Machine Learning and Optimization

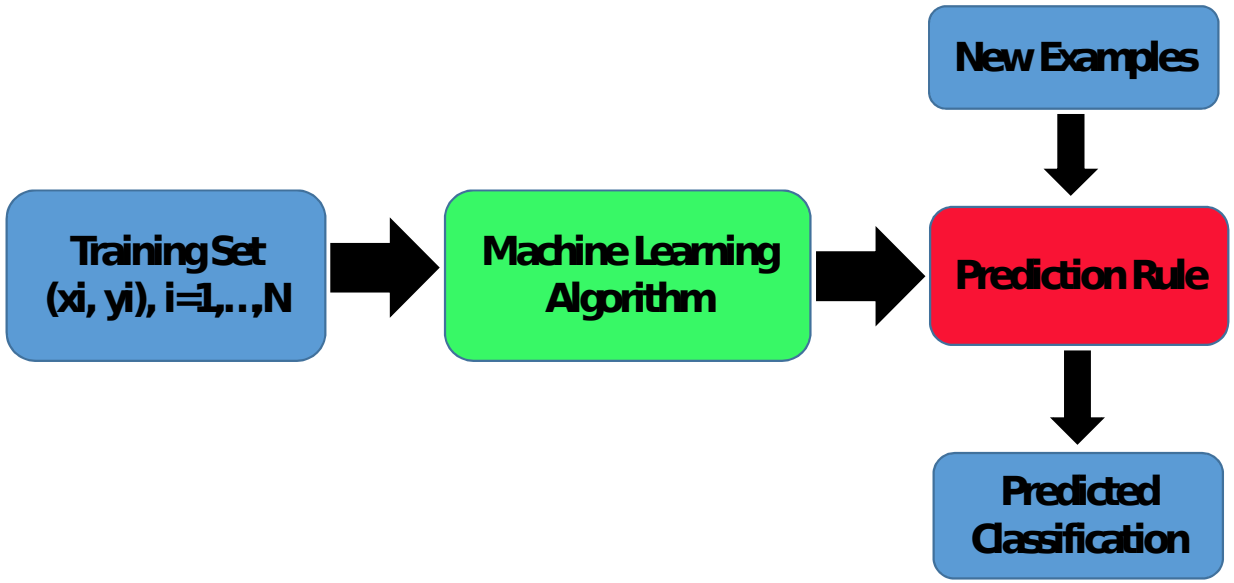
November 2017

Aida Khajavirad

Carnegie Mellon University
aida@cmu.edu

Machine Learning

- Machine learning studies computer algorithms for learning to do better in the future based on what was experienced in the past using some sort of observations or data; e.g. spam filtering, medical diagnosis, face detection.



- Supervised learning (regression and classification), Unsupervised learning (clustering, density estimation)

Parameter Estimation

- Maximum Likelihood Estimation (MLE): Let the observations $x_1, \ldots, x_m \in \mathbb{R}^n$ be described by a PMF parameterized by θ: $p(x; \theta)$. Then a MLE is a value of θ under which the observations are most likely. Assuming x_i are independent:

$$\theta^*_{\mathrm{ML}} = \mathrm{argmax}_\theta \, p(x_1, \ldots, x_m; \theta) = \mathrm{argmax}_\theta \prod_{i=1}^{m} p(x_i; \theta)$$

- Log-likelihood function: $\log p(x_1, \ldots, x_m; \theta) = \sum_{i=1}^{m} \log p(x_i; \theta)$

- Maximum a posteriori Estimation (MAP):

$$\theta^*_{\mathrm{MAP}} = \mathrm{argmax}_\theta \, p(\theta | (x_1, \ldots, x_m)) = \mathrm{argmax}_\theta \, p((x_1, \ldots, x_m) | \theta) p(\theta)$$

  The MAP estimation procedure allows us to inject our prior beliefs about parameter values into the new estimate.

