6.7900 Machine Learning (Fall 2024)

Lecture 10: nonlinear predictors, neural networks

(supporting slides)

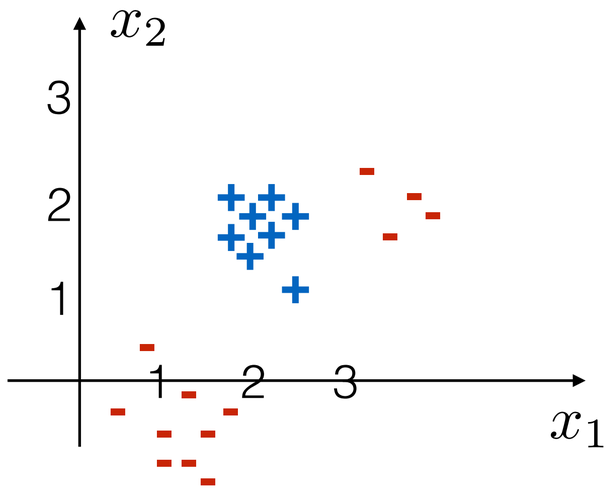
Outline for today's lecture

- A brief digression: building more expressive models with decision trees/forest

- Building more expressive models with constructed features, why not sufficient?
- Learning feature representations with neural networks
- Single hidden layer networks, signal transformation analysis
- Expressibility of neural networks
  - one hidden layer networks are universal
  - multi-layer networks express some (natural?) mappings much more compactly

Other non-linear methods

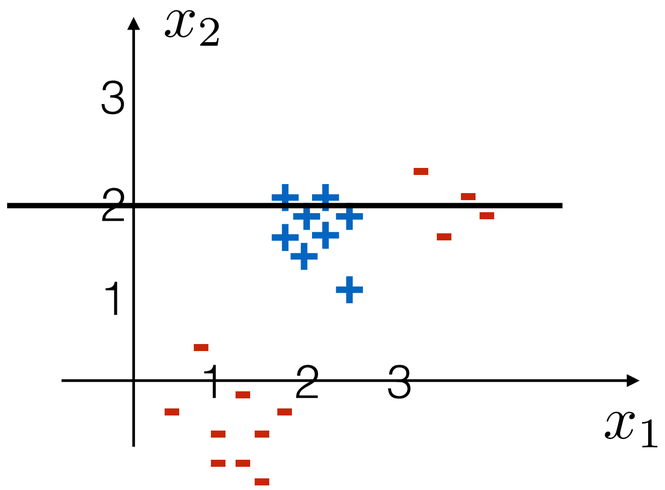
- Decision/regression trees (classes C = {-1,1})



$$N = 21,\ \hat{p}_1 = 8/21$$

Other non-linear methods

- Decision/regression trees (classes C = {-1,1})



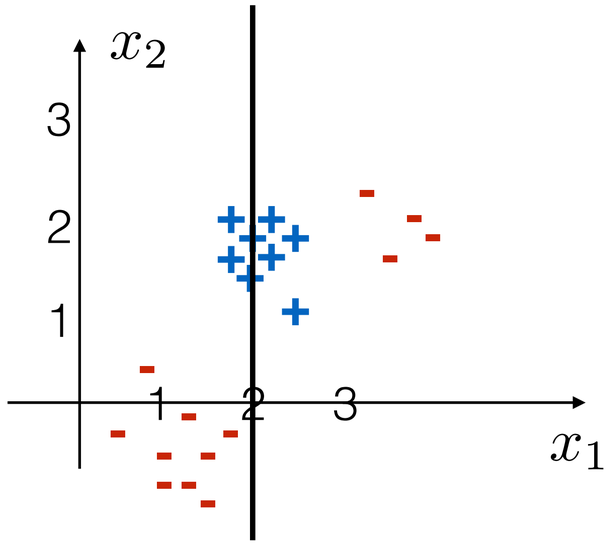
$$N = 21,\ \hat{p}_1 = 8/21$$

$x_2 < 2$ $\qquad$ $x_2 \geq 2$

- For each coordinate we try splitting in between successive values

Other non-linear methods

- Decision/regression trees (classes C = {-1,1})



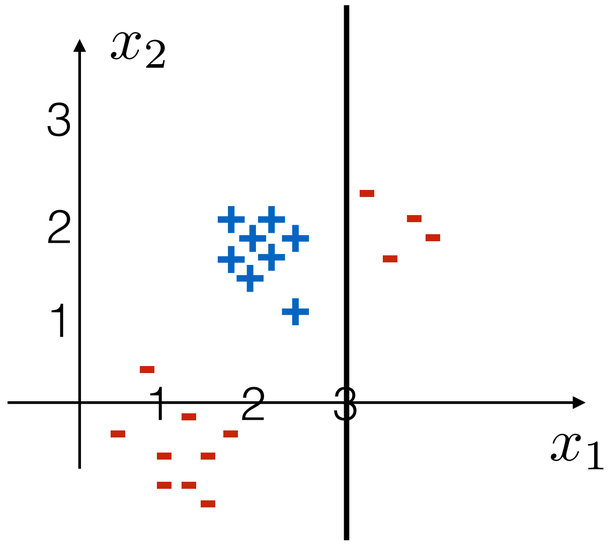
$$N = 21,\ \hat{p}_1 = 8/21$$

$x_1 < 2$ $\qquad$ $x_1 \geq 2$

- For each coordinate we try splitting in between successive values

Other non-linear methods

- Decision/regression trees (classes C = {-1,1})



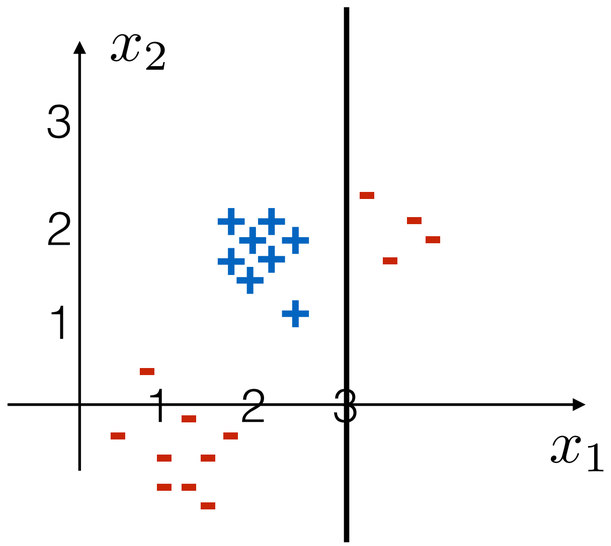
$$N = 21,\ \hat{p}_1 = 8/21$$

$x_1 < 3$ $\qquad$ $x_1 \geq 3$

- For each coordinate we try splitting in between successive values

Other non-linear methods

- Decision/regression trees (classes C = {-1,1})



$N = 21,\ \hat{p}_1 = 8/21$

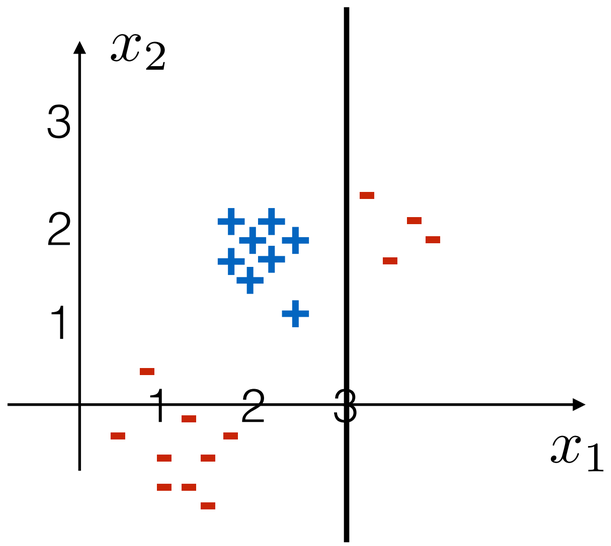
$x_1 < 3$ $\qquad$ $x_1 \geq 3$

$N^L = 17, \hat{p}_1^L = 8/17$ $\qquad$ $N^R = 4, \hat{p}_1^R = 0$

- For each coordinate we try splitting in between successive values

Other non-linear methods

- Decision/regression trees (classes C = {-1,1})



$N = 21,\ \hat{p}_1 = 8/21$

$x_1 < 3$ $\quad$ $x_1 \geq 3$

$N^L = 17, \hat{p}_1^L = 8/17$ $\quad$ $N^R = 4, \hat{p}_1^R = 0$

- We evaluate each candidate split by gini index, entropy, or squared error, and select the one that achieves the min error/impurity measure

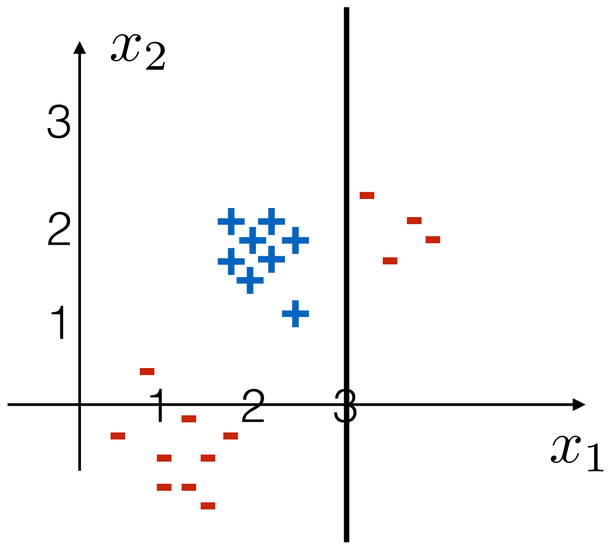
Find the coordinate & split value that minimizes the gini index

$$N_L \sum_{k \in C} \hat{p}_k^L (1 - \hat{p}_k^L) + N_R \sum_{k \in C} \hat{p}_k^R (1 - \hat{p}_k^R)$$

e.g, $\hat{p}_k^R$ = fraction of points falling in the right branch that have label k

Other non-linear methods

- Decision/regression trees (classes C = {-1,1})



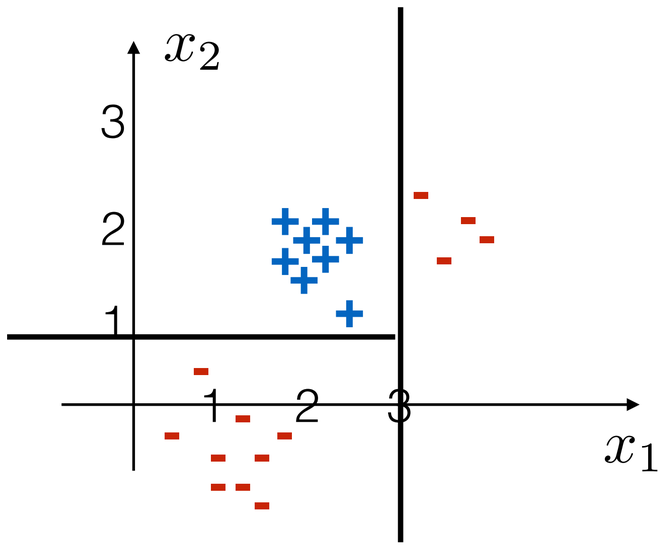
$N = 21,\ \hat{p}_1 = 8/21$

$x_1 < 3$ $\qquad$ $x_1 \geq 3$

$N^L = 17, \hat{p}_1^L = 8/17$ $\qquad$ $N^R = 4, \hat{p}_1^R = 0$

Other non-linear methods

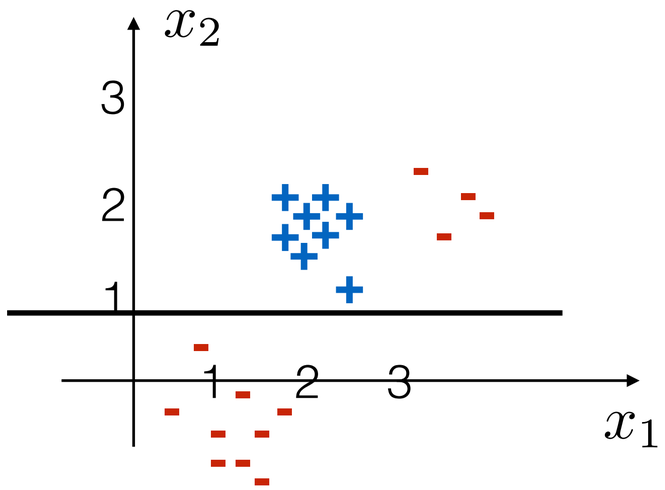
- Decision/regression trees (classes C = {-1,1})

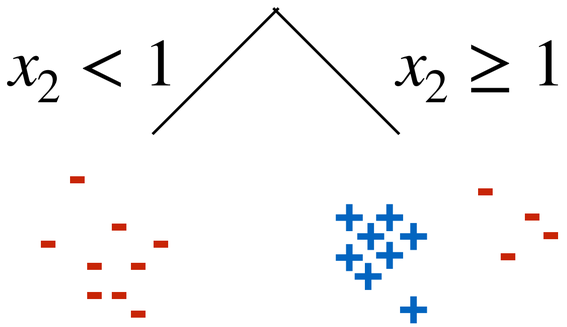


- We continue splitting recursively until a stopping criterion is reached (e.g., min number of points, level of purity, depth of tree, etc)

Other non-linear methods

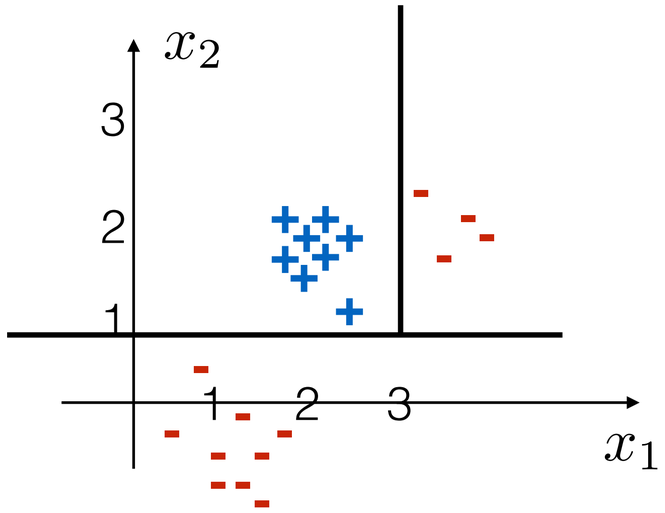
- Decision/regression trees (classes C = {-1,1})

