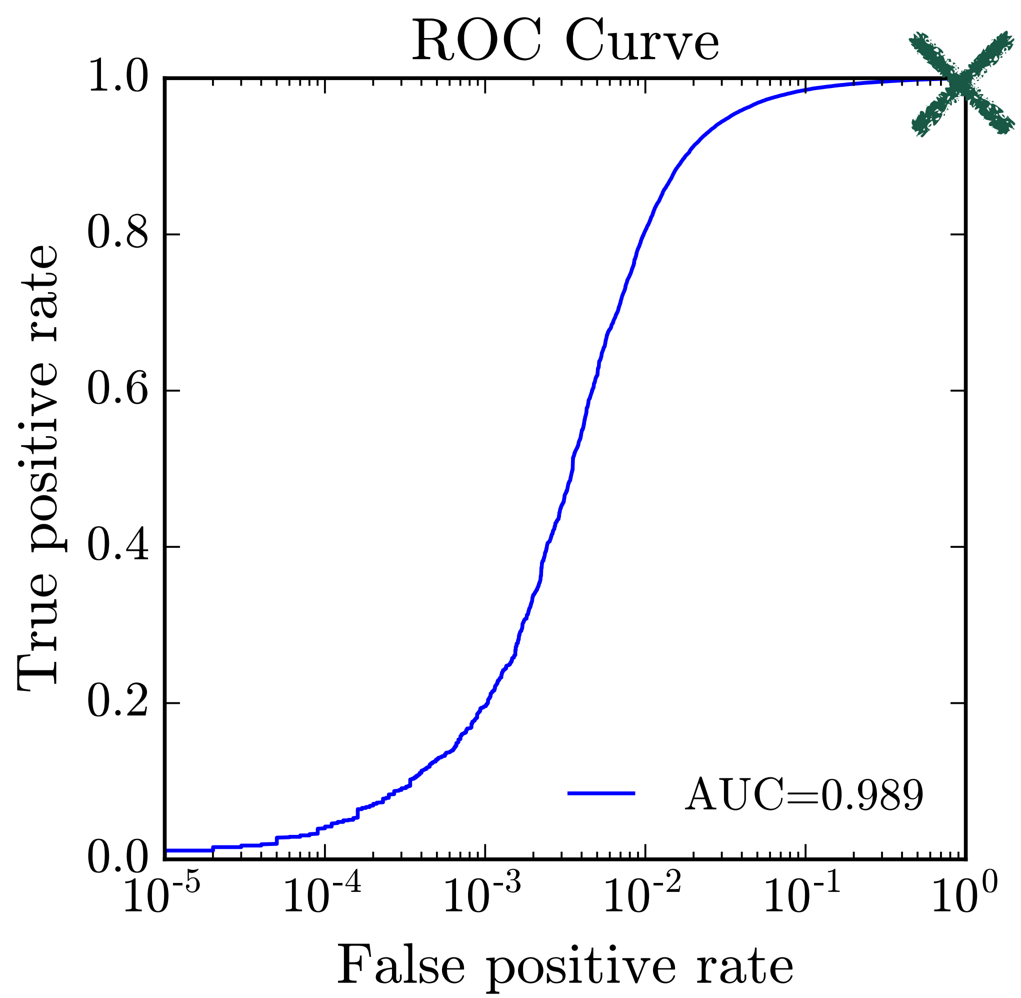
Quick interlude

- Going to get into collider machine learning soon
- Any questions so far?
- Examine metric to compare classifiers

How to quantify a classifier

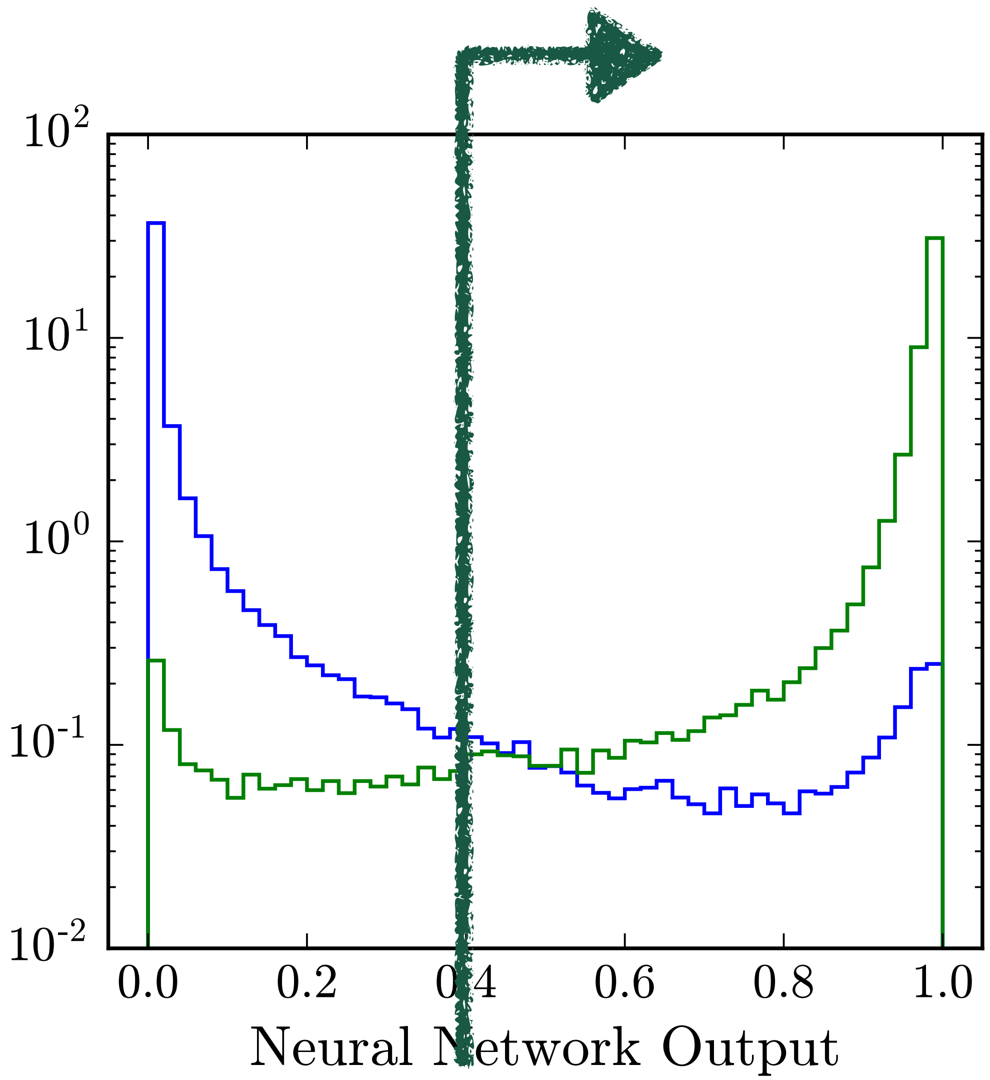
If we want to compare the performance of a classifier, one common option is the Receiver Operating Characteristic (ROC) Curve