Linear Regression

- Linear measurement model: $y_i = \theta^T x_i + v_i$, $i = 1, \ldots, m$ where $\theta \in \mathbb{R}^n$ is the vector of unknown parameters and v_i is IID noise with density $p(z)$.
- Gaussian noise: $p(z) = 1/\sqrt{2\phi\sigma^2}\exp(-z^2/(2\sigma^2))$. The log-likelihood is

$$-m/2\log(2\phi\sigma^2) - 1/\sigma^2 \sum_{i=1}^{m} (\theta^T x_i - y_i)^2$$

ML estimate with a Gaussian noise is the least square solution

$$\text{minimize}_\theta \sum_{i=1}^{m} (\theta^T x_i - y_i)^2$$

- Laplace noise $p(z) = 1/(2b)\exp -|z|/b$ The log-likelihood function is

$$-m\log(2b) - 1/b \sum_{i=1}^{m} |\theta^T x_i - y_i|$$

ML estimate with a Laplace noise is the l_1-norm solution (robust regression)

$$\text{minimize}_\theta \sum_{i=1}^{m} |\theta^T x_i - y_i|$$

Bayesian Linear Regression

- IID Gaussian priors give ridge regression (aka Tikhonov regularization):

$$\text{minimize}_\theta \sum_{i=1}^{m} (\theta^T x_i - y_i)^2 + \lambda ||\theta||_2^2$$

  for some $\lambda > 0$.

- IID Laplace priors give the LASSO estimate:

$$\text{minimize}_\theta \sum_{i=1}^{m} (\theta^T x_i - y_i)^2 + \lambda ||\theta||_1$$

  for some $\lambda > 0$. Can be equivalently stated in a constrained form:

$$\text{minimize}_\theta \sum_{i=1}^{m} (\theta^T x_i - y_i)^2 \text{ subject to } ||\theta||_1 \leq T,$$

  for some $T > 0$.

- Regularization controls the model complexity and avoids overfitting

Sparse Regression

- In many applications such as feature selection and compressed sensing, it is desirable to assume that the true regression coefficient θ is sparse.

- The best subset selection problem:

$$\text{minimize}_\theta ||y - X\theta||_2^2 \quad \text{subject to} \quad ||\theta||_0 \leq k,$$

  where l_0 (pseudo) norm of a vector θ counts the number of nonzeros in θ.

- The cardinality constraint makes the above problem NP-hard.

- Replace the nonconvex cardinality constraint by the convex constraint $||\theta||_1 \leq k$ and use LASSO as a heuristic to get sparse solutions.

$$\text{conv}\{(\theta, \gamma) : ||\theta||_\infty \leq 1, ||\theta||_0 \leq \gamma\} = \{(\theta, \gamma) : ||\theta||_\infty \leq 1, ||\theta||_1 \leq \gamma\}$$

Sparse Regression – MIQP Reformulation

- Suppose that $||\theta||_\infty \leq M_U$. Introduce binary variables $z_i \in \{0,1\}$ for all $i = 1, \ldots, n$. Then the best subset selection problem can be equivalently written as the following Mixed-Integer Quadratic Programm:

$$\begin{aligned} \text{minimize}_{\theta,z} \quad & \sum_{i=1}^{m} (\beta^T x_i - y_i)^2 \\ \text{subject to} \quad & -M_U z_i \leq \theta_i \leq M_U z_i \\ & \sum_{i=1}^{n} z_i \leq k \\ & z \in \{0,1\}^n \end{aligned}$$

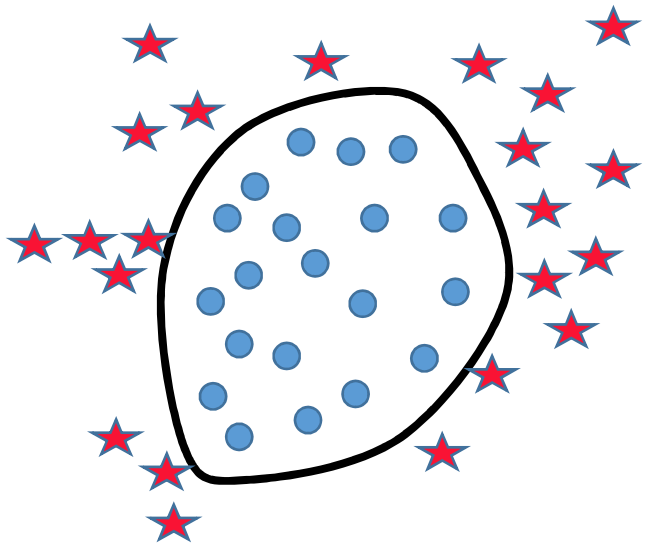
- Bertsimas et all show that the solutions provided by the MIP approach often significantly outperform Lasso in achieving sparse models.