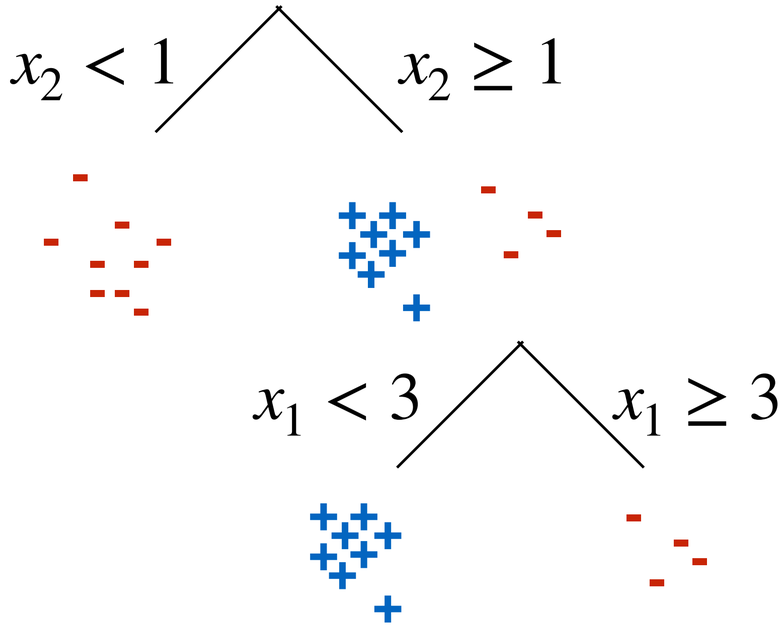




Other non-linear methods

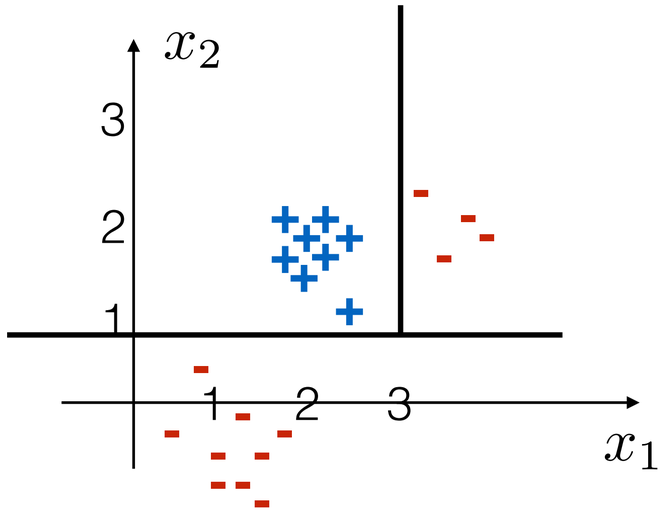
- Decision/regression trees (classes C = {-1,1})

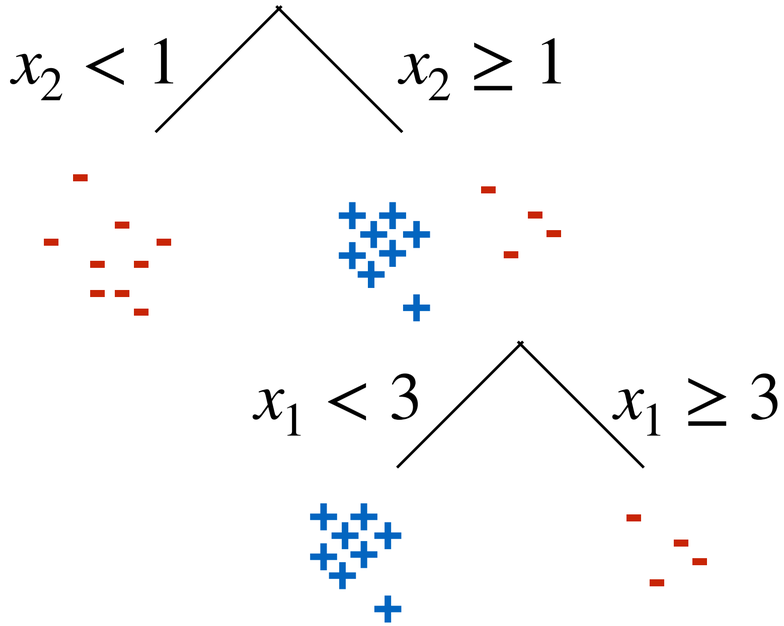




Other non-linear methods

- Decision/regression trees (classes C = {-1,1})





- Individual tree structures are not robust; shallow trees not powerful enough
- We can increase robustness and/or power by creating an ensemble of trees
- Effective methods for ensembles: random forest, XGBoost, etc

Random forest ensemble

- We build multiple trees by randomizing how they are constructed.
  - for each tree, we resample N examples with replacement (some examples no longer in the sample for that tree, some repeated)
  - for each split in the tree construction, we randomly restrict the model to consider only a subset of coordinates for that split
  - we repeat the process, building M trees in total

Random forest ensemble

- We build multiple trees by randomizing how they are constructed.
  - for each tree, we resample N examples with replacement (some examples no longer in the sample for that tree, some repeated)
  - for each split in the tree construction, we randomly restrict the model to consider only a subset of coordinates for that split
  - we repeat the process, building M trees in total

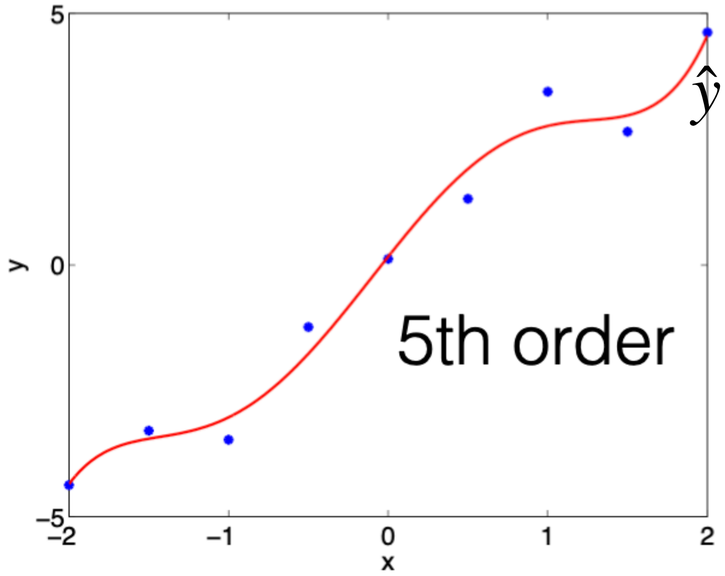
- If the errors that the trees make on test examples are independent, and if each tree is correct with probability $\mu \in (0.5,1]$ then their average prediction (majority voting) is correct with higher probability

$$\text{Bin}(n > M/2 \,|\, M, \mu) = \sum_{k=M/2+1}^{M} \binom{M}{k} \mu^k (1-\mu)^{M-k}$$

This probability approaches 1 for large M even if individual trees are just slightly better than random

Building complexity with feature expansion

- We can always "construct" non-linear features (e.g., polynomial) and add them to our linear regression or classification model
- The resulting model remains linear in the parameters (= easy to estimate), non-linear in terms of original examples (= powerful)



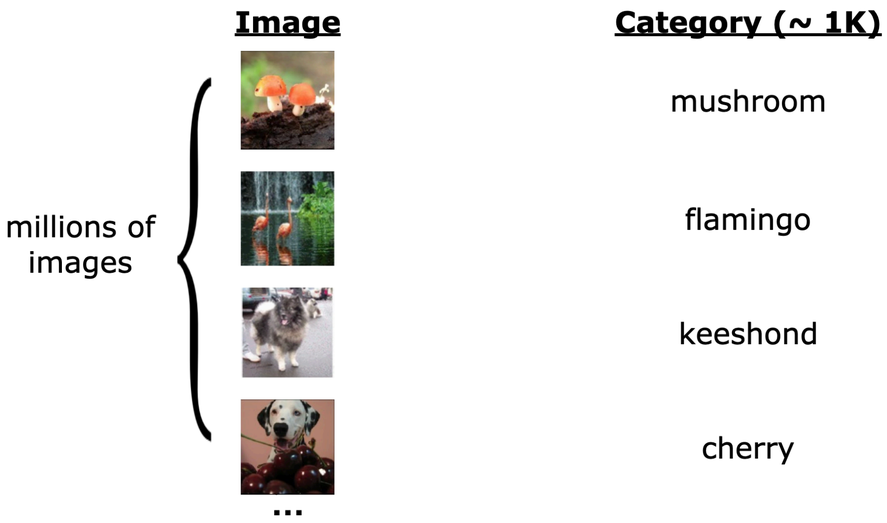
$$\hat{y} = \theta^T \phi(x)$$

$$\phi(x) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ x^4 \\ x^5 \end{bmatrix}$$

- But how do we decide which features (and how many) to use?

Handcrafted (or automated) features fall short

- E.g., image classification. Lots of earlier work on deriving feature descriptors of images (e.g., SIFT). But images are diverse, lots of categories…
- Learning what those features should be and how they should be combined became much more effective around 2012

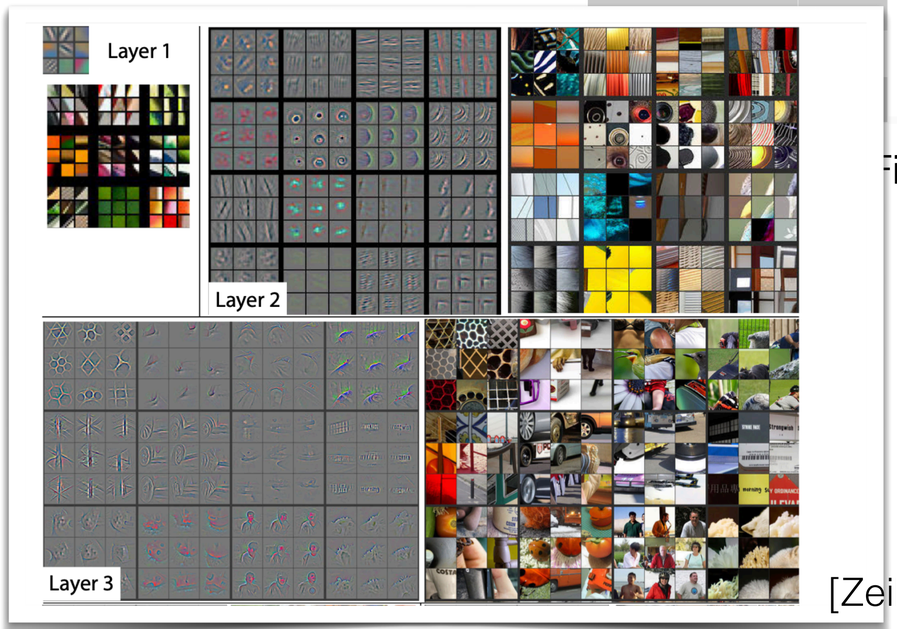


2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

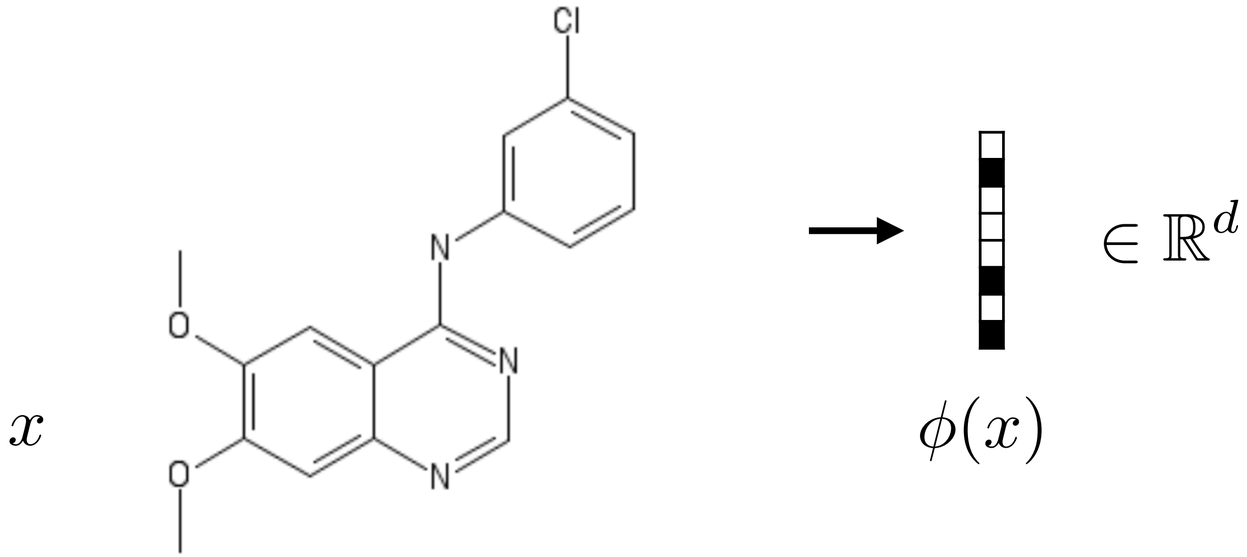
Figure by Yann LeCun]



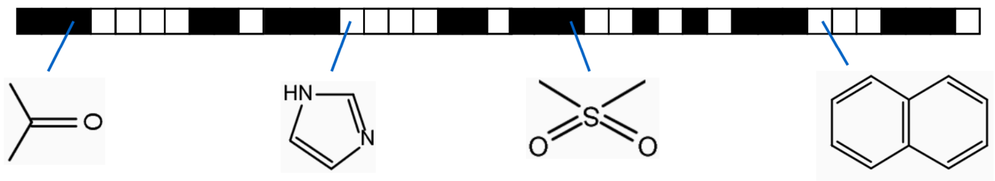
[Zeiler et al. 2013]

Handcrafted (or automated) features fall short

- E.g., molecular property prediction. We can map a complex object into a feature vector by checking which pieces it contains (“bag of words”)



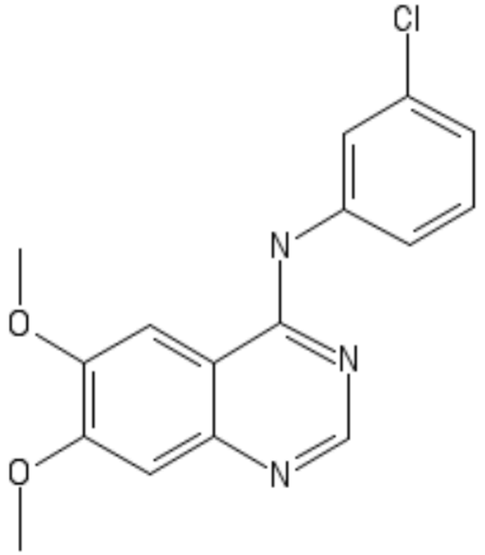
- In keyed fingerprints (binary vectors) each coordinate specifies whether a particular substructure is present in the molecule

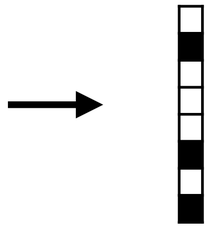


[Landrum 2012]

Handcrafted (or automated) features fall short

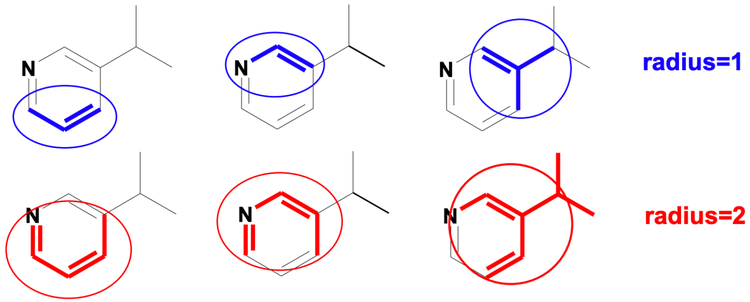
- E.g., molecular property prediction. We can map a complex object into a feature vector by checking which pieces it contains (“bag of words”)

x 



$\phi(x)$ $\in \mathbb{R}^d$

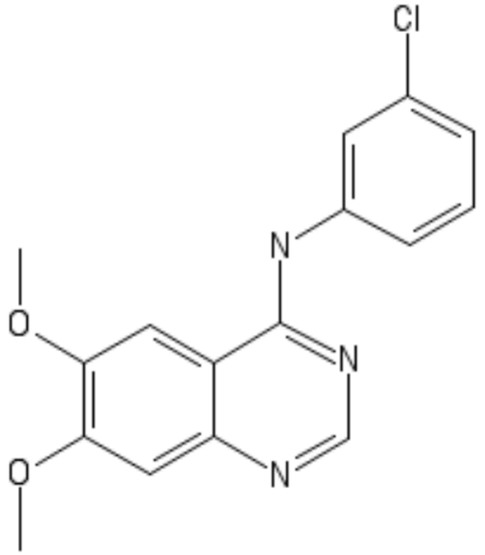
- Circular / extended connectivity fingerprints build binary valued feature vectors algorithmically