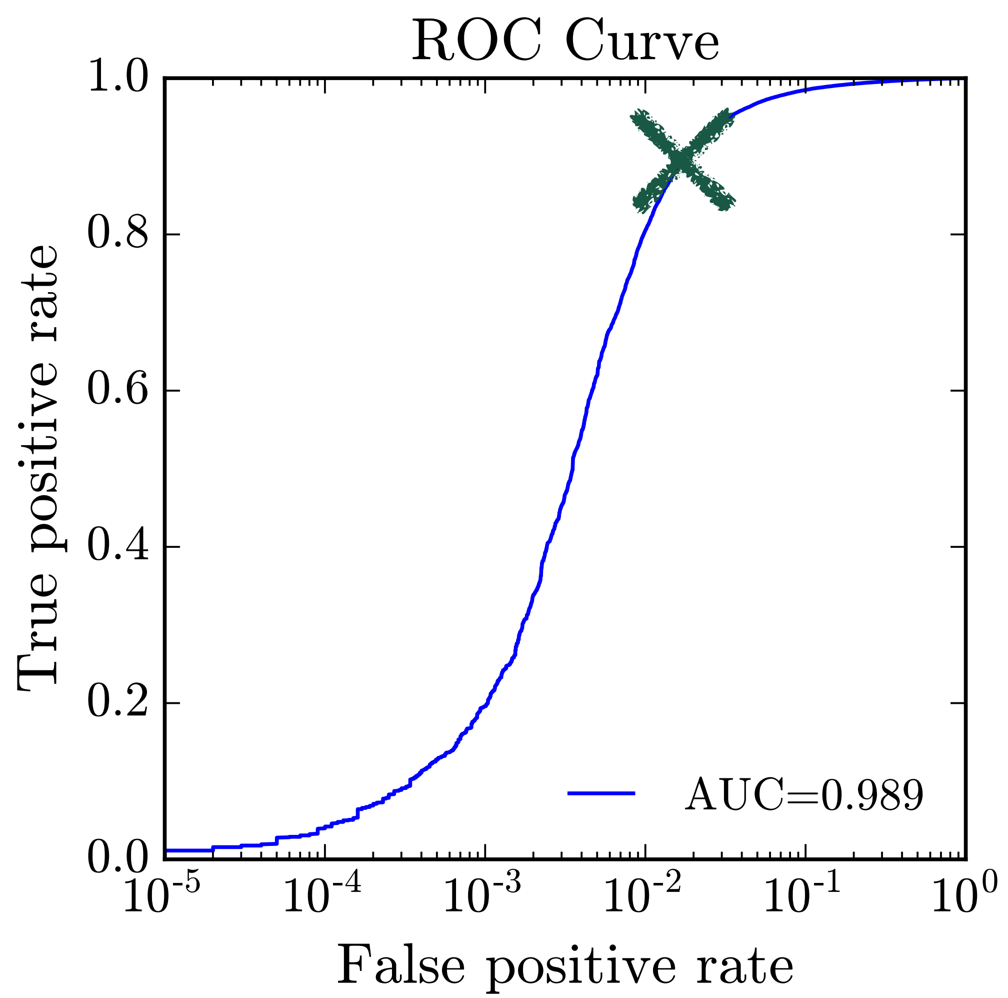




How to quantify a classifier

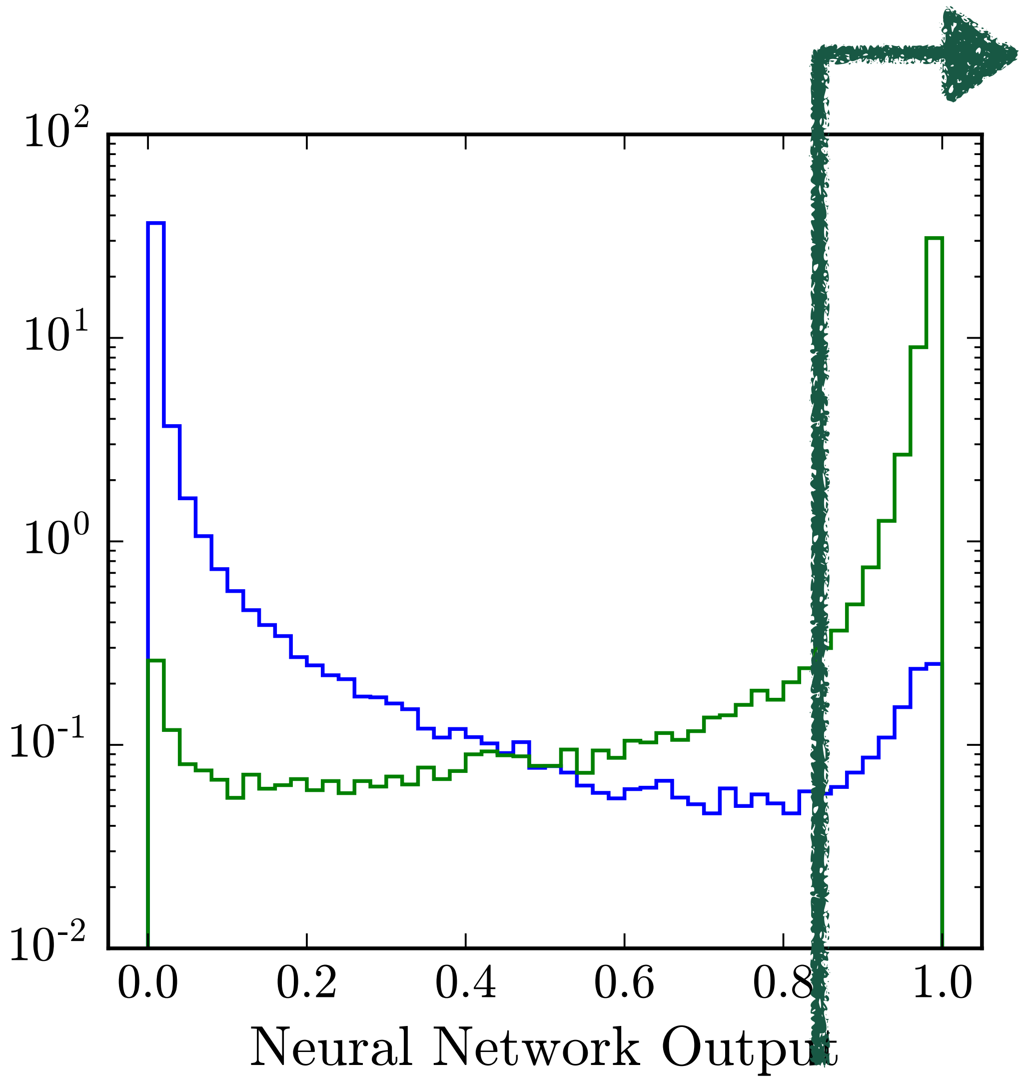
If we want to compare the performance of a classifier, one common option is the Receiver Operating Characteristic (ROC) Curve

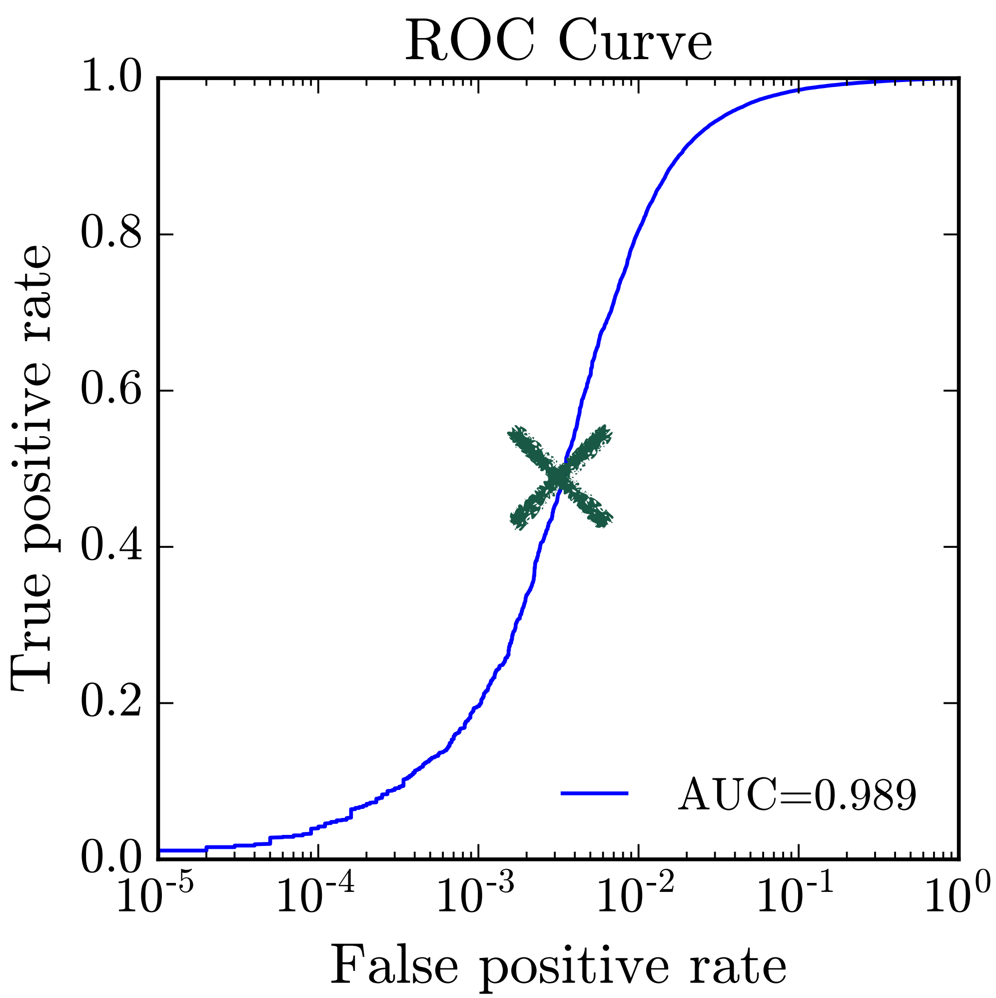




How to quantify a classifier

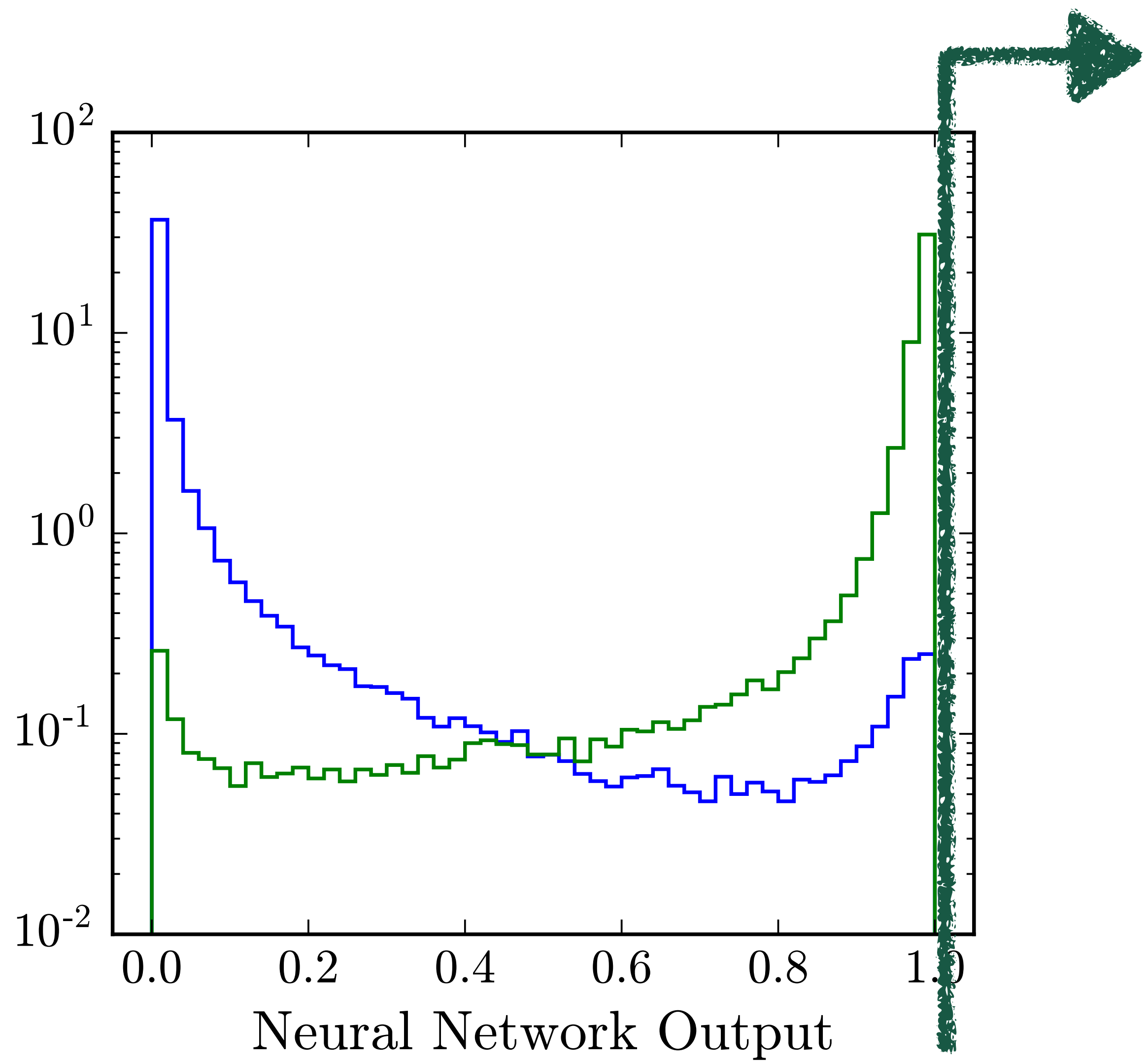
If we want to compare the performance of a classifier, one common option is the Receiver Operating Characteristic (ROC) Curve