Classification

- In pattern recognition and classification problems, we are given two sets of points in $\mathbb{R}^n$, $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots y_M\}$, and wish to find a function f (or a family of functions) such that

$$f(x_i) > 0,\ i = 1, \ldots, N, \quad f(y_i) < 0,\ i = 1, \ldots M$$

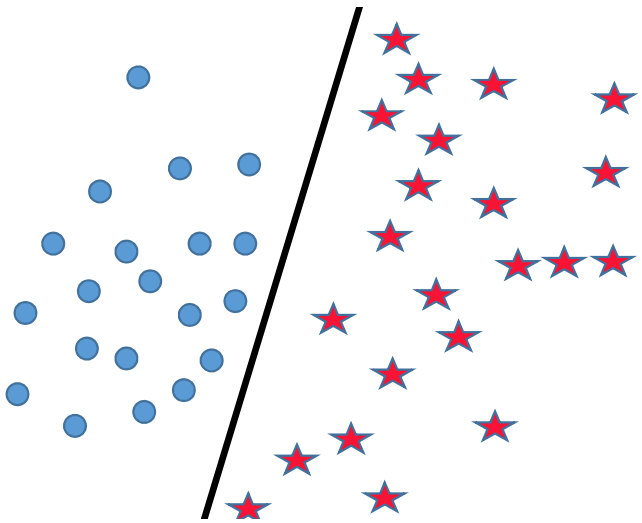
  If these inequalities hold, we say that f separates or classifies the two sets of points.



Linear Classification

- Separate two sets of points $\{x_1, \ldots, x_N\}$ and $\{y_1, \ldots, y_M\}$ by a hyperplane:

$$a^T x_i + b > 0,\ i = 1, \ldots, N, \quad a^T y_i + b < 0,\ i = 1, \ldots, M$$



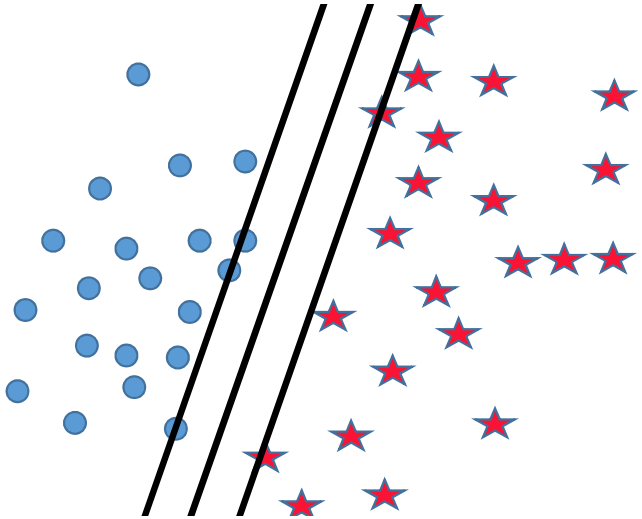
- These inequalities are homogenous in a and b and hence equivalent to:

$$a^T x_i + b \geq 1,\ i = 1, \ldots, N, \quad a^T y_i + b \leq -1,\ i = 1, \ldots, M$$

- We can find (a, b) by solving a linear optimization problem

Robust Linear Classification

- Separate the two sets of points by the maximum margin:



- The distance between the two hyperplanes $H_1 = \{z : a^T z + b = 1\}$ and $H_2 = \{z : a^T z + b = -1\}$ is $2/||a||_2$

$$\begin{array}{ll} \text{minimize}_{a,b} & ||a||_2 \\ \text{subject to} & a^T x_i + b \geq 1, \ i = 1, \ldots, N \\ & a^T y_i + b \leq -1, \ i = 1, \ldots, M \end{array}$$

Approximate Linear Classification

- Minimize the number of misclassified points (not tractable):

$$\begin{array}{ll} \text{minimize}_{a,b,u,v} & ||u||_0 + ||v||_0 \\ \text{subject to} & a^T x_i + b \geq 1 - u_i,\ i = 1, \ldots, N \\ & a^T y_i + b \leq -1 + v_i,\ i = 1, \ldots, M \\ & u \geq 0,\ v \geq 0 \end{array}$$