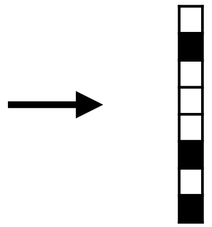


[Landrum 2012]

Handcrafted (or automated) features fall short

- E.g., molecular property prediction. We can map a complex object into a feature vector by checking which pieces it contains (“bag of words”)

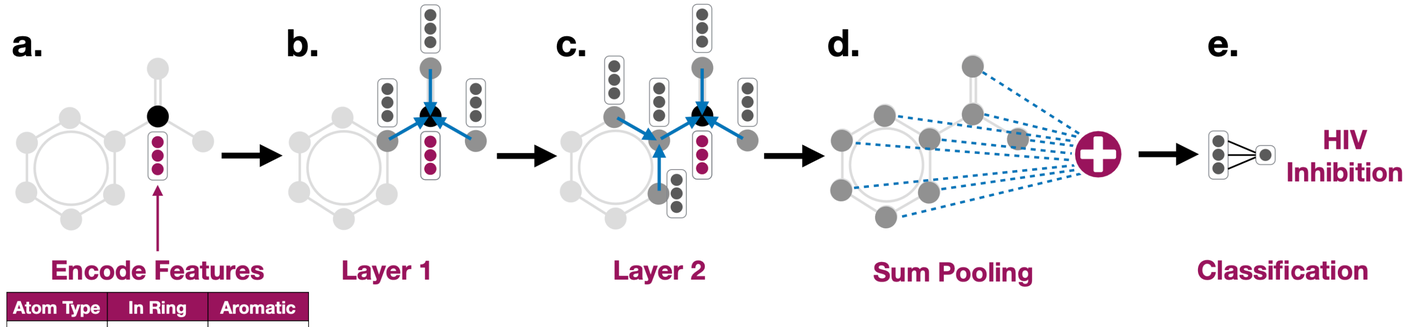
x 



$\in \mathbb{R}^d$

$\phi(x)$

- But learned representations (graph neural networks or GNNs) are often more effective



Handcrafted (or automated) features fall short

- E.g., text generation. We can decompose any distribution over sentences into an auto-regressive style generation (chain rule of probabilities)

$$P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \cdots P(x_n \mid x_{n-1}, \ldots, x_1)$$

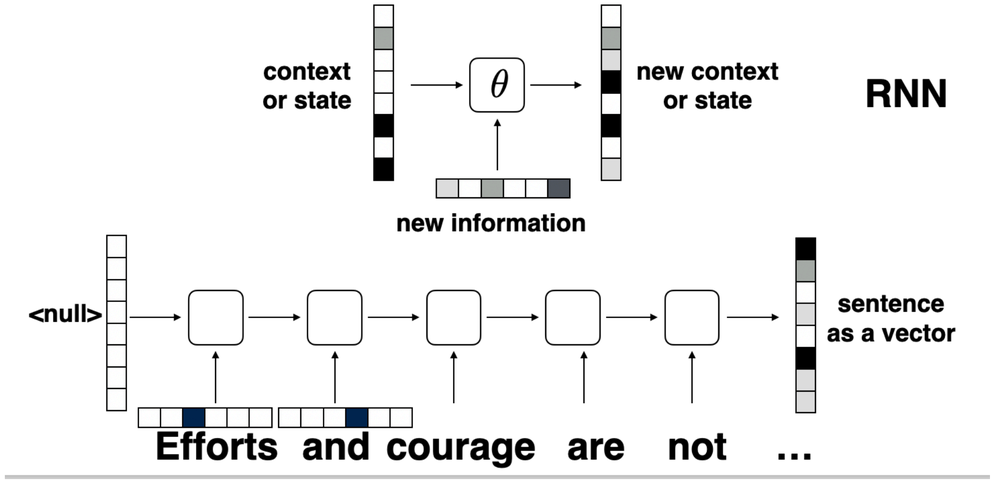
- In order to specify the associated probability tables, we would need to have a distribution over words (e.g., ~ 60K) for each combination of preceding words

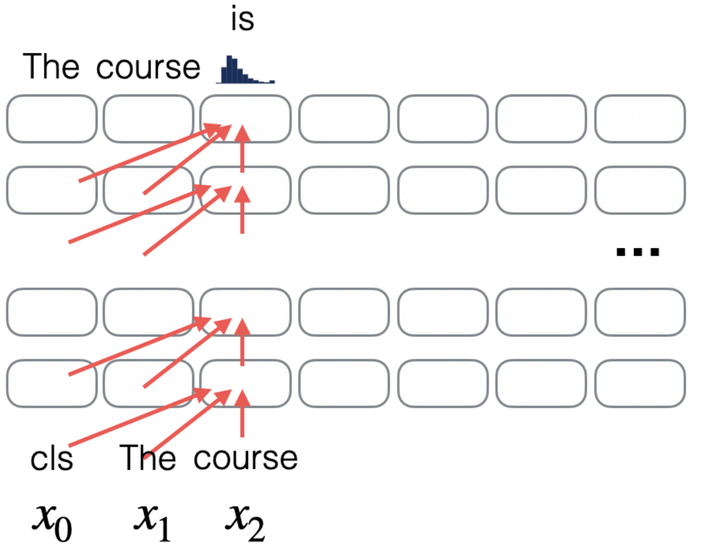
Handcrafted (or automated) features fall short

- E.g., text generation. We can decompose any distribution over sentences into an auto-regressive style generation (chain rule of probabilities)

$$P(x_1, x_2, \ldots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \cdots P(x_n \mid x_{n-1}, \ldots, x_1)$$

- In order to specify the associated probability tables, we would need to have a distribution over words (e.g., ~ 60K) for each combination of preceding words
- An effective summarization (learned feature description) of the preceding text is essential to solve this task well





Transformer

Composing complex models (MLP)

- A linear model $f(x;\theta) = w^T x + b \qquad \theta = \{w, b\}$

$1x1 \quad 1xd \quad dx1 \quad 1x1$

Composing complex models (MLP)

- A linear model $f(x;\theta) = w^T x + b \quad \theta = \{w, b\}$
  $1x1 \quad 1xd \;\; dx1 \;\; 1x1$
- A linear model with features $f(x;\theta) = w^T \phi(x) + b \quad \theta = \{w, b\}$
  $1x1 \quad 1xm \;\; mx1 \quad 1x1$

Composing complex models (MLP)

- A linear model $f(x;\theta) = \underset{1x1}{w^T}\underset{1xd}{x} + \underset{dx1}{b} \quad \underset{1x1}{} \quad \theta = \{w, b\}$
- A linear model with features $f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\phi(x)} + \underset{mx1}{b} \quad \underset{1x1}{} \quad \theta = \{w, b\}$
- A linear model with learnable linear features

$$f(x;\theta) = \underset{1x1}{w^T}(\underset{1xm}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \quad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

Composing complex models (MLP)

- A linear model $f(x;\theta) = \underset{1x1}{w^T}\underset{1xd}{x} + \underset{dx1}{b}\underset{1x1}{} \qquad \theta = \{w, b\}$
- A linear model with features $f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\phi(x)} + \underset{mx1}{b}\underset{1x1}{} \qquad \theta = \{w, b\}$
- A linear model with learnable linear features… still just a linear model!!

$$f(x;\theta) = \underset{1x1}{w^T}(\underset{1xm}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \qquad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

Composing complex models (MLP)

- A linear model $f(x;\theta) = \underset{1x1}{w^T}\underset{1xd}{x} + \underset{dx1}{b} \quad \underset{1x1}{} \quad \theta = \{w, b\}$
- A linear model with features $f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\phi(x)} + \underset{mx1}{b} \quad \underset{1x1}{} \quad \theta = \{w, b\}$
- A linear model with learnable linear features… still just a linear model!!

$$f(x;\theta) = \underset{1x1}{w^T}(\underset{1xm}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \quad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

- One hidden layer model (linear + non-linear + linear)

$$f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\tanh}(\underset{mx1}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \quad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

Composing complex models (MLP)

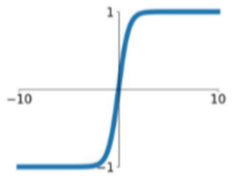
- A linear model $f(x;\theta) = \underset{1x1}{w^T}\underset{1xd}{x} + \underset{dx1}{b}\underset{1x1}{} \qquad \theta = \{w, b\}$
- A linear model with features $f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\phi(x)} + \underset{mx1}{b}\underset{1x1}{} \qquad \theta = \{w, b\}$
- A linear model with learnable linear features… still just a linear model!!

$$f(x;\theta) = \underset{1x1}{w^T}(\underset{1xm}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \qquad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

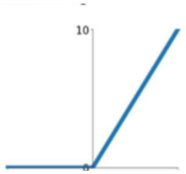
- One hidden layer model (linear + non-linear + linear)

$$f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\tanh}(\underset{mx1}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \qquad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

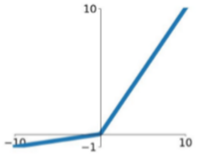
tanh
$\tanh(x)$



ReLU
$\max(0, x)$



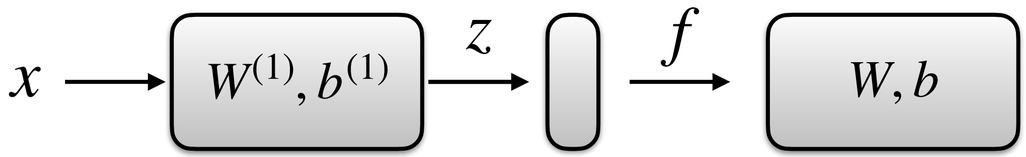
Leaky ReLU
$\max(0.1x, x)$



Etc.

Expressibility of neural models

- **Universal approximation theorem (informal):** A two layer neural network (one hidden layer network) has capacity to approximate any continuous function from a compact input set $K \subset \mathbb{R}^d$ to $\mathbb{R}^m$ to an arbitrary accuracy. The activation function has to be non-linear and non-polynomial



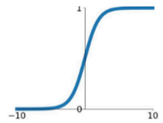
$$f(x;\theta) = Wf + b$$
$$= Wg(W^{(1)}x + b^{(1)}) + b$$

$$z = W^{(1)}x + b^{(1)} \quad f = g(z)$$

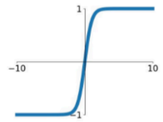
linear non-linear linear

The dim of the hidden layer, i.e., dim(z) may be large

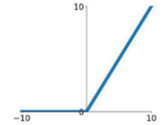
Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$



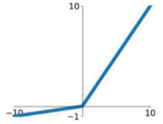
tanh
$\tanh(x)$



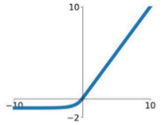
ReLU
$\max(0, x)$



Leaky ReLU
$\max(0.1x, x)$



ELU
$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$



Composing complex models (MLP)

- A linear model $f(x;\theta) = \underset{1x1}{w^T}\underset{1xd}{x} + \underset{dx1}{b} \quad \underset{1x1}{} \quad \theta = \{w, b\}$
- A linear model with features $f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\phi(x)} + \underset{mx1}{b} \quad \underset{1x1}{} \quad \theta = \{w, b\}$
- A linear model with learnable linear features… still just a linear model!!

$$f(x;\theta) = \underset{1x1}{w^T}(\underset{1xm}{W^{(1)}}\underset{mxd}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \qquad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

- One hidden layer model (linear + non-linear + linear)

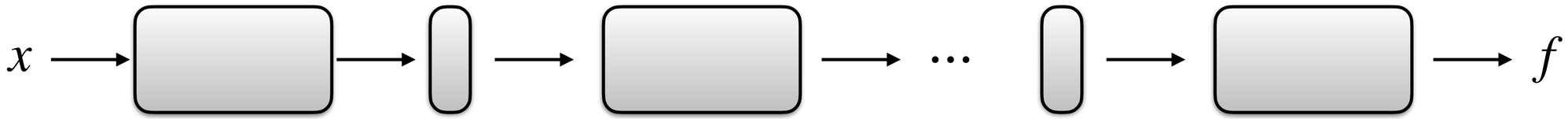
$$f(x;\theta) = \underset{1x1}{w^T}\underset{1xm}{\tanh}(\underset{mxd}{W^{(1)}}\underset{mx1}{x} + \underset{mx1}{b^{(1)}}) + \underset{1x1}{b} \qquad \theta = \{w, b, W^{(1)}, b^{(1)}\}$$

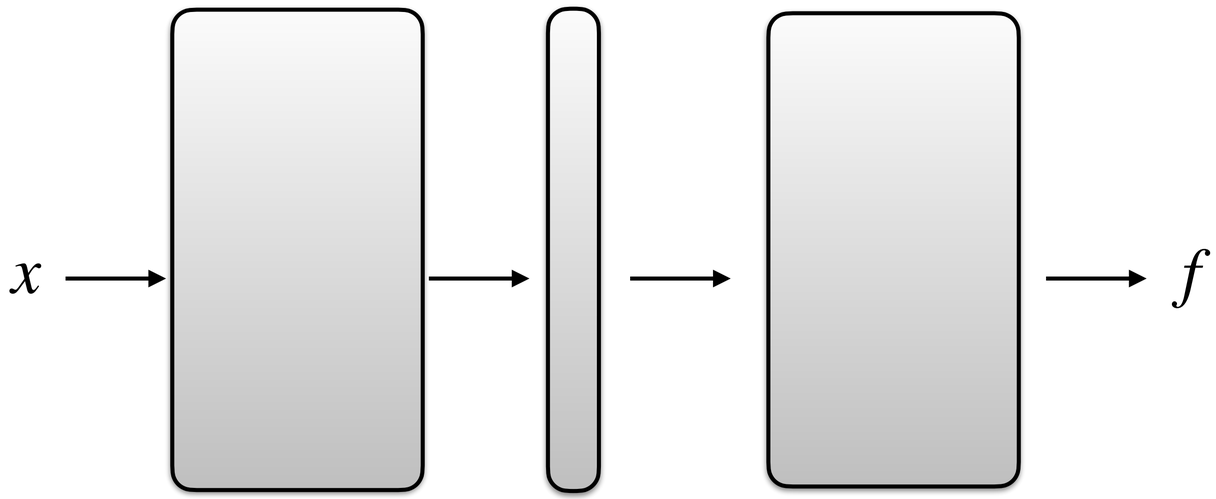
- A multi-layer neural perceptron (MLP, multiple linear +non-linear steps), e.g.,

$$f(x;\theta) = \underset{1x1}{w^T}\underset{1xk}{\tanh}\Big(\underset{kxm}{W^{(2)}}\underset{mx1}{\tanh}(\underset{mxd}{W^{(1)}}\underset{mx1}{x} + \underset{mx1}{b^{(1)}}) + \underset{kx1}{b^{(2)}}\Big) + b \qquad \theta = \{w, b, W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}\}$$