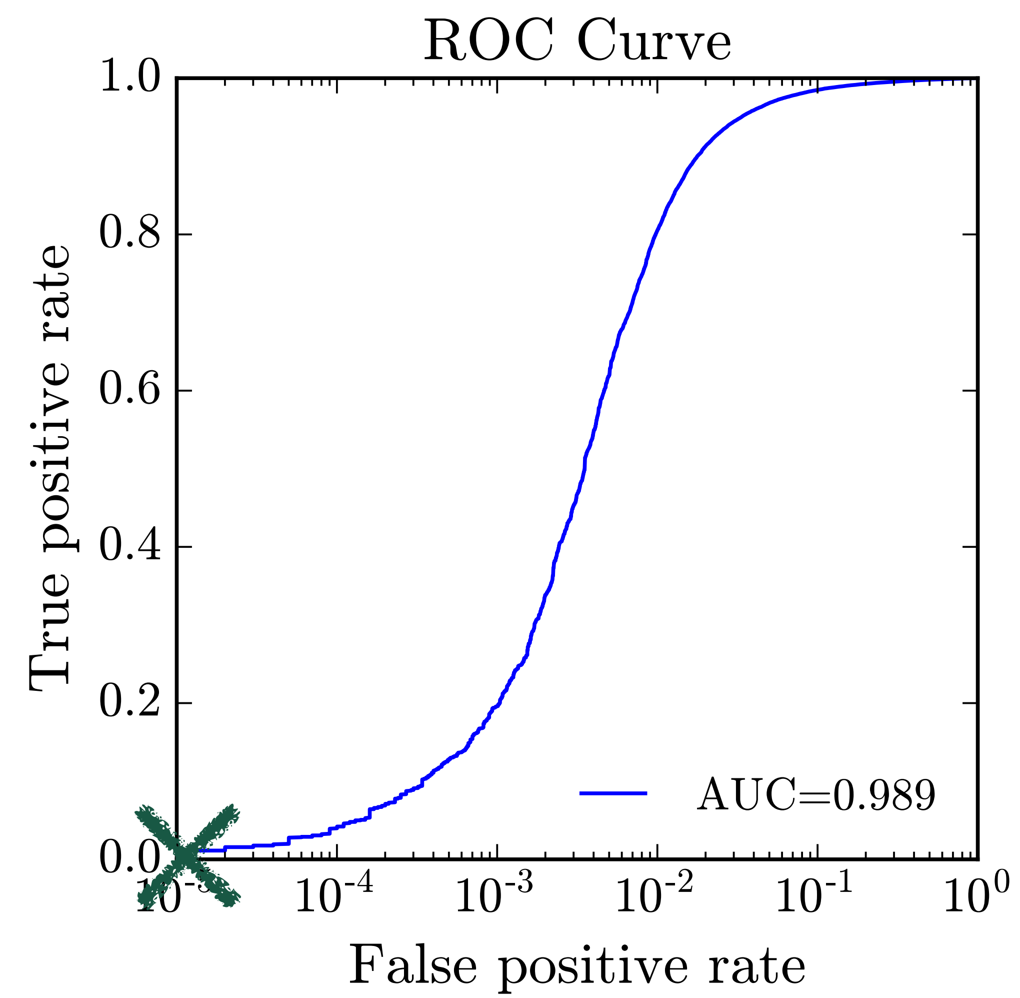




How to quantify a classifier

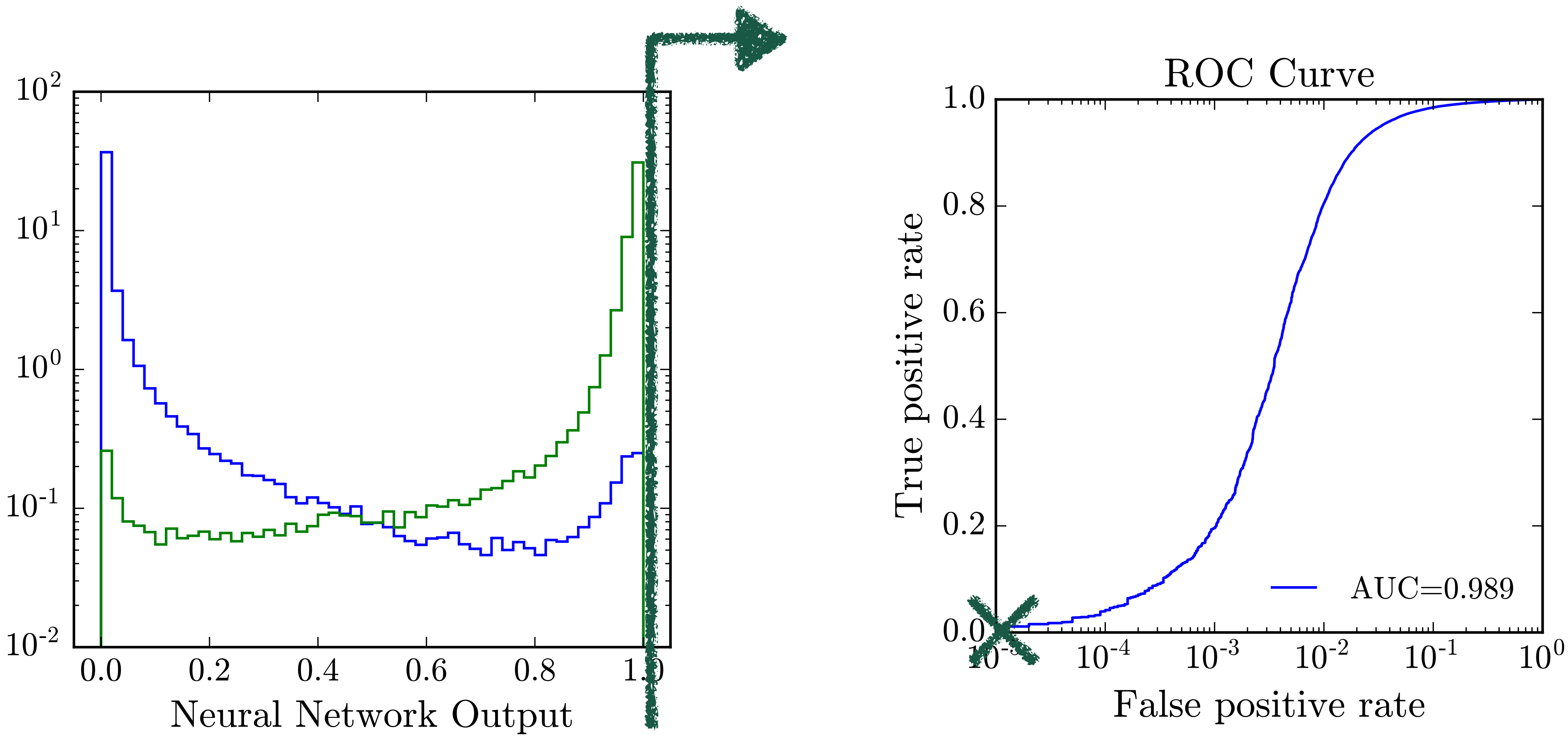
If we want to compare the performance of a classifier, one common option is the Receiver Operating Characteristic (ROC) Curve





How to quantify a classifier

If we want to compare the performance of a classifier, one common option is the Receiver Operating Characteristic (ROC) Curve



AUC = 1.0 is perfect, this is not attainable for most problems

Machine learning particle physics

At the LHC

Can identify and measure photons, electrons, muons, and things made of quarks

Neutrinos (and some BSM particles) escape detection

Beams travel in ±z direction, no momentum in (x, y) plane

Machine learning particle physics

At the LHC

Can identify and measure photons, electrons, muons, and things made of quarks

Energy and momentum vector

Neutrinos (and some BSM particles) escape detection

Beams travel in ±z direction, no momentum in (x, y) plane

Machine learning particle physics

At the LHC

Can identify and measure photons, electrons, muons, and things made of quarks

Energy and momentum vector

jets (b-jets)

Neutrinos (and some BSM particles) escape detection

Beams travel in ±z direction, no momentum in (x, y) plane

Machine learning particle physics

At the LHC

Can identify and measure photons, electrons, muons, and things made of quarks

Energy and momentum vector

jets (b-jets)

Neutrinos (and some BSM particles) escape detection

Beams travel in ±z direction, no momentum in (x, y) plane

Missing momentum in (x, y) plane

Machine learning particle physics

At the LHC

Can identify and measure photons, electrons, muons, and things made of quarks

Energy and momentum vector

jets (b-jets)

Neutrinos (and some BSM particles) escape detection

Beams travel in ±z direction, no momentum in (x, y) plane

Which heavy particle decayed to the final state particles?