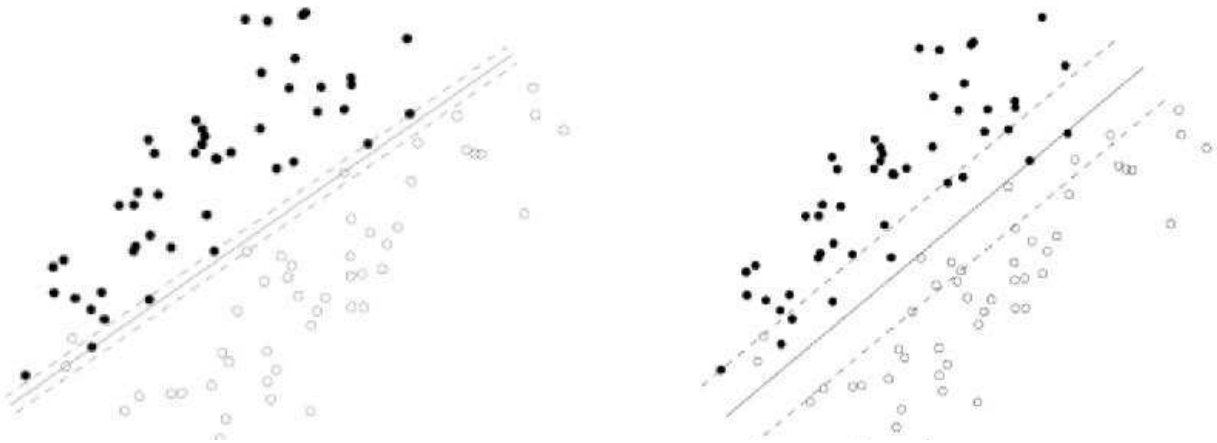
- Use the l_1-norm trick to obtain an LP:

$$\begin{array}{ll} \text{minimize}_{a,b,u,v} & \sum_{i=1}^{N} u_i + \sum_{i=1}^{M} v_i \\ \text{subject to} & a^T x_i + b \geq 1 - u_i,\ i = 1, \ldots, N \\ & a^T y_i + b \leq -1 + v_i,\ i = 1, \ldots, M \\ & u \geq 0,\ v \geq 0 \end{array}$$

Support Vector Machines (SVM)

- Address the trade-off between the size of the margin and the classification error:

$$\begin{aligned} \text{minimize}_{a,b,u,v} \quad & ||a||_2 + \gamma(\sum_{i=1}^{N} u_i + \sum_{i=1}^{M} v_i) \\ \text{subject to} \quad & a^T x_i + b \geq 1 - u_i, \ i = 1, \ldots, N \\ & a^T y_i + b \leq -1 + v_i, \ i = 1, \ldots, M \\ & u \geq 0, \ v \geq 0 \end{aligned}$$



SVM Classification

- Feature vectors $x_i \in \mathbb{R}^n$, $i = 1, \ldots, N$, binary labels $y_i \in \{-1, 1\}$. Linear classifier defined by $a \in \mathbb{R}^n$, $b \in R$: $f(x) = a^T x + b$. Perfect separation if $y_i(a^T x_i + b) \geq 1$:

$$\begin{aligned} \text{minimize}_{a,b,\xi} \quad & \frac{1}{2}||a||_2^2 + \gamma \sum_{i=1}^{N} \xi_i \\ \text{subject to} \quad & y_i(a^T x_i + b) \geq 1 - \xi_i, \; i = 1, \ldots, N \\ & \xi_i \geq 0, \; i = 1, \ldots, N \end{aligned}$$

- Define $K_{ij} = (y_i y_j) x_i^T x_j$. Then the dual is given by (a convex QP):

$$\begin{aligned} \text{minimize}_{\lambda} \quad & \frac{1}{2}\lambda^T K \lambda - \sum_{i=1}^{N} \lambda_i \\ \text{subject to} \quad & \sum_{i=1}^{N} y_i \lambda_i = 0 \\ & 0 \leq \lambda_i \leq \gamma, \; i = 1, \ldots, N \end{aligned}$$