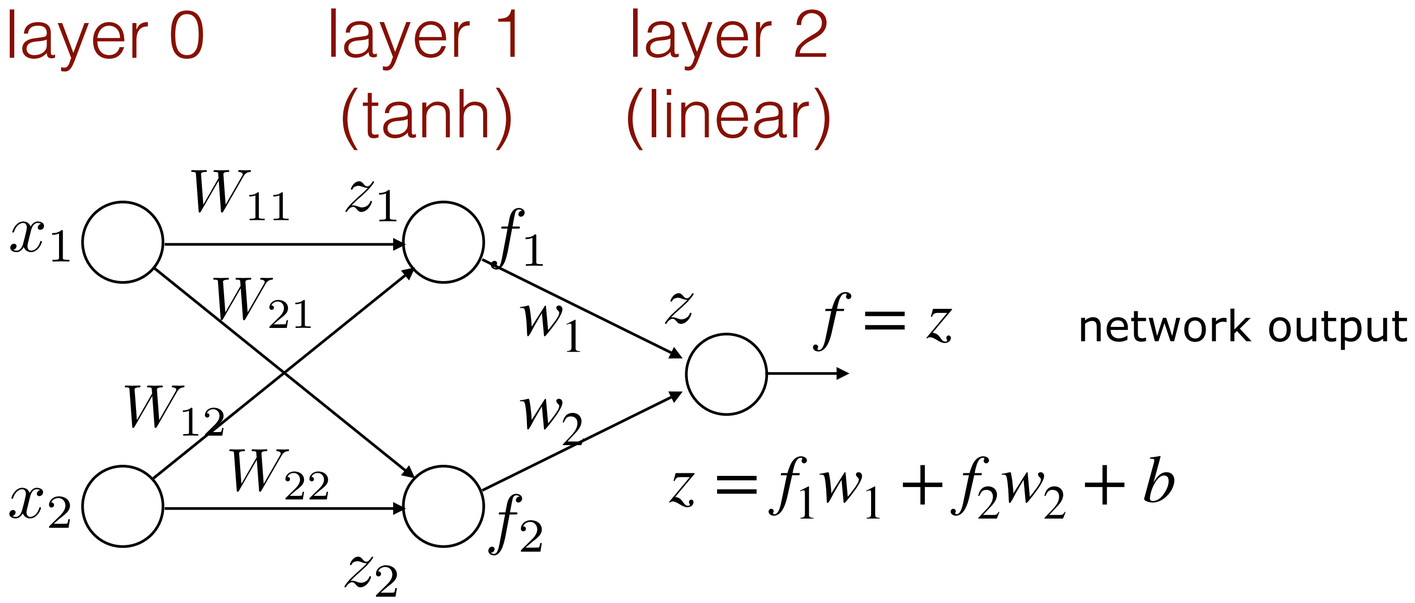
Expressibility: benefit of deep architectures

- Some mappings are expressible using multi-layer (deep) architectures much more easily than as two-layer networks (still universal)
- Effect of depth (**theorem**, informal): some functions that are easily computable with a finite width deep architectures may require exponentially many hidden units if expressed as two-layer models





One hidden layer model



pre-activations / aggregate inputs

$$z_1 = W_{11}x_1 + W_{12}x_2 + b_1$$

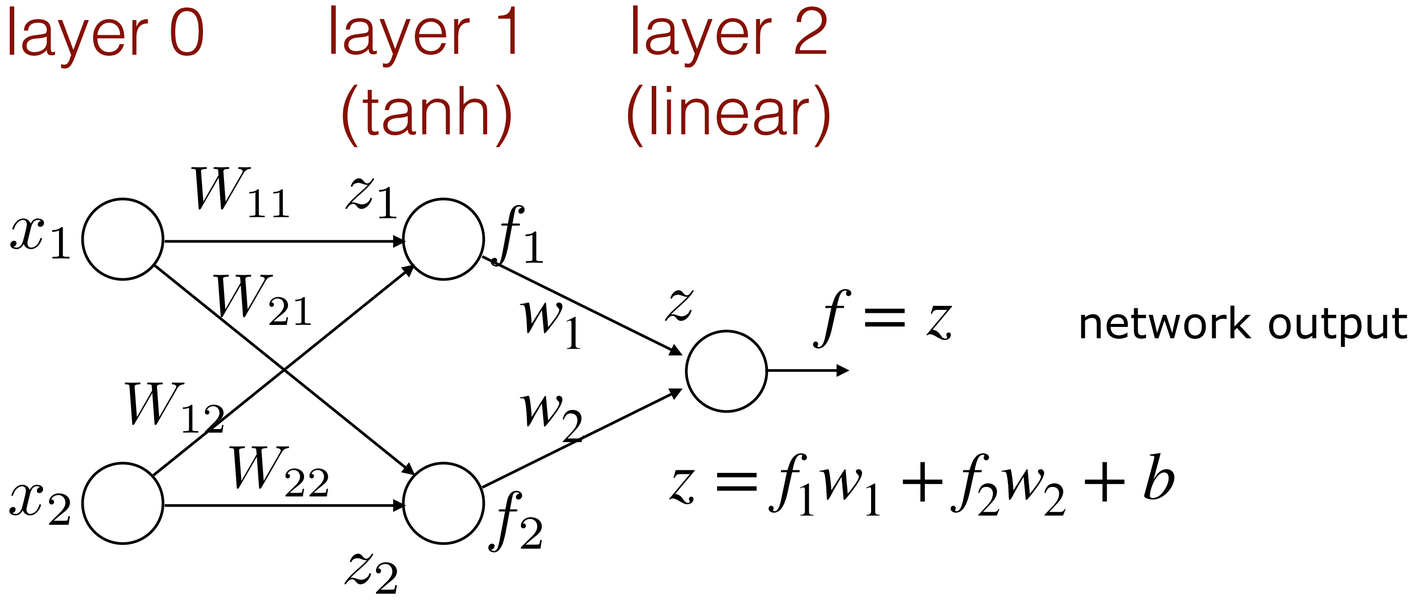
$$z_2 = W_{21}x_1 + W_{22}x_2 + b_2$$

unit activations

$$f_1 = \tanh(z_1)$$

$$f_2 = \tanh(z_2)$$

One hidden layer model



pre-activations / aggregate inputs

$$z_1 = W_{11}x_1 + W_{12}x_2 + b_1$$

$$z_2 = W_{21}x_1 + W_{22}x_2 + b_2$$

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = W \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

unit activations

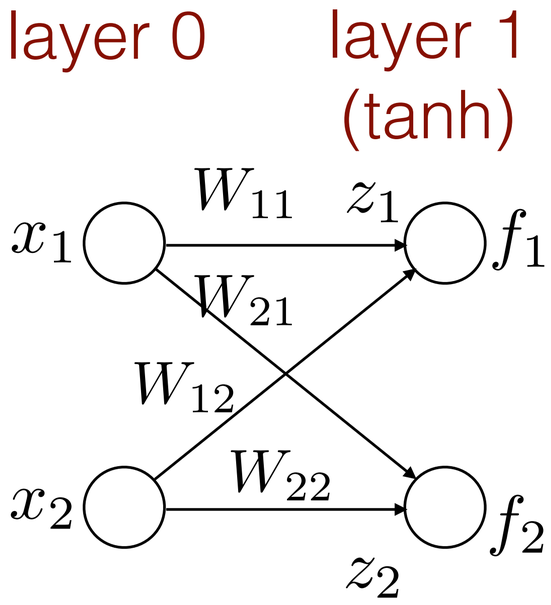
$$f_1 = \tanh(z_1)$$

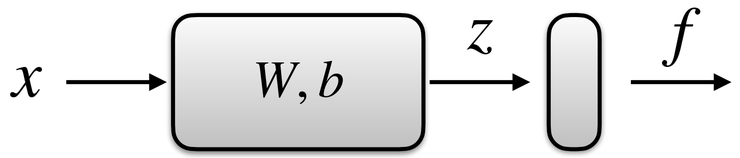
$$f_2 = \tanh(z_2)$$

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \tanh\left(\begin{matrix} z_1 \\ z_2 \end{matrix}\right)$$

One hidden layer model

- Neural signal transformation

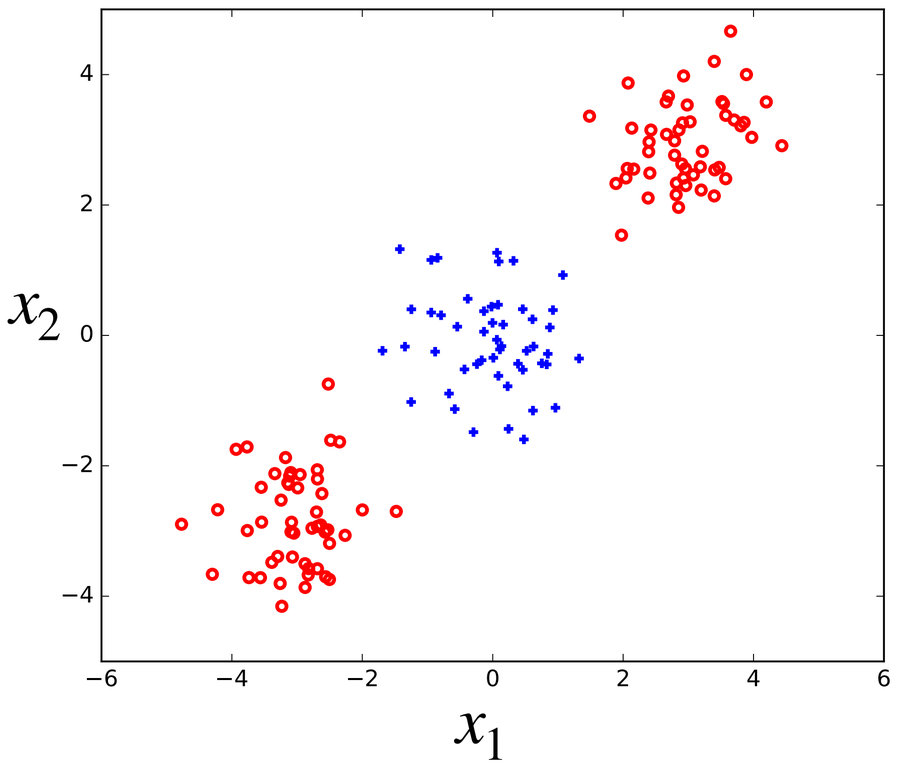




linear
tanh()
ReLU()
...

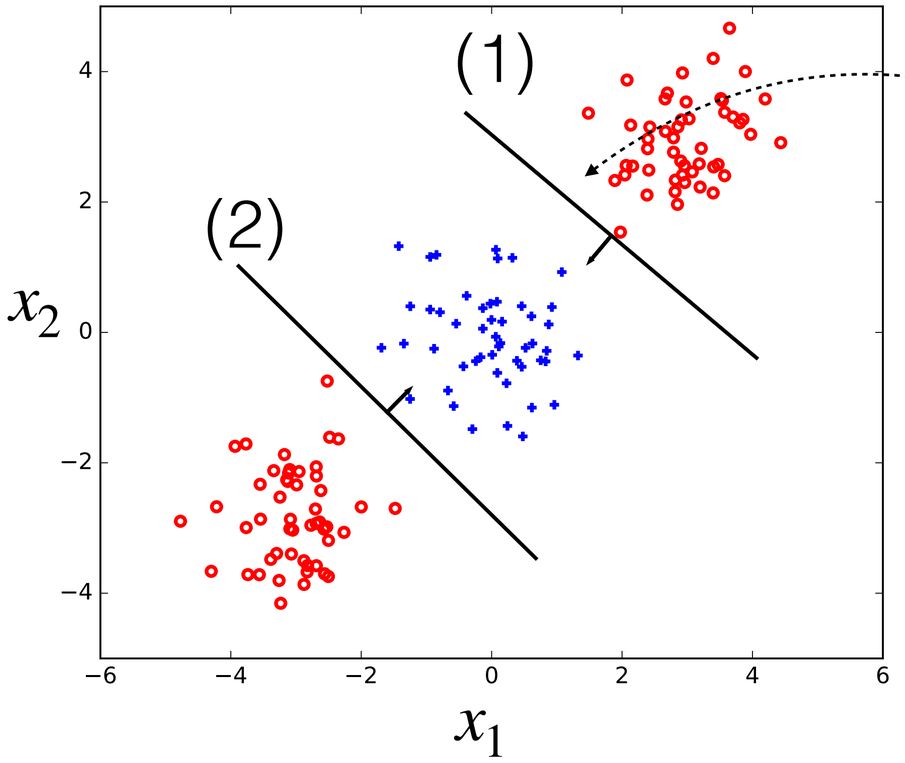
mapping $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ to $f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$

Example Problem

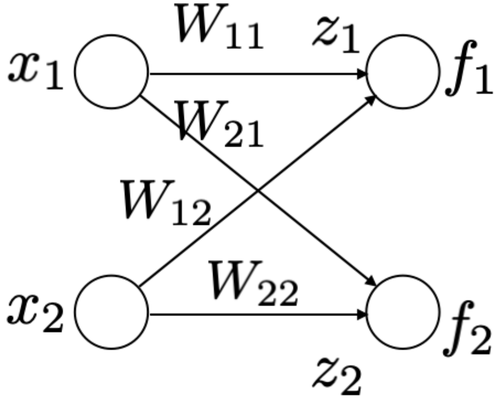


Hidden layer representation

Hidden layer units (defined by the dec boundaries)