Missing momentum in (x, y) plane

Deep learning in HEP

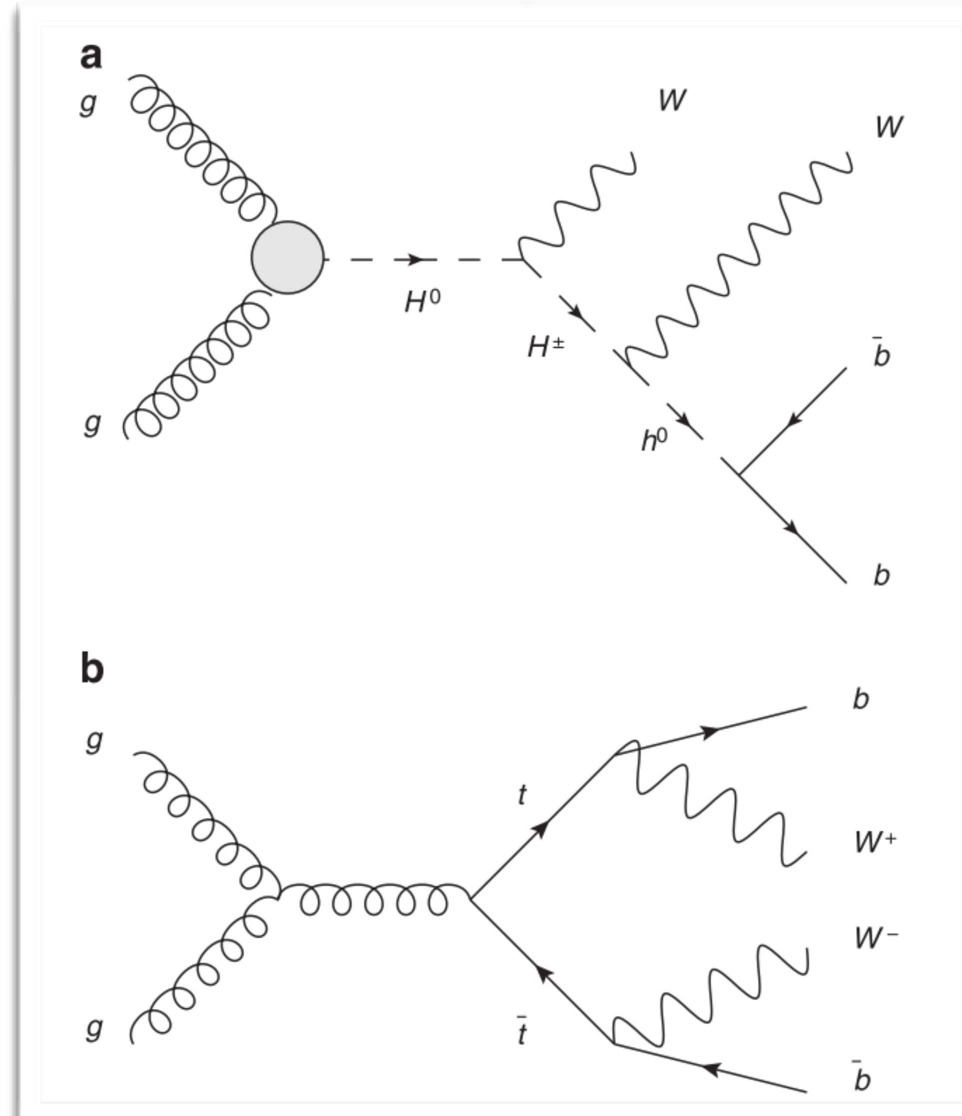
ARTICLE

Received 19 Feb 2014 | Accepted 4 Jun 2014 | Published 2 Jul 2014 DOI: 10.1038/ncomms5308

Searching for exotic particles in high-energy physics with deep learning

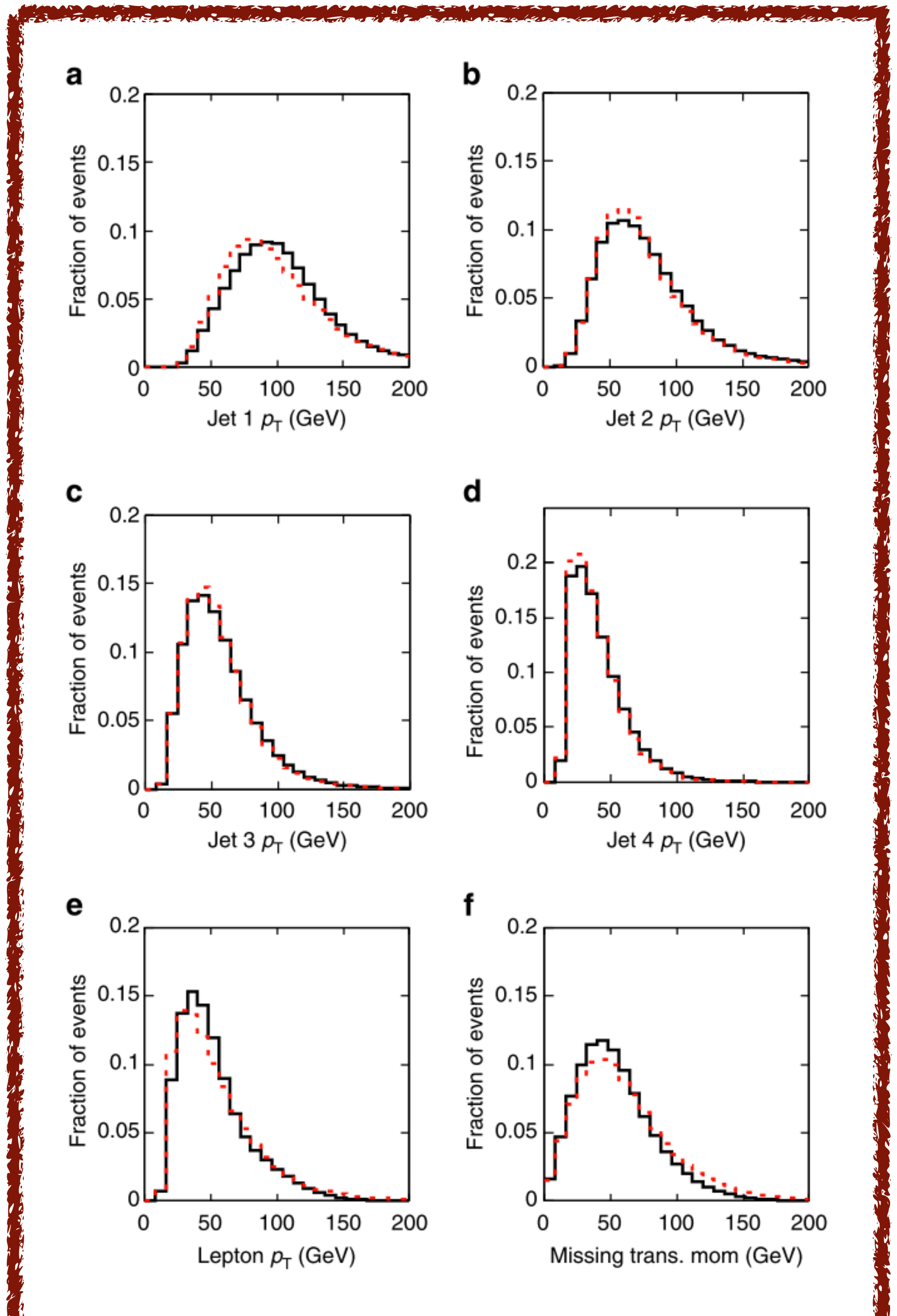
[1402.4735]

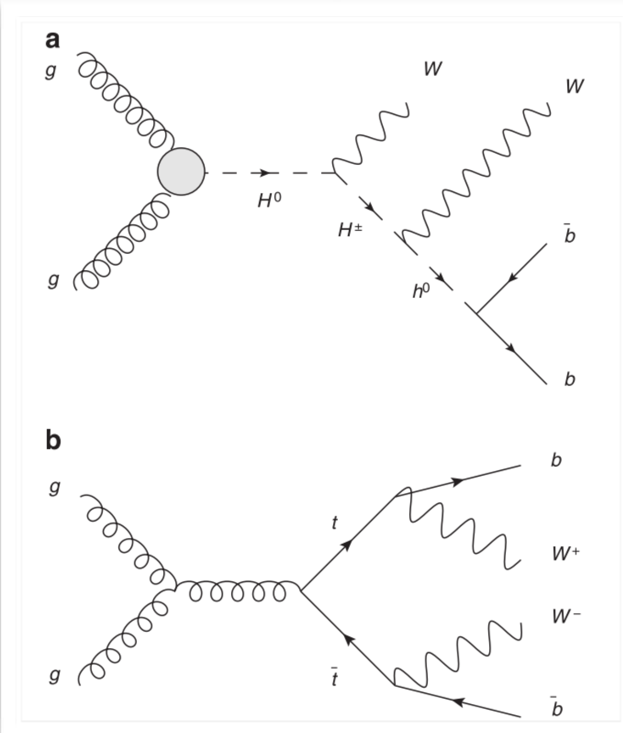
P. Baldi[1], P. Sadowski[1] & D. Whiteson[2]



- One of first papers to show deep learning ***outperforming*** standard techniques in HEP
- Compares shallow and deep networks on raw and high-level features