The Kernel Trick

- By KKT conditions, at an optimal solution we have $a = \sum_{i=1}^{N} \lambda_i y_i x_i$. Thus, the classifier can be written as:

$$f(x) = \sum_{i=1}^{N} \lambda_i y_i (x_i^T x) + b$$

- A much more powerful classifier can be obtained, by lifting the feature vector x_i into a higher-dimensional space by a function $\phi : \mathbb{R}^n \to \mathbb{R}^t$ and classify in that space. Dual formulation remains the same by redefining K as:

$$K_{ij} = (y_i y_j) \phi(x_i)^T \phi(x_j),$$

  which gives the classifier $f(x) = \sum_{i=1}^{N} \lambda_i y_i \phi(x_i)^T \phi(x) + b$

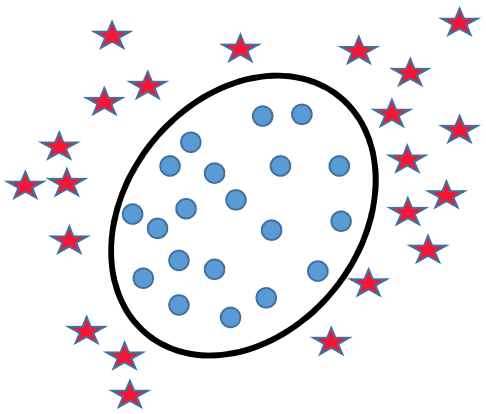
- Only need to compute inner products; instead of ϕ work with a kernel function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. If K is continuous, symmetric in arguments, and positive definite, there exists a Hilbert space and a function ϕ in this space such that $K(x, \bar{x}) = \phi(x)^T \phi(\bar{x})$.

The Kernel Trick

- Select a kernel k, form $K_{ij} = y_i y_j k(x_i, x_j)$, solve the dual to obtain λ and b, and use the classifier

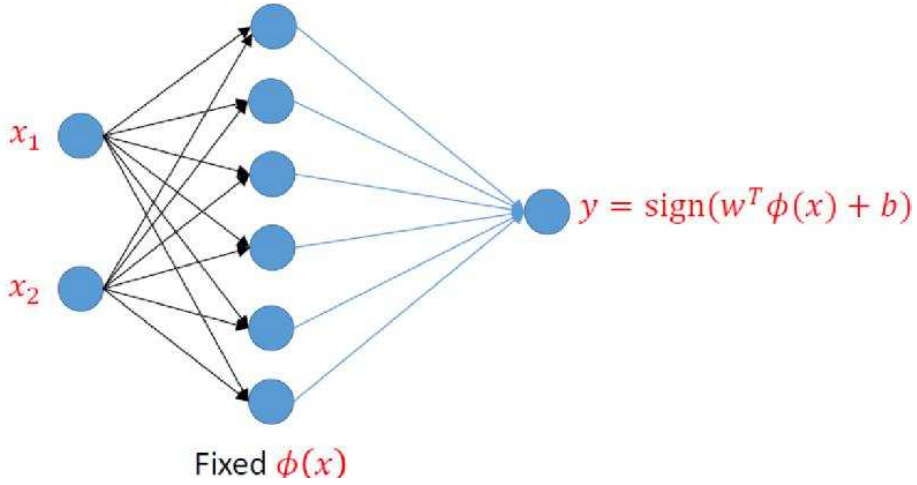
$$\sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + b$$

- Most popular kernels:
  - Linear: $k(x, \bar{x}) = x^T \bar{x}$
  - Gaussian: $k(x, \bar{x}) = \exp(||x - \bar{x}||^2)$
  - Polynomial: $k(x, \bar{x}) = (x^T \bar{x} + 1)^d$
- $\phi(x_1, x_2) = (x_1, x_2, x_1^2, x_1 x_2, x_2^2)$