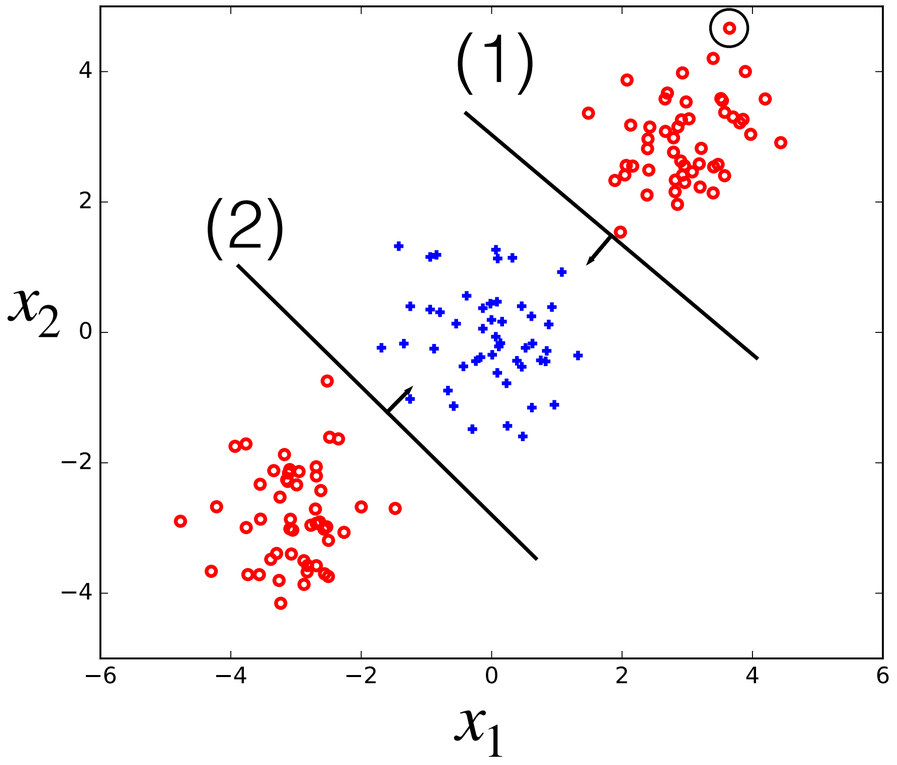


specifies $\begin{bmatrix} W_{11} \\ W_{12} \end{bmatrix}$, b_1

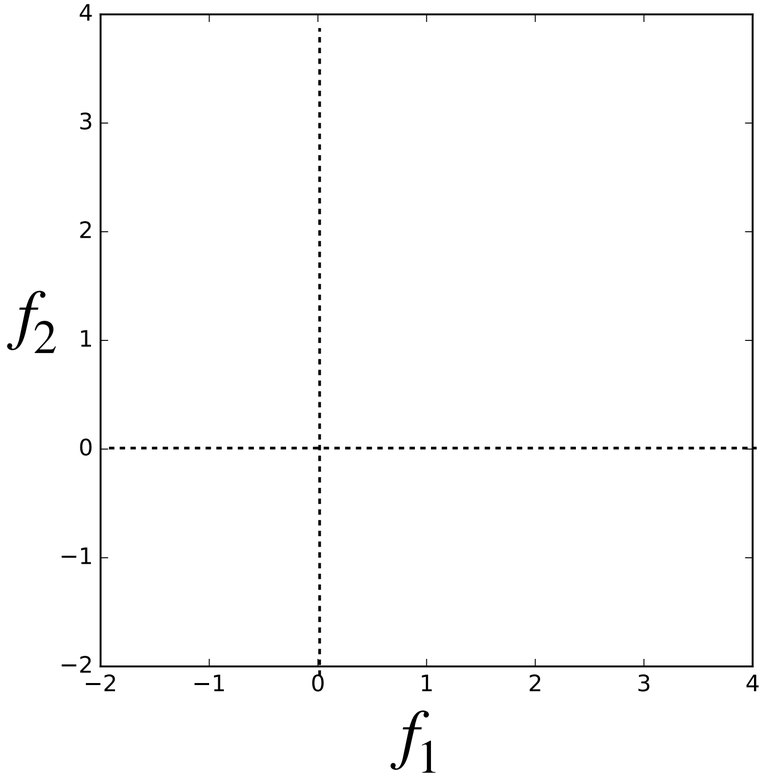


Hidden layer representation

Hidden layer units

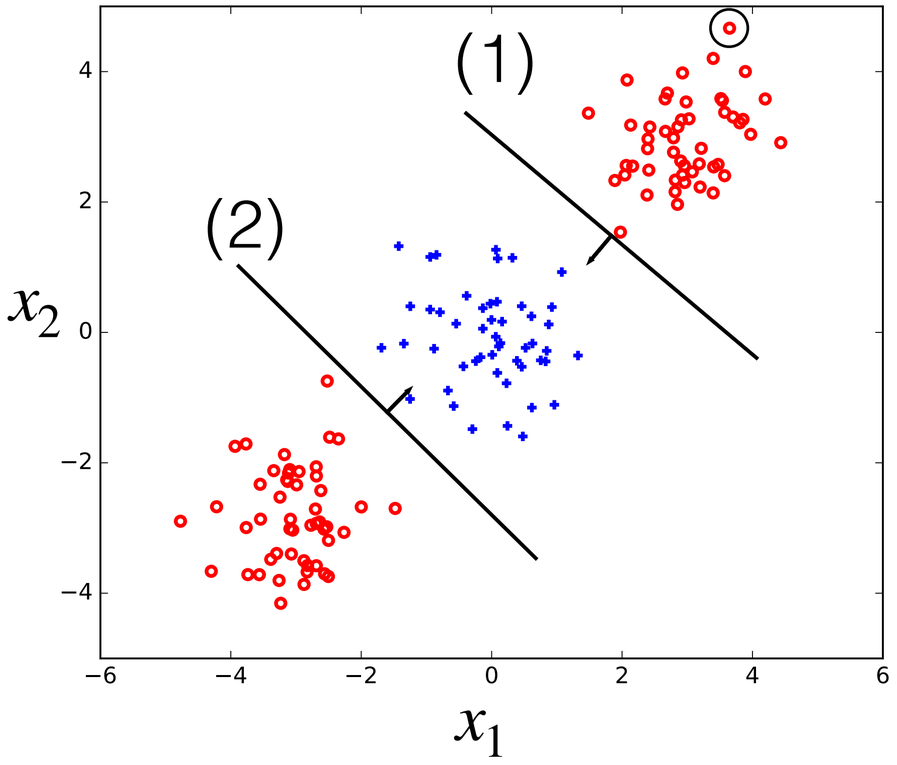


Linear activation

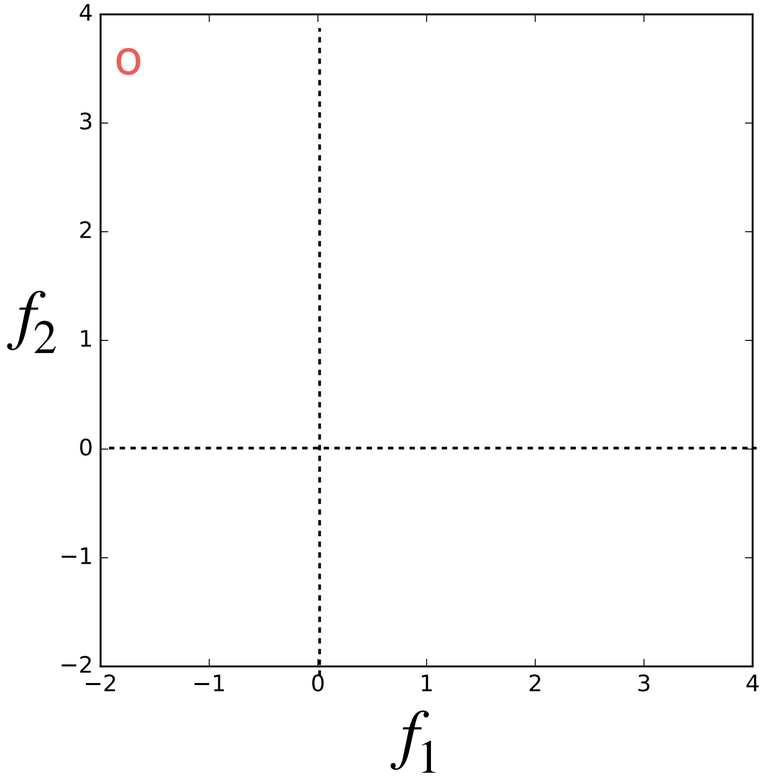


Hidden layer representation

Hidden layer units

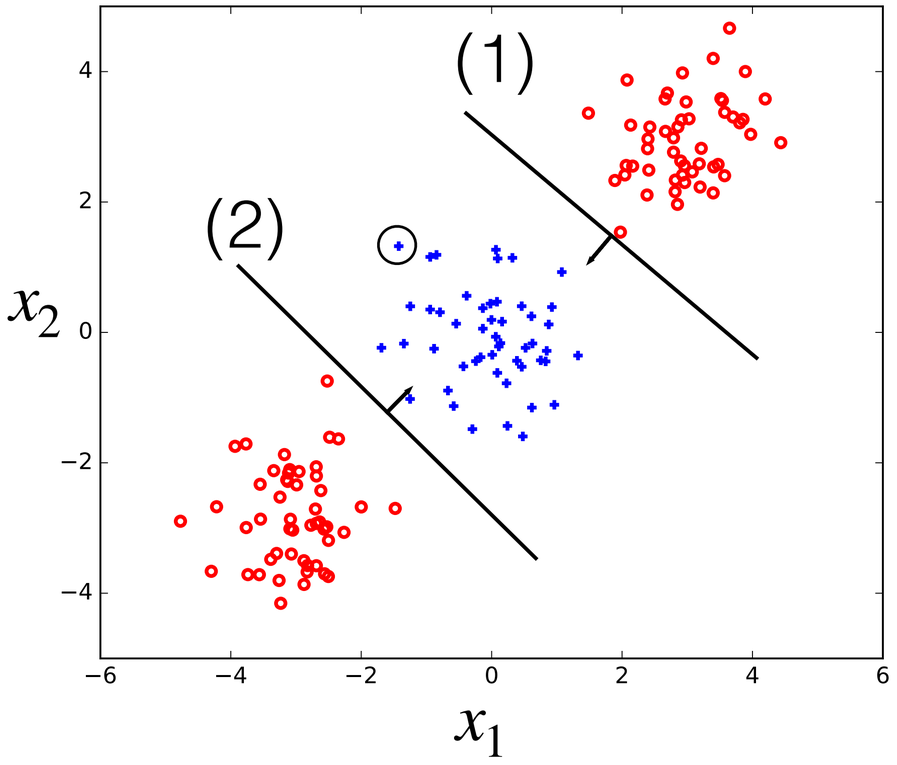


Linear activation

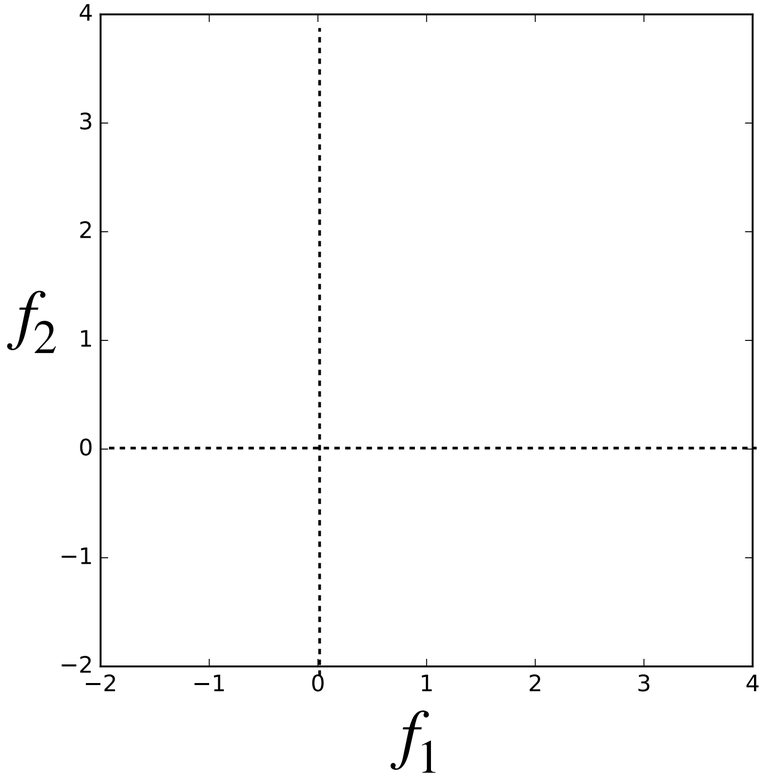


Hidden layer representation

Hidden layer units

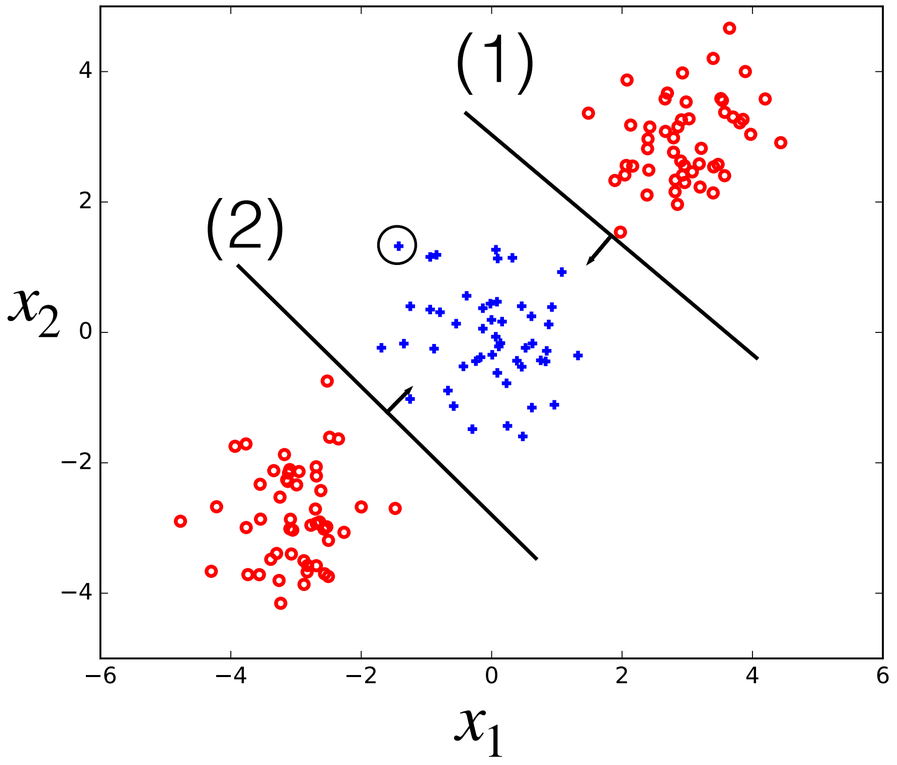


Linear activation

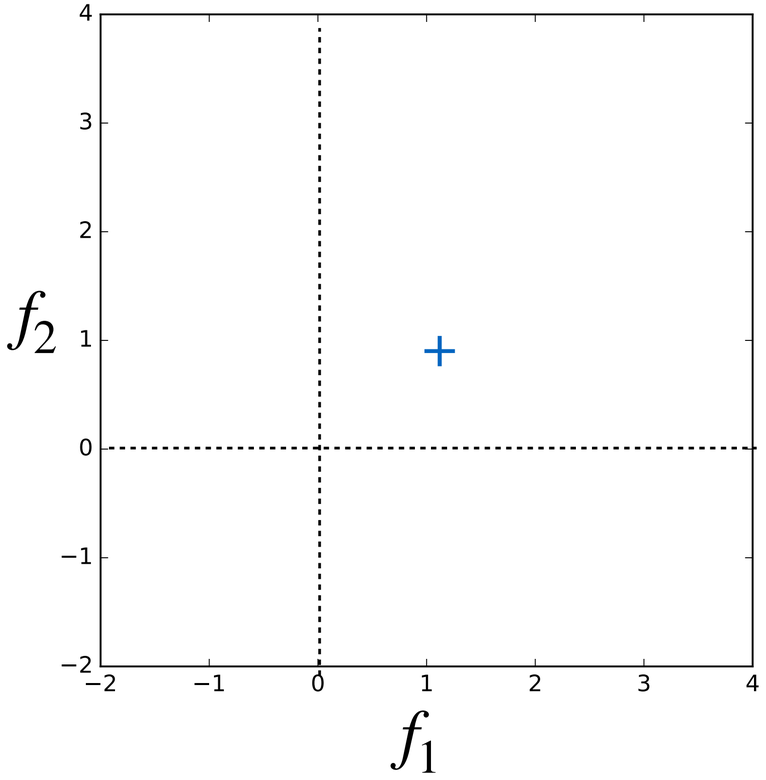


Hidden layer representation

Hidden layer units

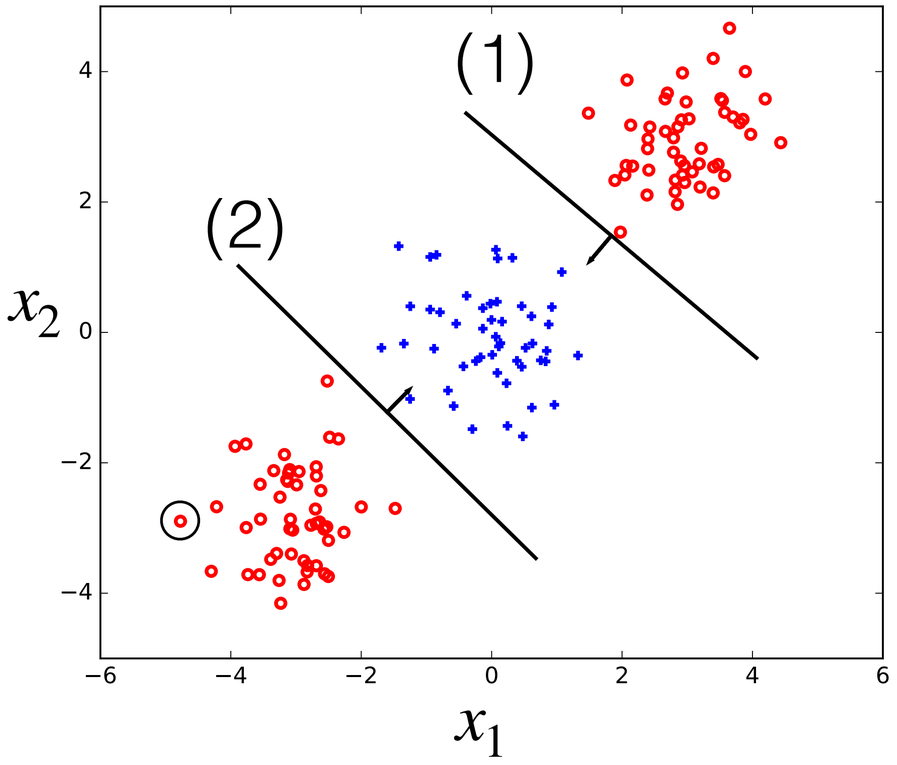


Linear activation

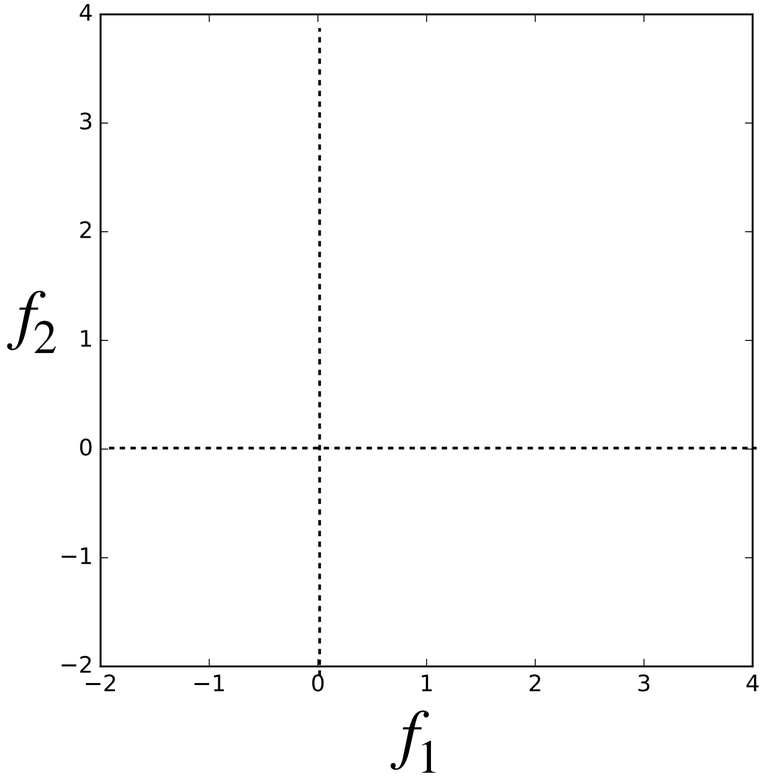