Deep learning in HEP

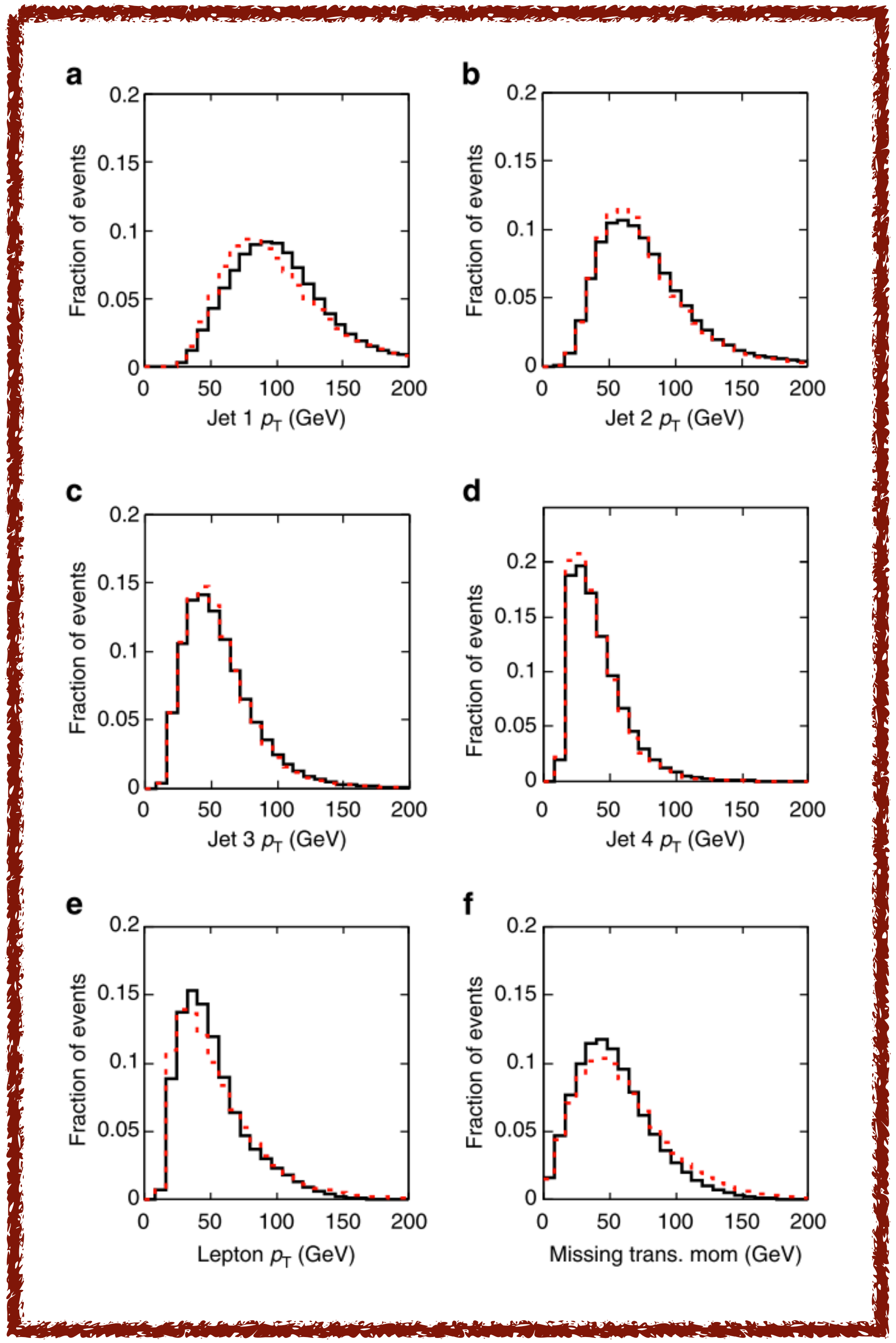


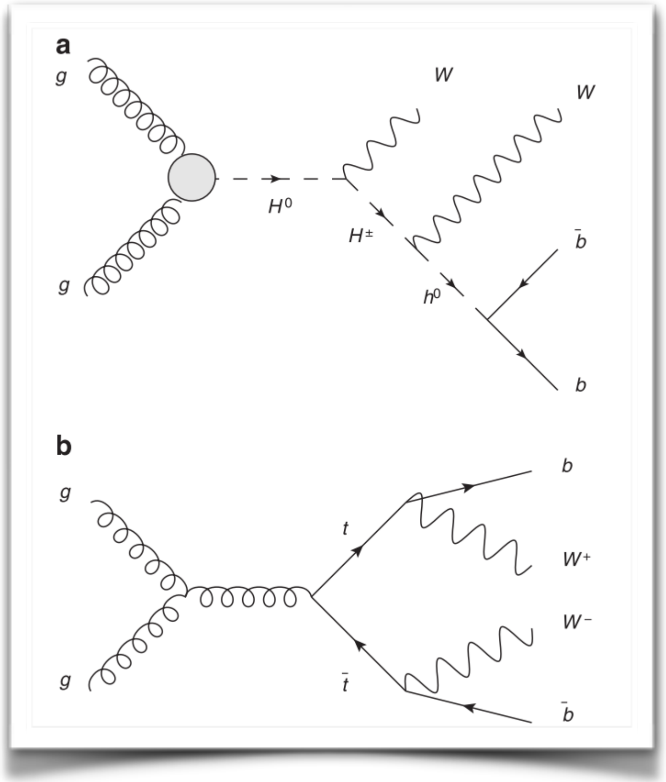


21 raw features for semi-leptonic channel

Not much separation in individual features

Deep learning in HEP

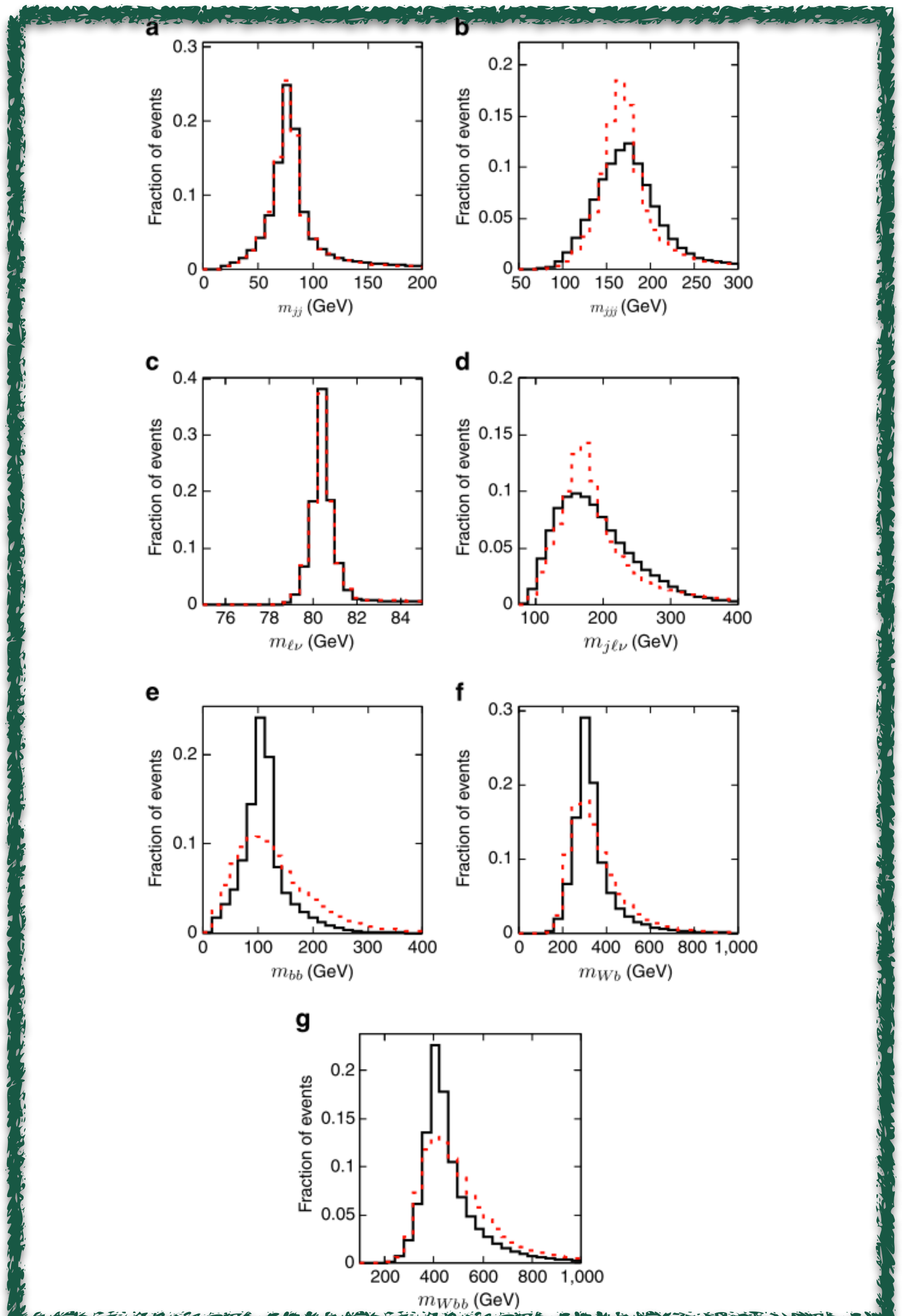




Invariant masses of intermediate sates

m_{jj} m_{jjj} $m_{\ell\nu}$ $m_{j\ell\nu}$

m_{bb} m_{Wb} m_{Wbb}



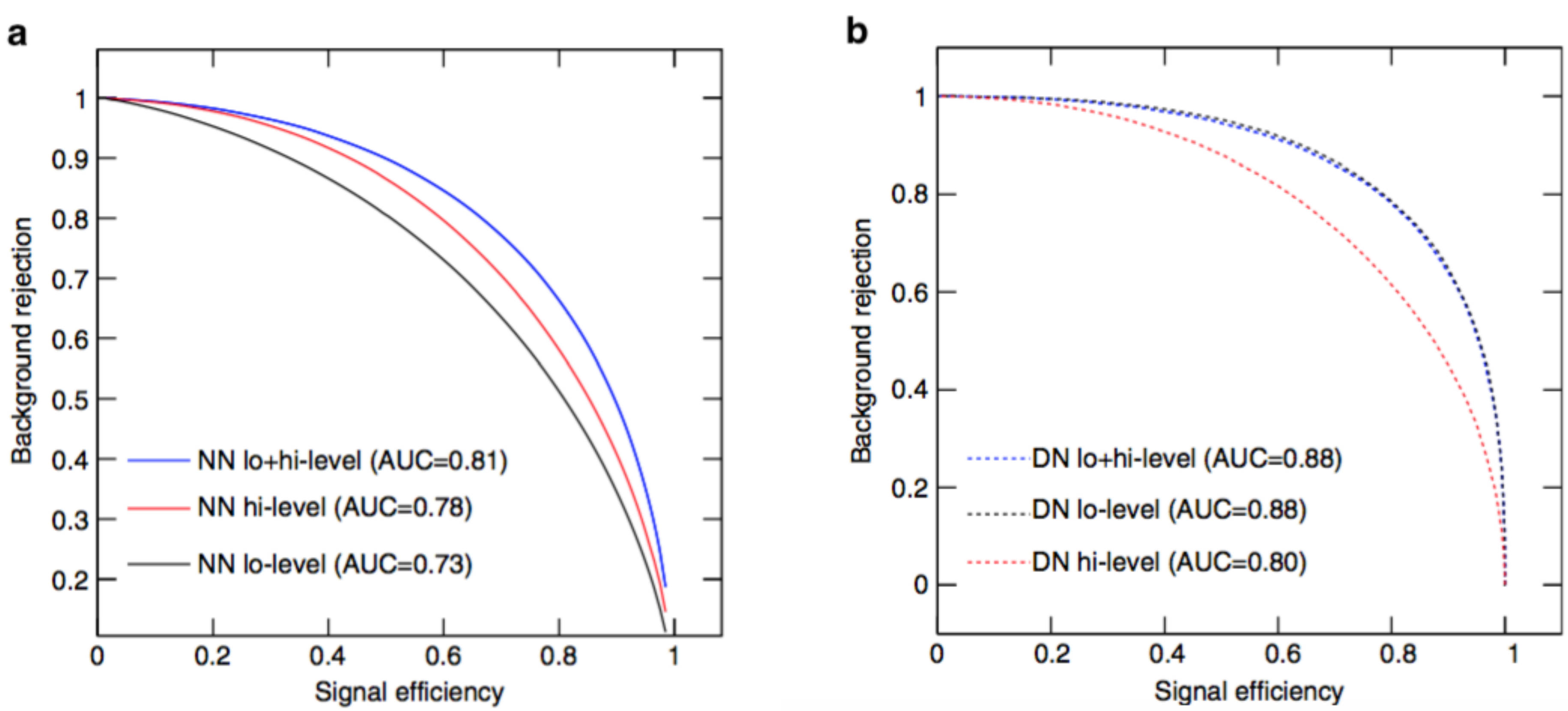
Deep learning in HEP

11 million training examples
1 hidden layer shallow network