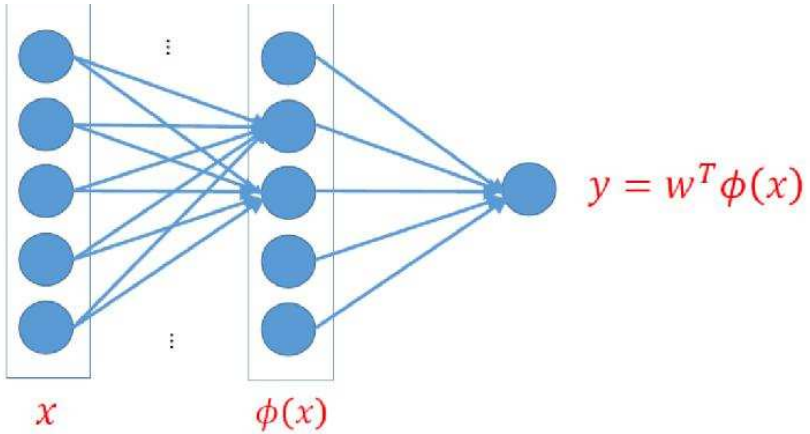


Neural Networks – Motivation

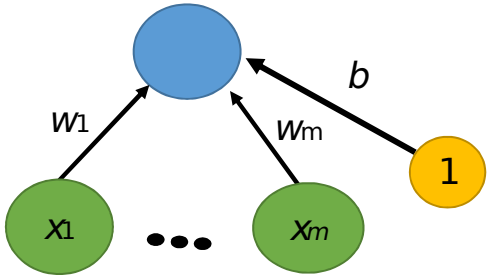
- Polynomial kernel SVM:



- Why don't we also learn $\phi(x)$?



Artificial Neuron



- Neuron pre-activation: $a(x) = b + w^T x$, where w are the connection weights and b is the neuron bias
- Neuron activation: $h(x) = g(b + w^T x)$, where g is called the activation function
- Common activation functions:
  - sigmoid $g(a) = 1/(1 + \exp(-a))$
  - hyperbolic tangent $g(a) = (\exp(2a) - 1)/(\exp(2a) + 1)$
  - rectified linear activation function $g(a) = \max\{0, a\}$

Single hidden-layer neural network

- Hidden layer pre-activation:

$$\mathbf{a}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}$$

$$\left(a(\mathbf{x})_i = b_i^{(1)} + \sum_j W_{i,j}^{(1)} x_j\right)$$

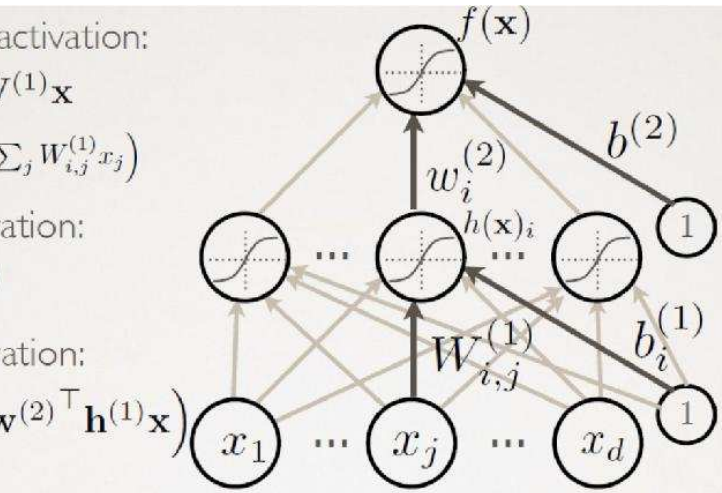
- Hidden layer activation:

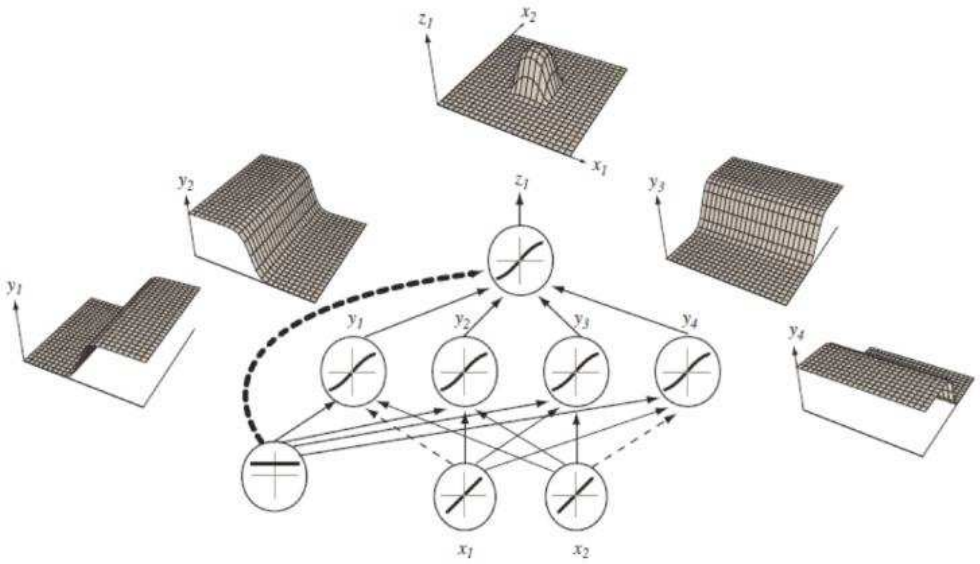
$$\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$$

- Output layer activation:

$$f(\mathbf{x}) = o\left(b^{(2)} + \mathbf{w}^{(2)\top}\mathbf{h}^{(1)}\mathbf{x}\right)$$

output activation function

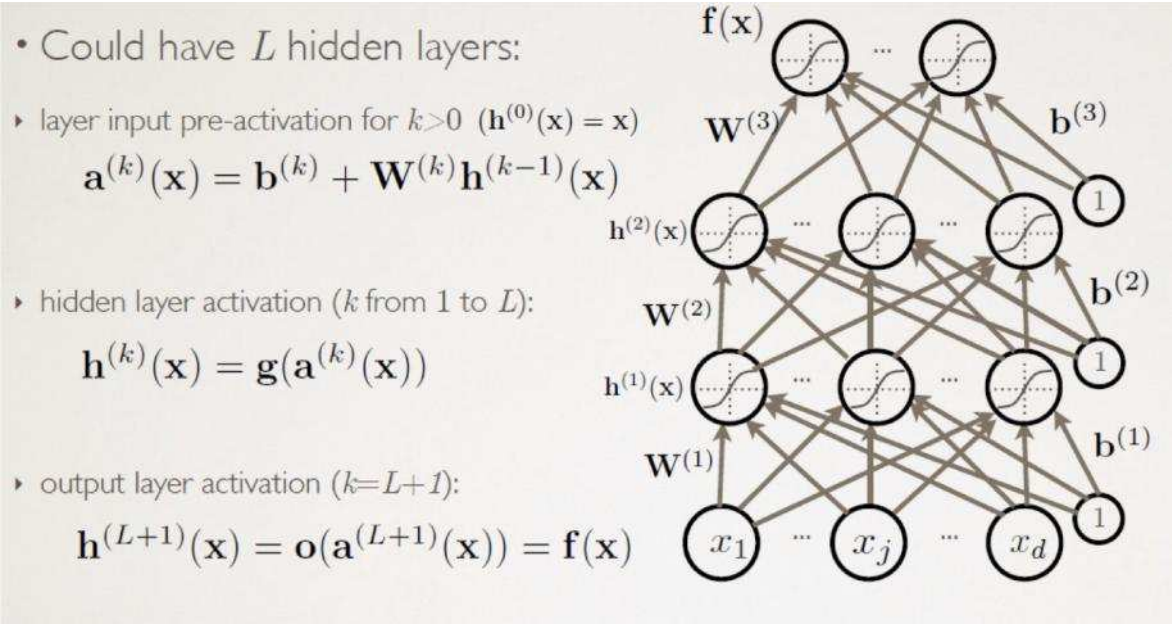




Multi-layer neural network

- Universal approximation theorem (Hornik, 1991): a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units"

- It does not mean there is a learning algorithm that can find the necessary parameter values!



Deep learning

- A deep architecture can represent certain functions exponentially more compactly

- Any Boolean function can be represented by a single hidden layer network; however, it might require an exponential number of hidden units

- There are Boolean functions which
  - require an exponential number of hidden units in the single layer case
  - require a polynomial number of hidden units if we can adapt the number of layers

- Training is hard! Heuristic methods such as stochastic gradient descent tend to work well in practice; many many success stories!