Hidden layer representation

Hidden layer units

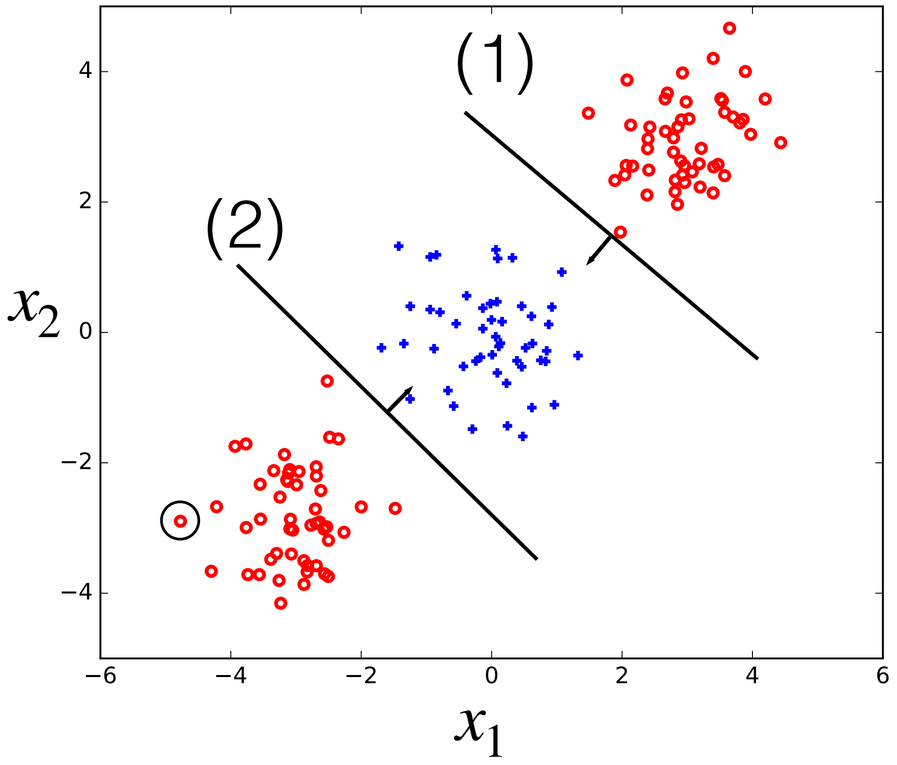


Linear activation

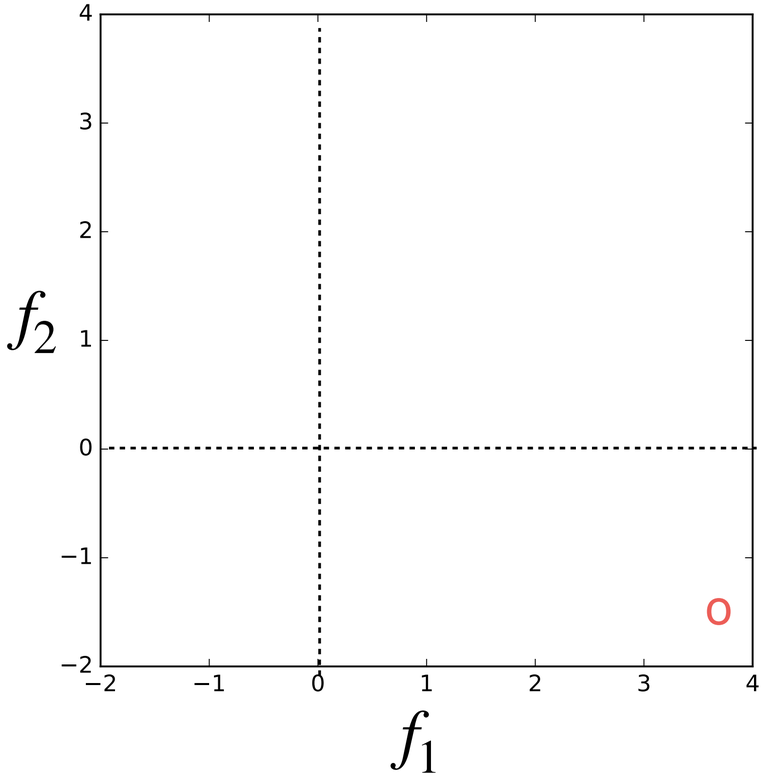


Hidden layer representation

Hidden layer units

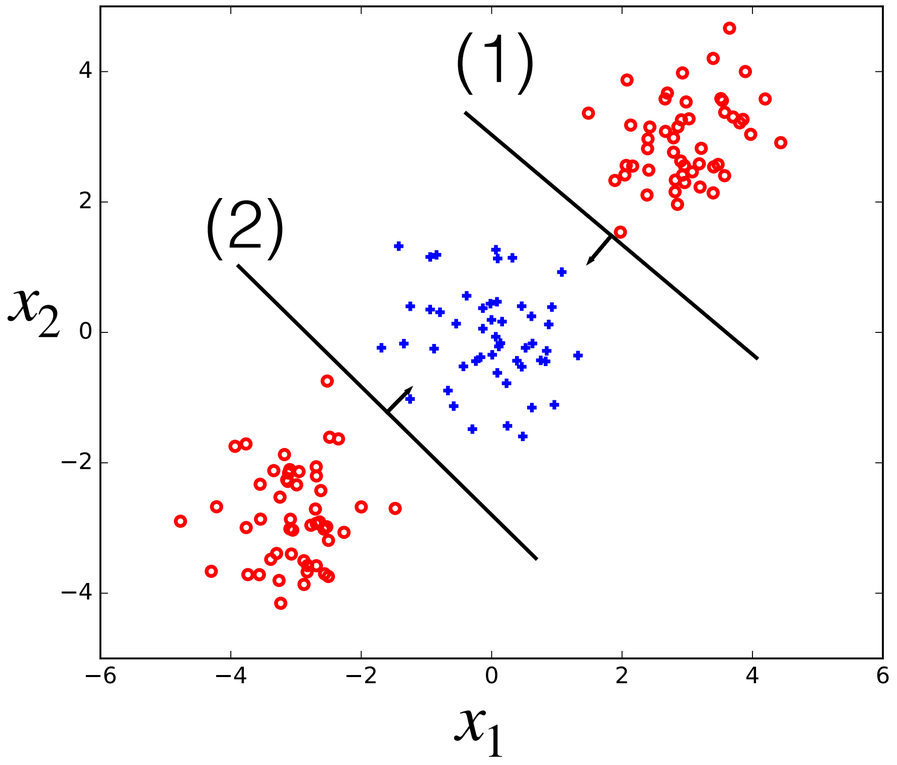


Linear activation

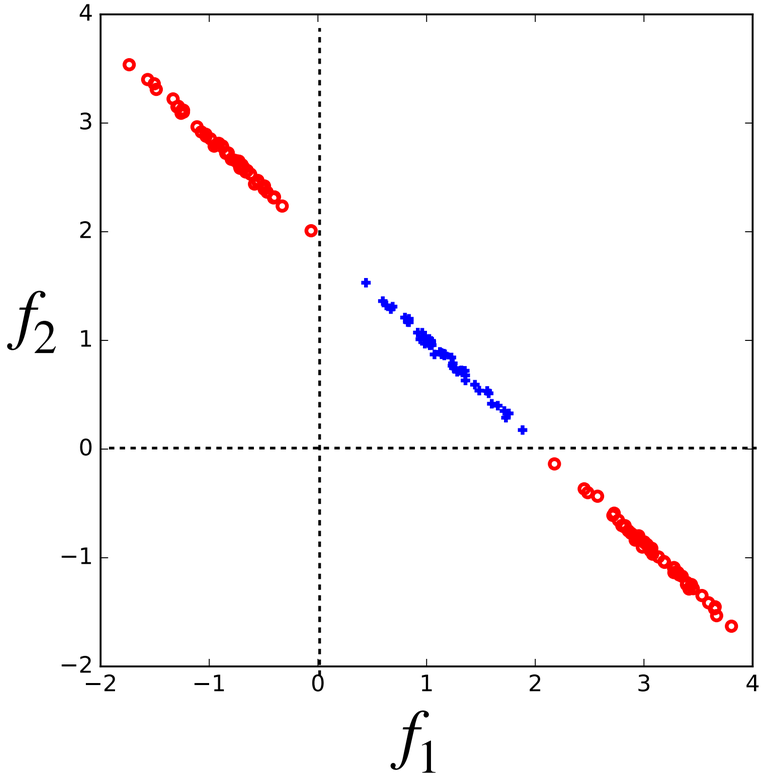


Hidden layer representation

Hidden layer units

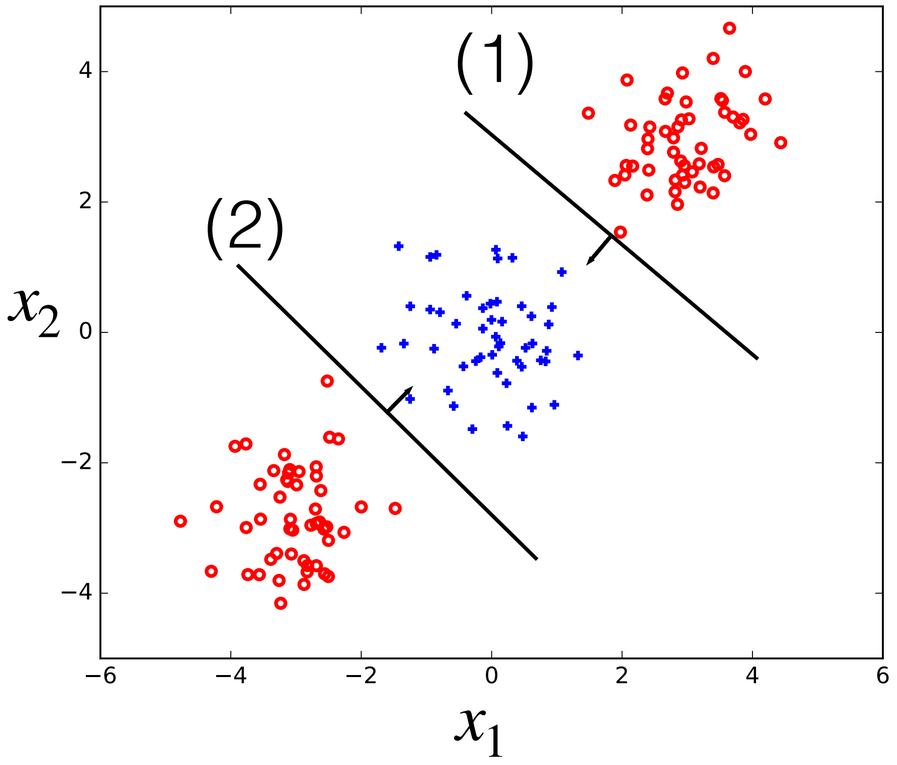


Linear activation

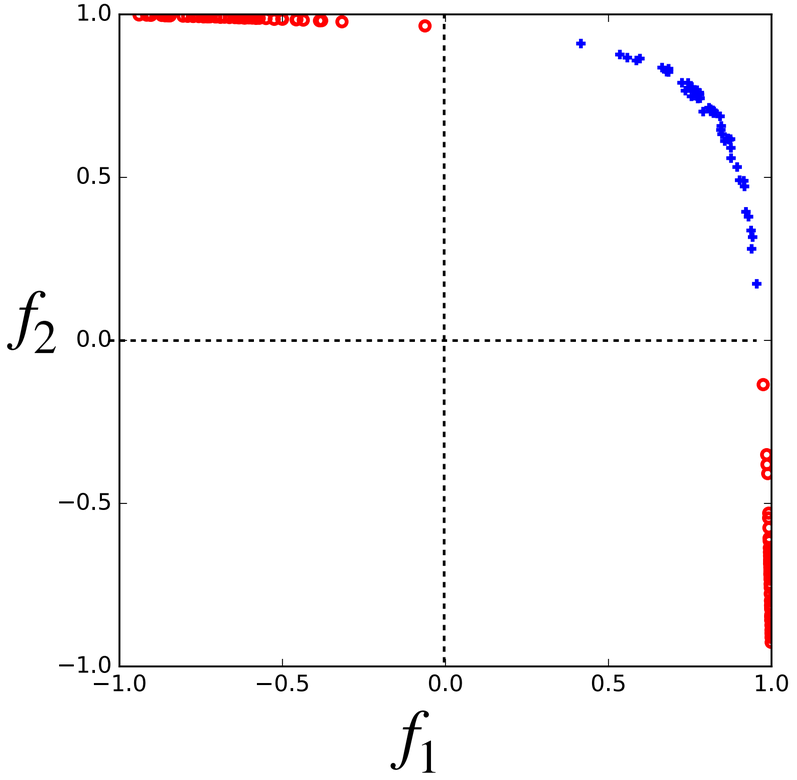


Hidden layer representation

Hidden layer units

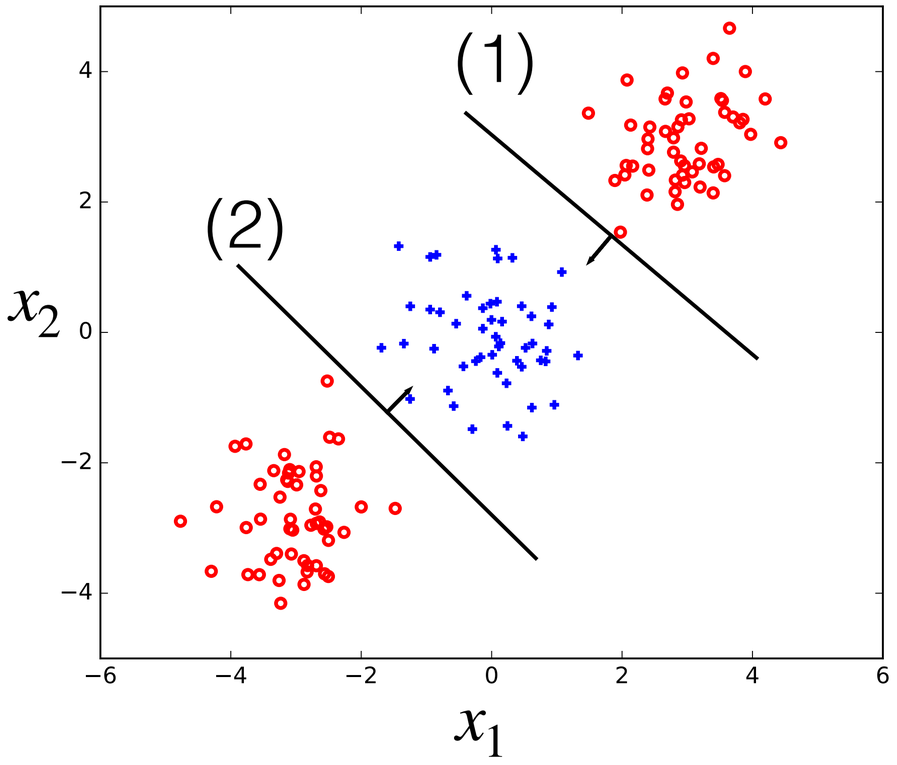


tanh activation

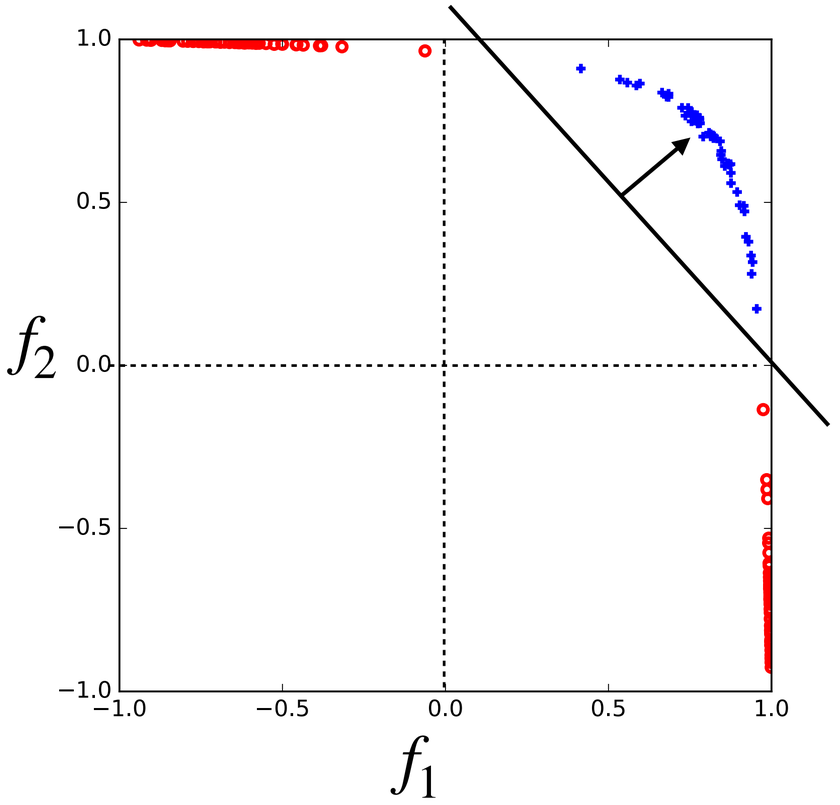


Hidden layer representation

Hidden layer units

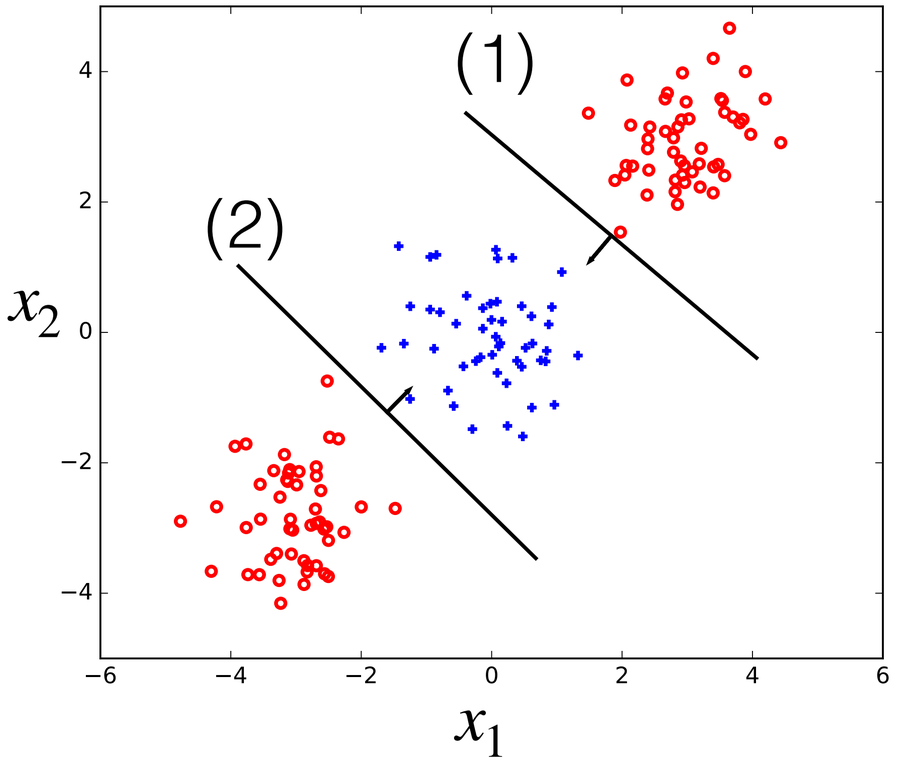


tanh activation

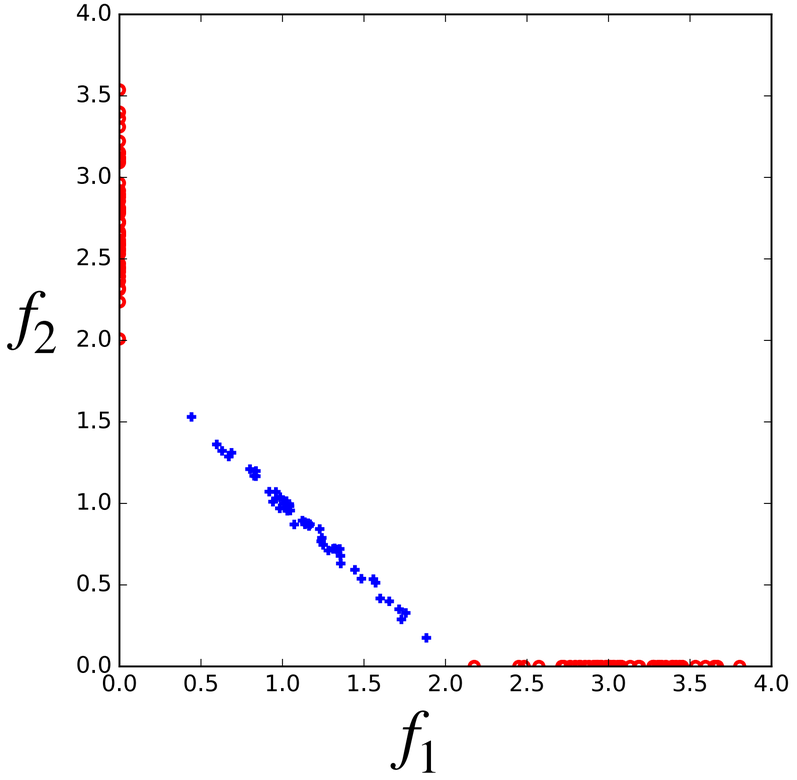