5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 ($<$0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and 1,000 ± 50 background events.



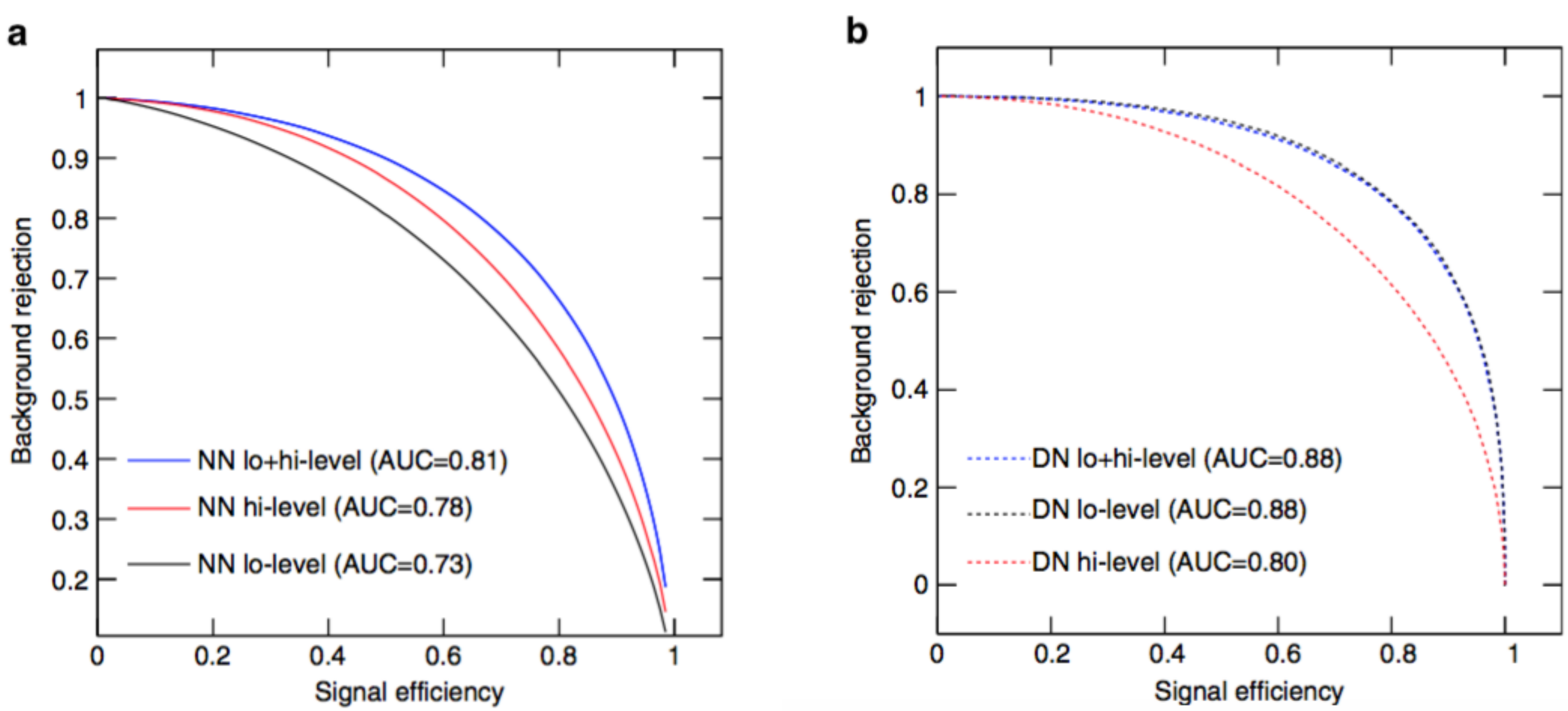
Deep learning in HEP

11 million training examples
1 hidden layer shallow network
5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and $1{,}000 \pm 50$ background events.



Deep learning in HEP

11 million training examples

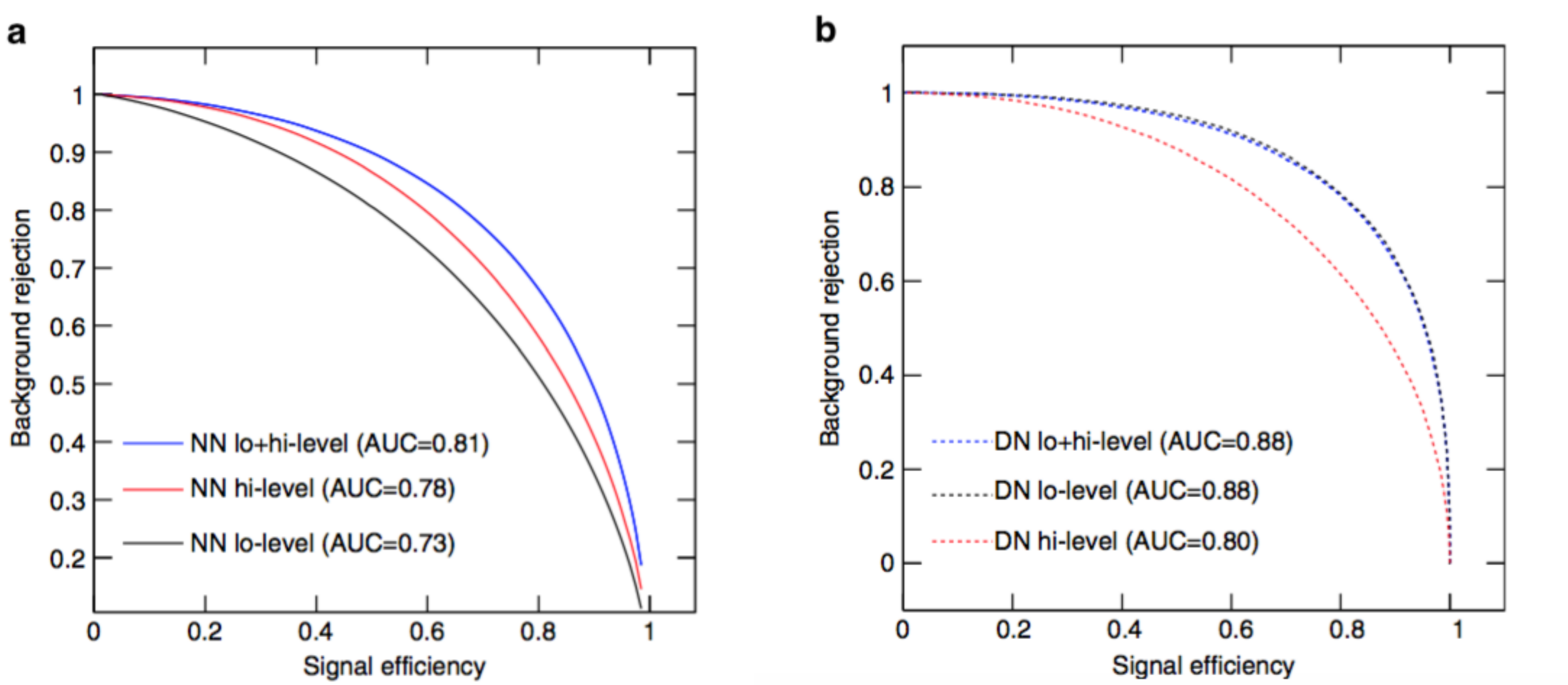
1 hidden layer shallow network

5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and $1,000 \pm 50$ background events.



Deep learning in HEP

11 million training examples

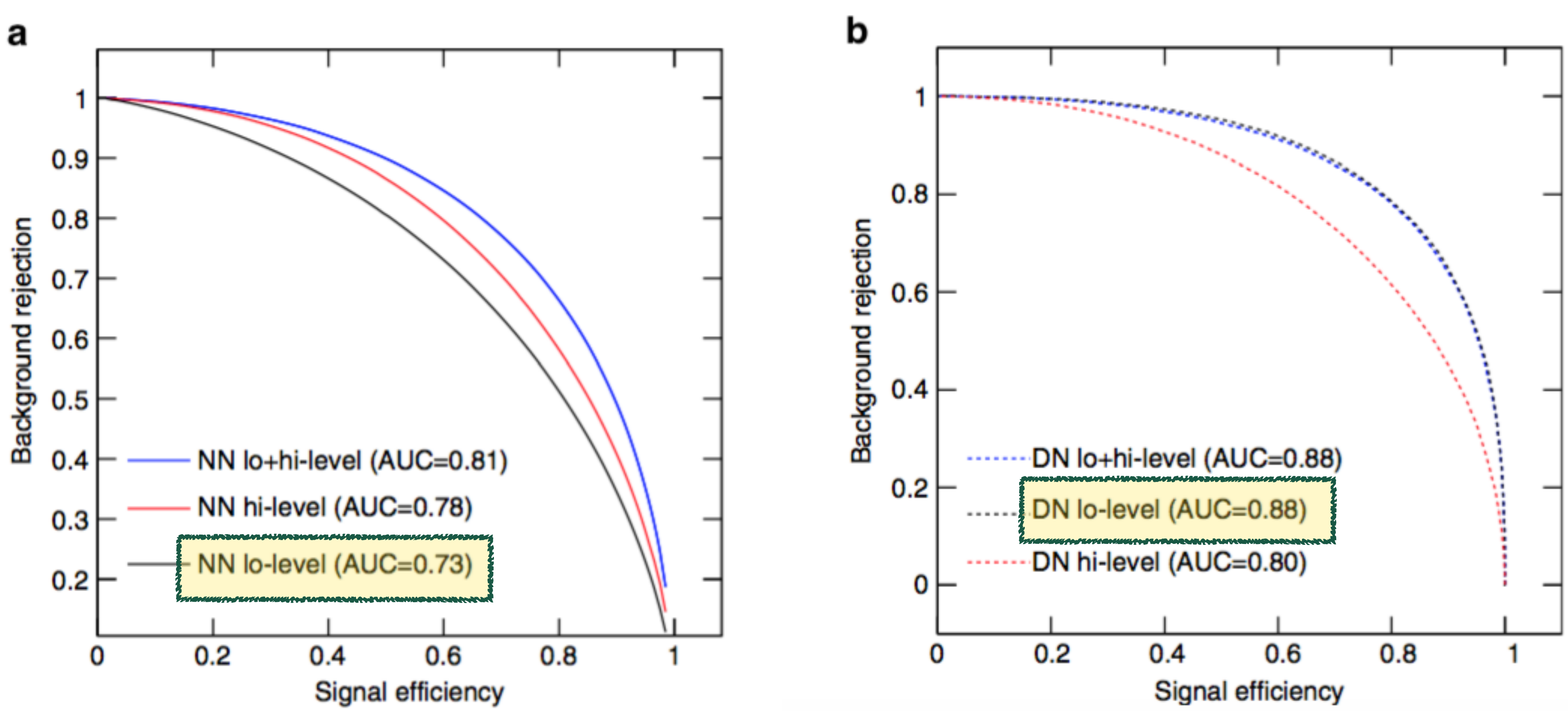
1 hidden layer shallow network

5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and $1,000 \pm 50$ background events.



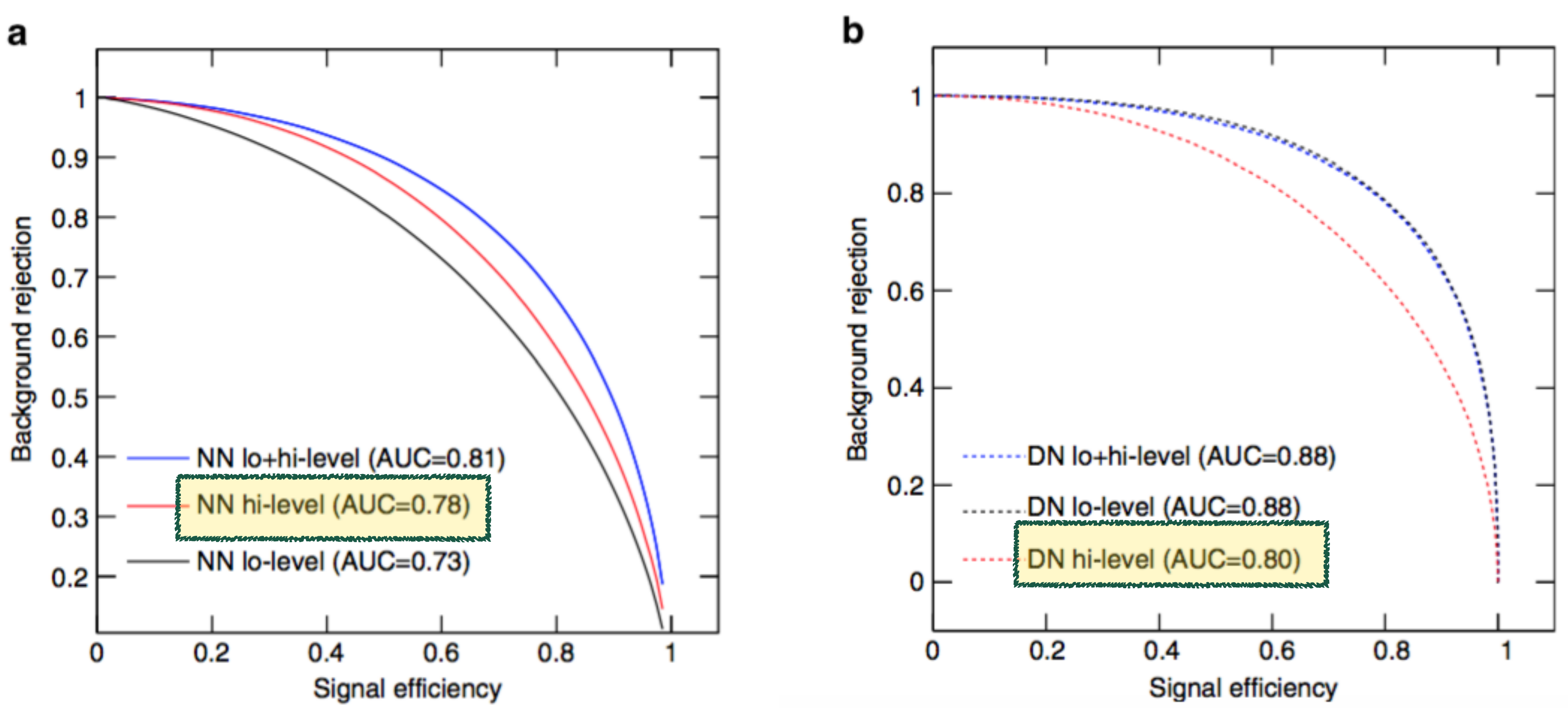
Deep learning in HEP

11 million training examples
1 hidden layer shallow network
5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and $1{,}000 \pm 50$ background events.