Hidden layer representation

Hidden layer units

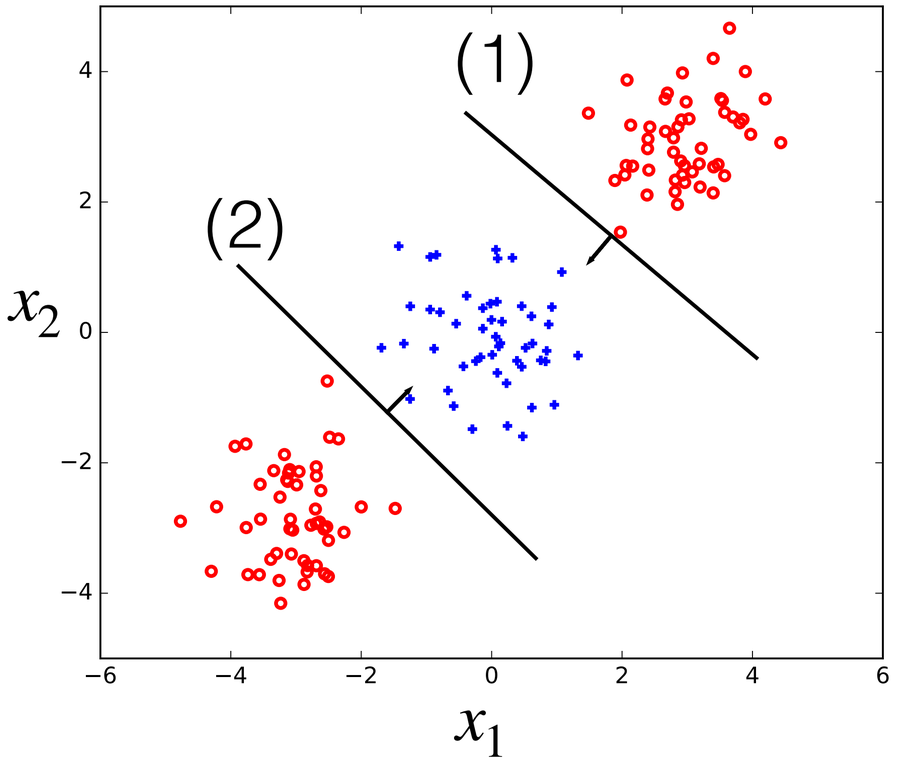


ReLU activation

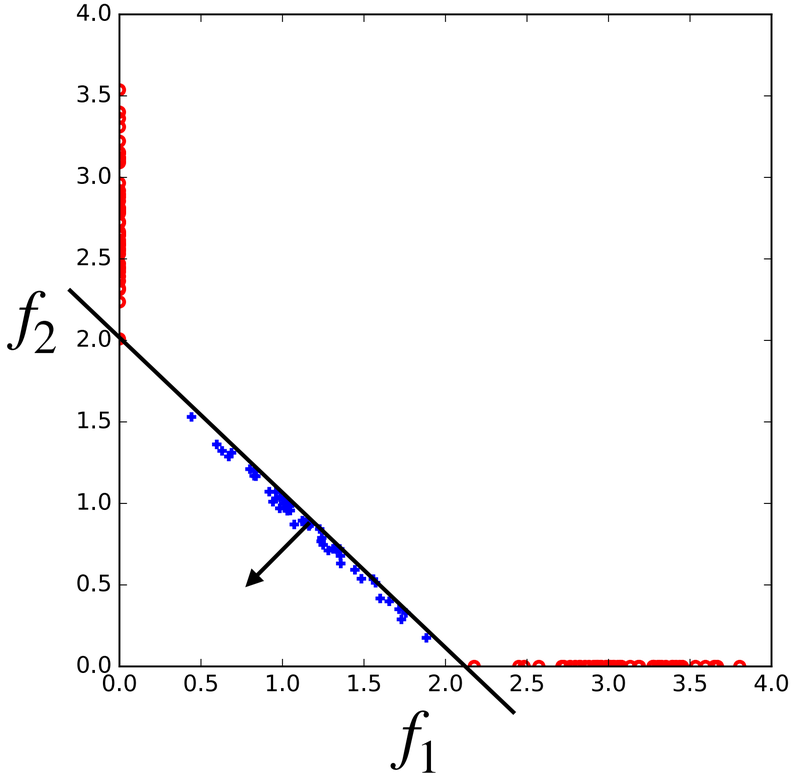


Hidden layer representation

Hidden layer units



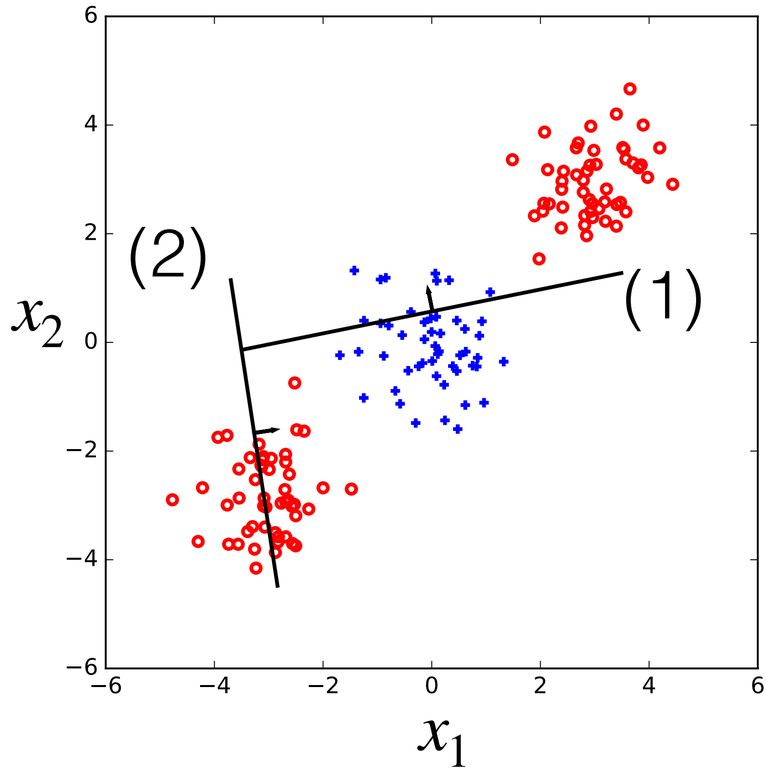
ReLU activation



Random hidden units

- Small number of random hidden units not likely useful

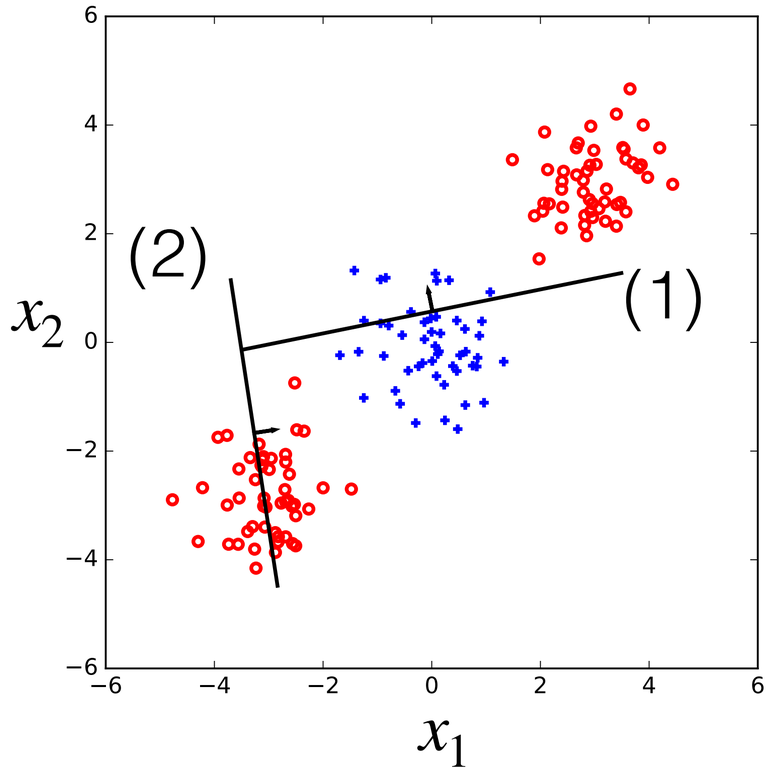
Hidden layer units



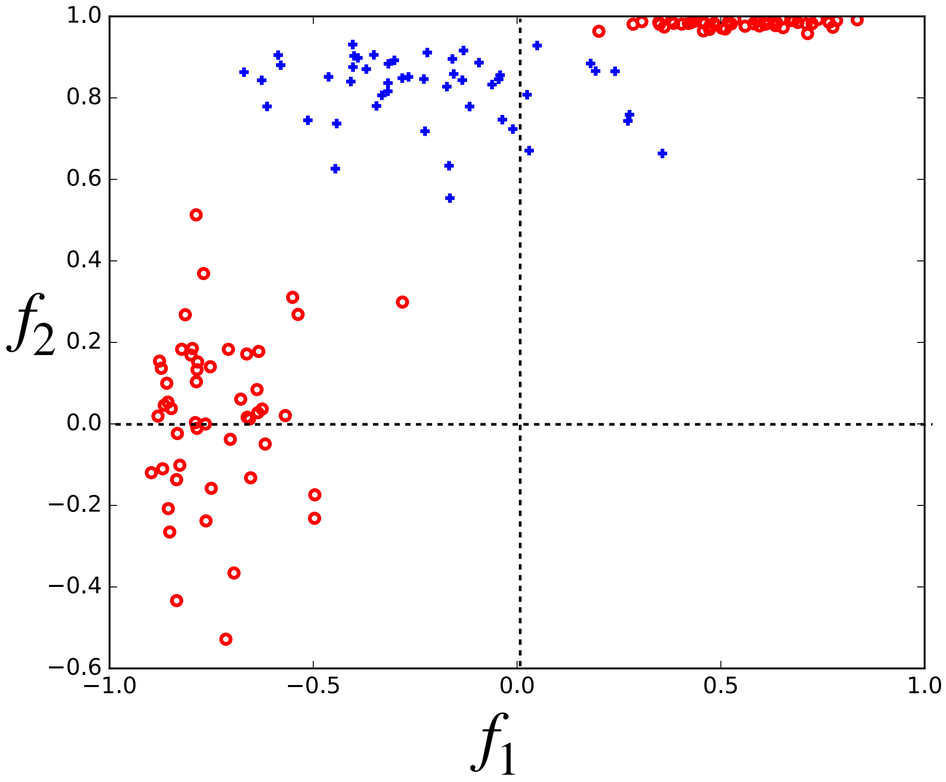
Random hidden units

- Small number of random hidden units not likely useful

Hidden layer units



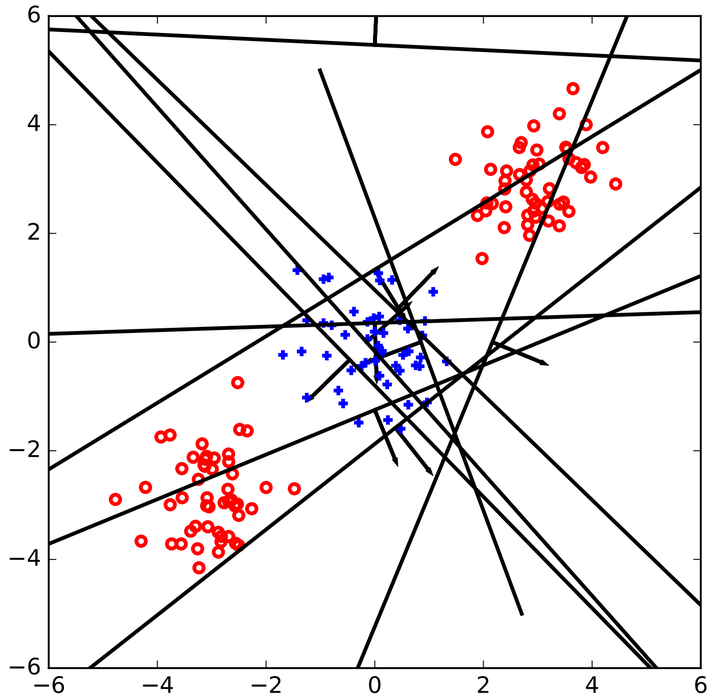
tanh activation



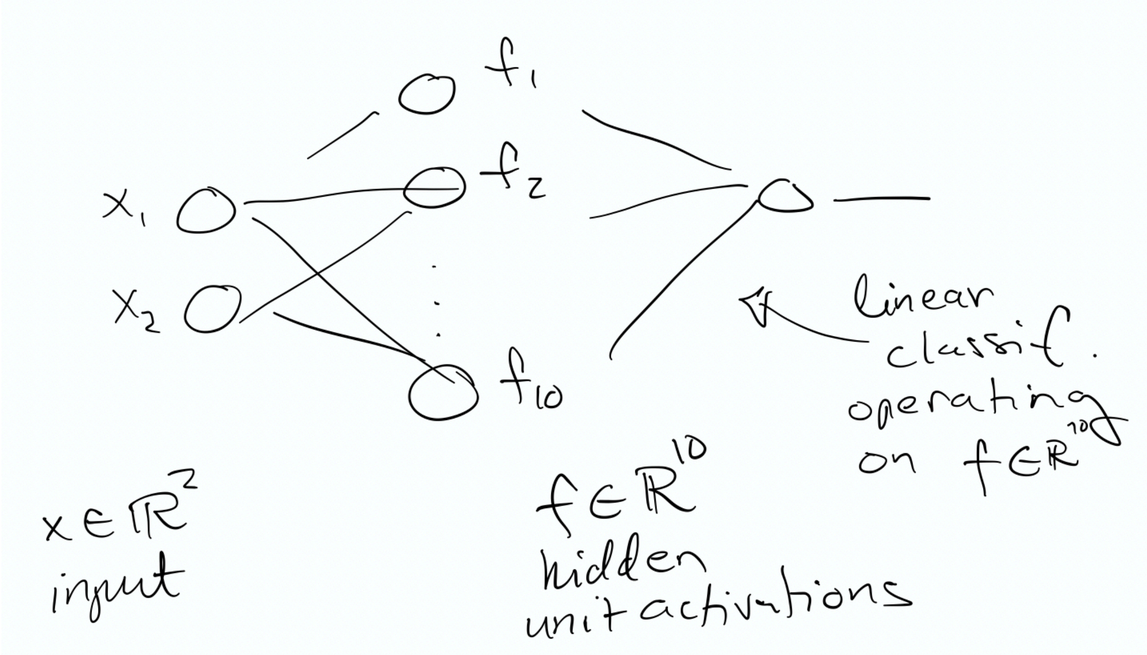
Random hidden units

- But a larger number of random hidden units gives a meaningful feature expansion

Hidden layer units