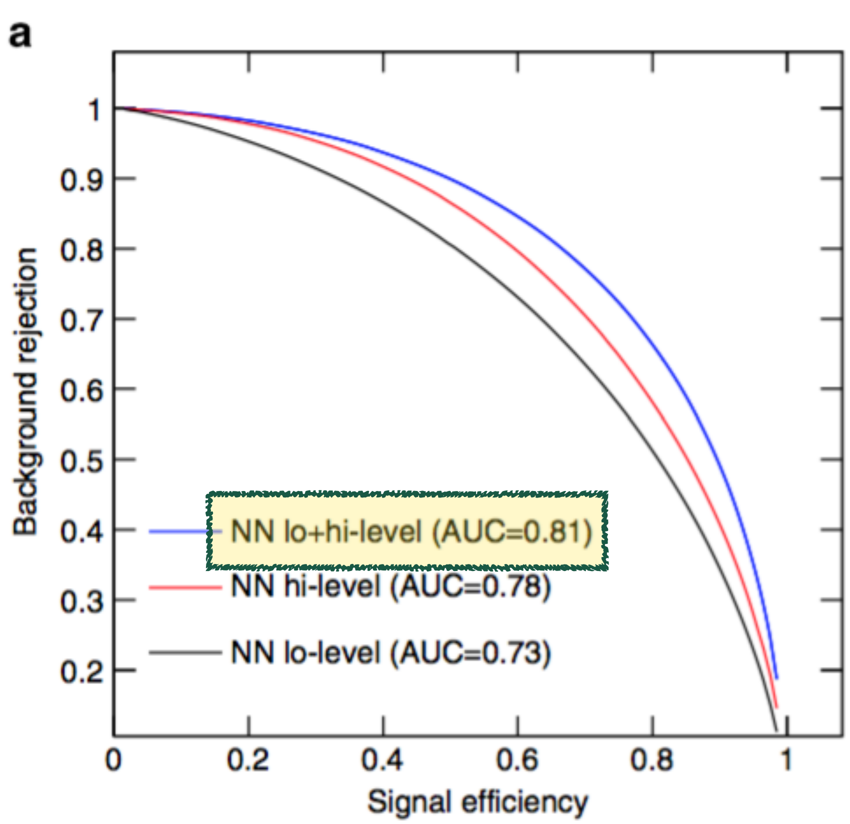
Deep learning in HEP

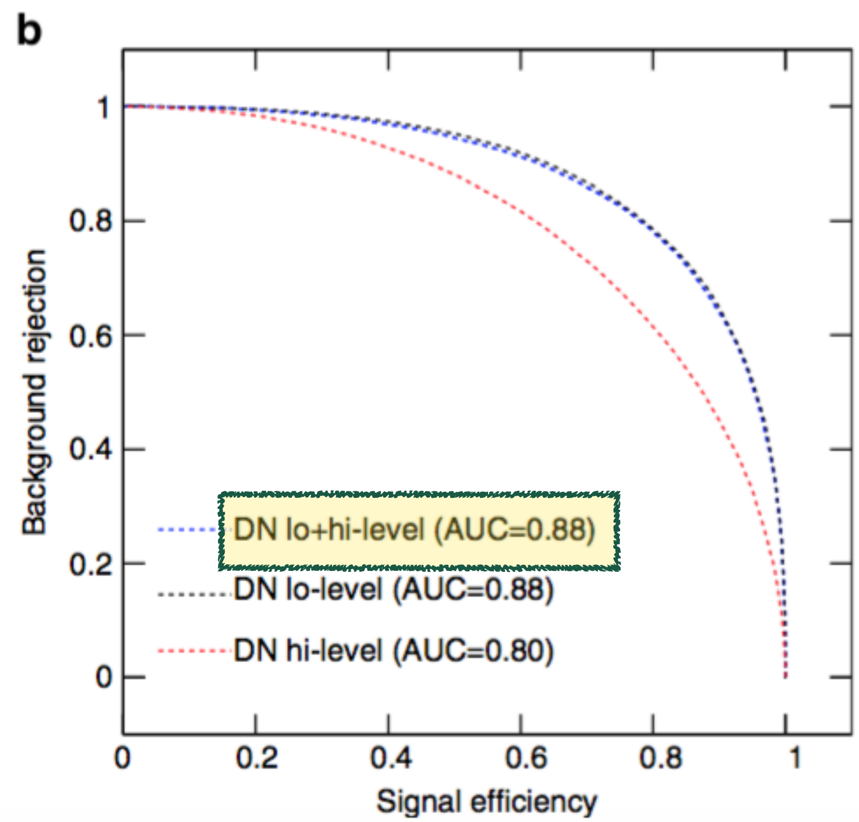
11 million training examples
1 hidden layer shallow network
5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and $1{,}000 \pm 50$ background events.





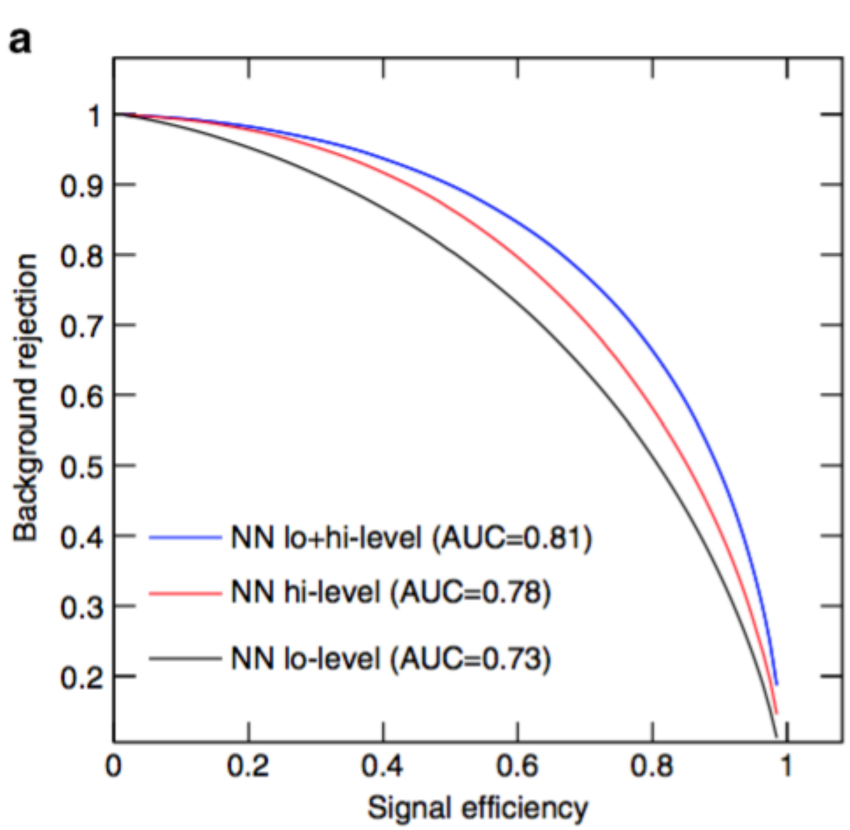
Deep learning in HEP

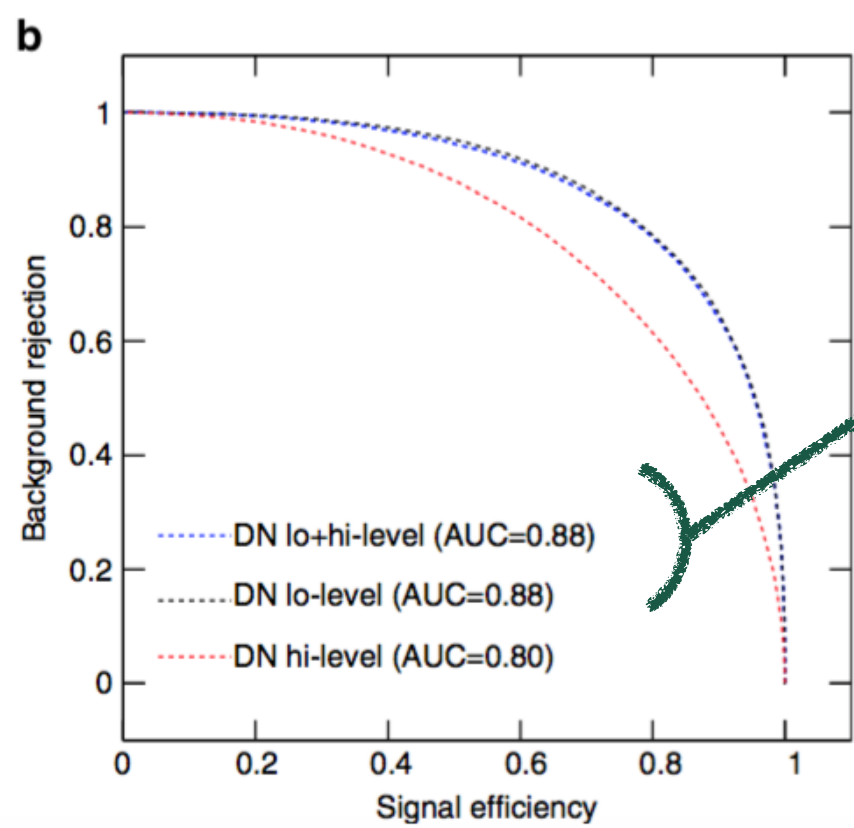
11 million training examples
1 hidden layer shallow network
5 layer deep network

Table 1 | Performance for Higgs benchmark.

Technique	Low-level	High-level	Complete
AUC			
BDT	0.73 (0.01)	0.78 (0.01)	0.81 (0.01)
NN	0.733 (0.007)	0.777 (0.001)	0.816 (0.004)
DN	0.880 (0.001)	0.800 (<0.001)	0.885 (0.002)
Discovery significance			
NN	2.5σ	3.1σ	3.7σ
DN	4.9σ	3.6σ	5.0σ

Comparison of the performance of several learning techniques: boosted decision trees (BDT), shallow neural networks (NN), and deep neural networks (DN) for three sets of input features: low-level features, high-level features and the complete set of features. Each neural network was trained five times with different random initializations. The table displays the mean area under the curve (AUC) of the signal-rejection curve in Fig. 7, with s.d. in parentheses as well as the expected significance of a discovery (in units of Gaussian σ) for 100 signal events and $1,000 \pm 50$ background events.





High-level not helping much

Deep learning in HEP

Deep learning, using raw information, can outperform physics inspired observables.

- Let the machine use *all the information* available

Why isn't every experimental analysis done with machine learning then?

What data should the neural networks be trained on?

What does "raw information" mean?

Looking forward

- In today's tutorial, you will learn to do linear and logistic regression, from scratch (using linear algebra packages).
- From logistic regression, you will expand to program a neural network from scratch.

- In tomorrow's lecture, we will look at recent machine learning results in HEP.
  - How to represent the data
  - Generalizing from Monte Carlo to real data
  - How to train on unlabelled data (real data)