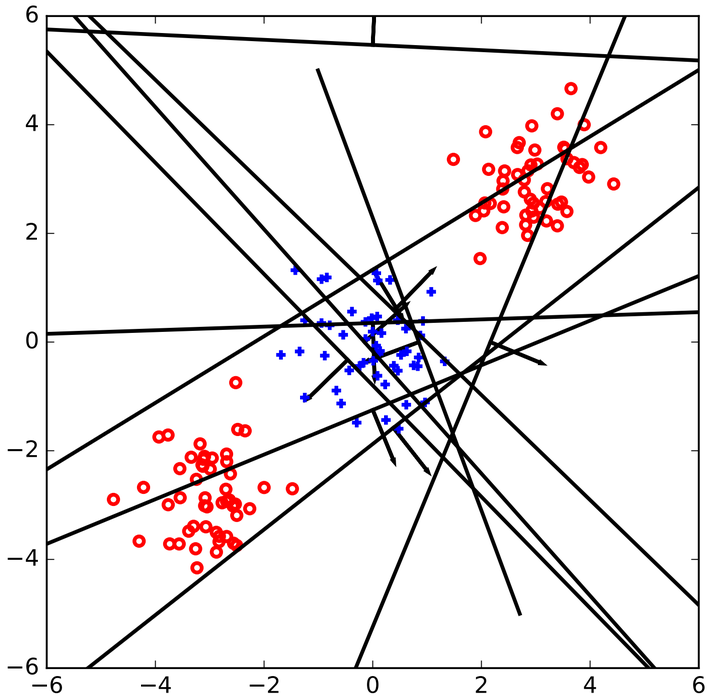
(10 randomly chosen units)



Random hidden units

- But a larger number of random hidden units gives a meaningful feature expansion

Hidden layer units



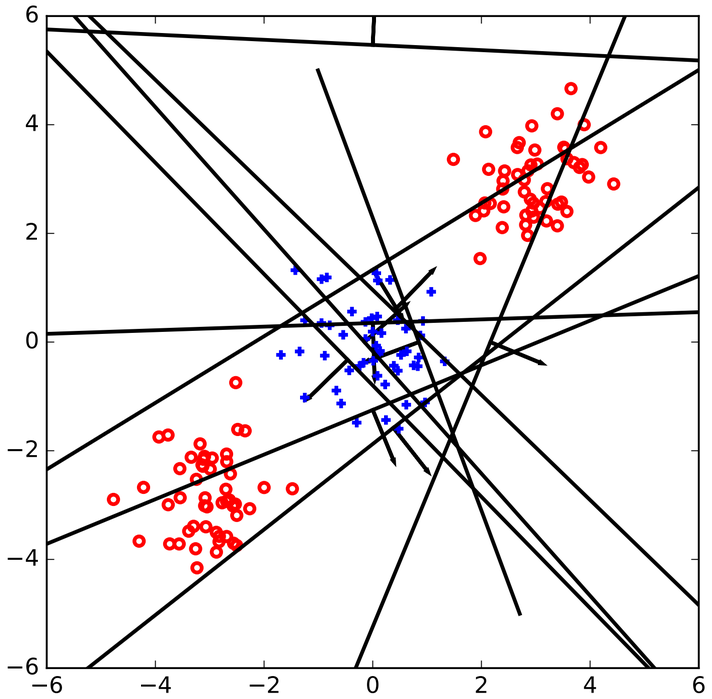
(10 randomly chosen units)

Are the points
linearly separable
in the resulting
10 dimensional space?

Random hidden units

- But a larger number of random hidden units gives a meaningful feature expansion

Hidden layer units



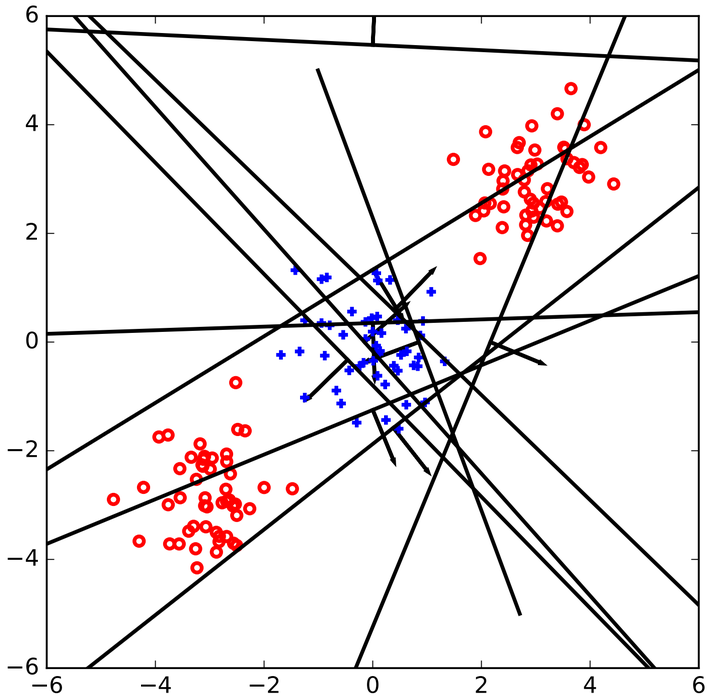
(10 randomly chosen units)

Are the points
linearly separable
in the resulting
10 dimensional space?

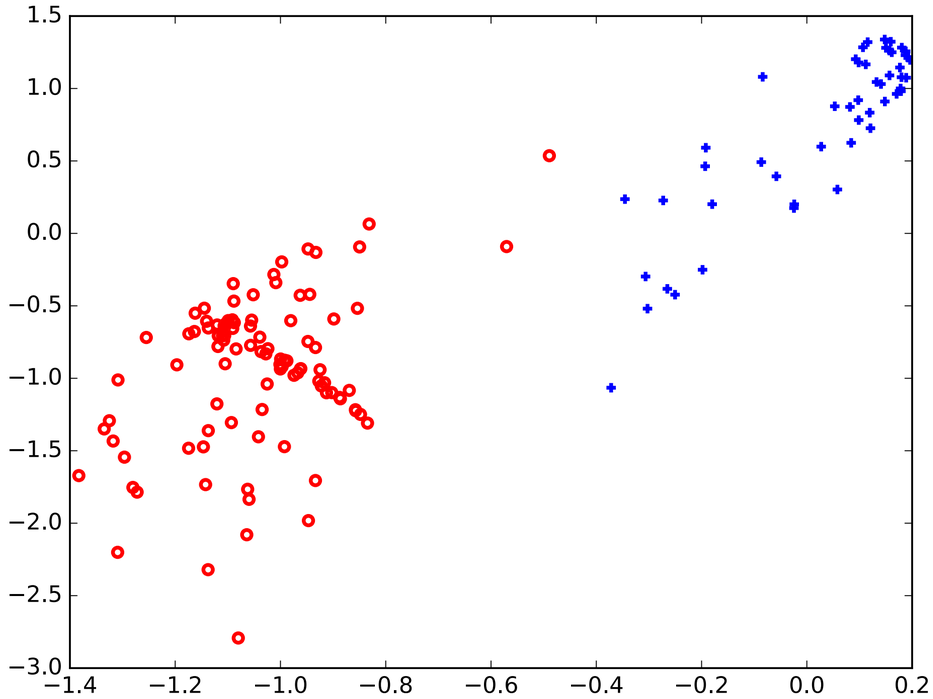
YES!

Random hidden units

Hidden layer units



(10 randomly chosen units)



what are the coordinates??