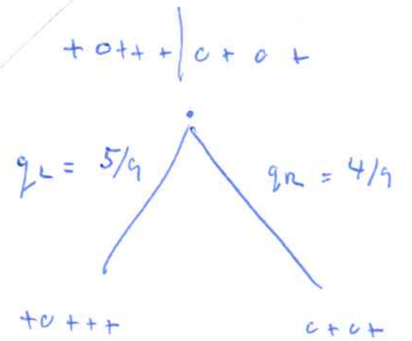
node of tree



q_L, q_R are proportions in left, right subtrees

p_L, p_R are left, right class distributions.

$p_L = (1/5, 4/5)$ $\quad p_R = (1/2, 1/2)$

Choose split that minimizes average impurity.

Say use H = entropy.

Minimize $\quad q_L H(p_L) + q_R H(p_R)$

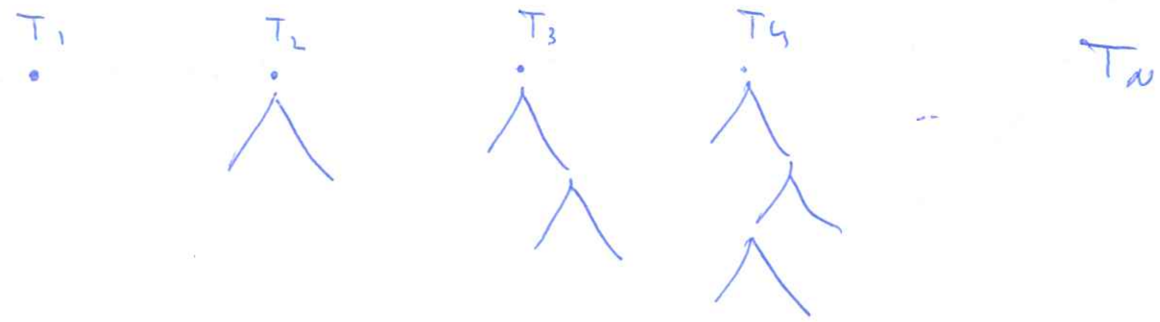
NB. Tree construction is <u>greedy</u> making <u>locally</u> optimal choices.

How deep to grow tree?

tree-overfit.r

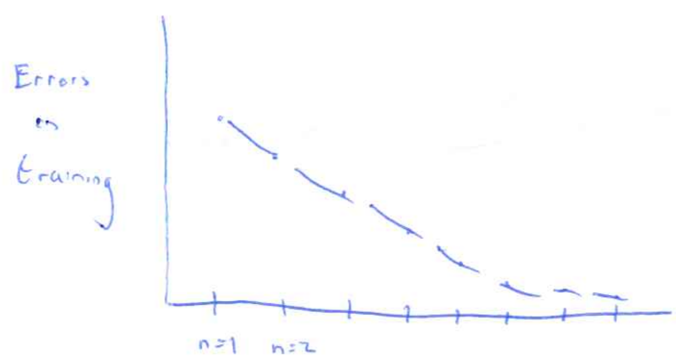
Overfitting

Let $T_1, T_2, \ldots T_N$ be successive trees grown during training.

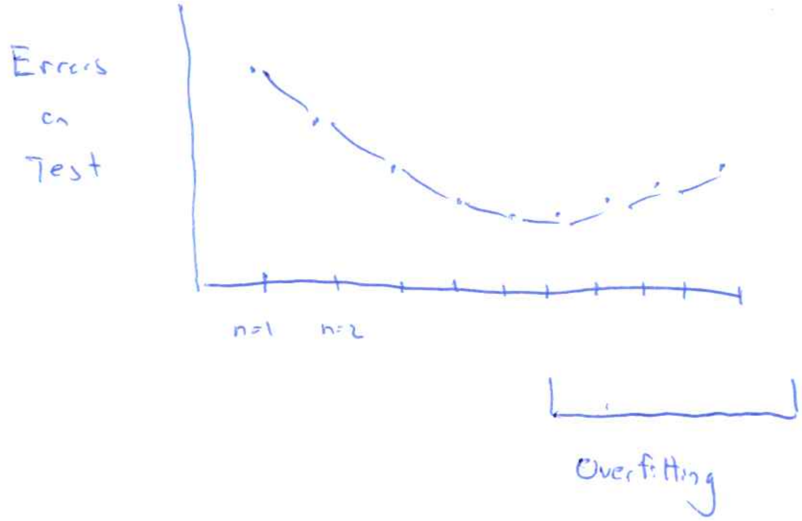


T_n has n terminal nodes $(|T_n| = n)$

The following graphs typical



More branches gives better result.



Performance improves for a while, then gets worse.

66 avoid overfitting?

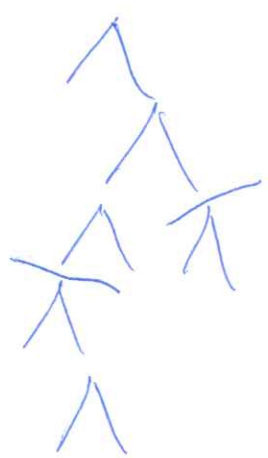
-ze complexity in choosing tree, T

Let $\alpha \in [0, \infty)$ be complexity parm. Define

$$R_\alpha(T) = \underbrace{R(T)}_{\text{errors on training}} + \underbrace{\alpha |T|}_{\text{charge } \alpha \text{ for each split}}$$

Grow initial tree (deep) and consider rooted subtrees

Root = all trees created by pruning initial tree.



Initial tree

$\in$ Root

Define

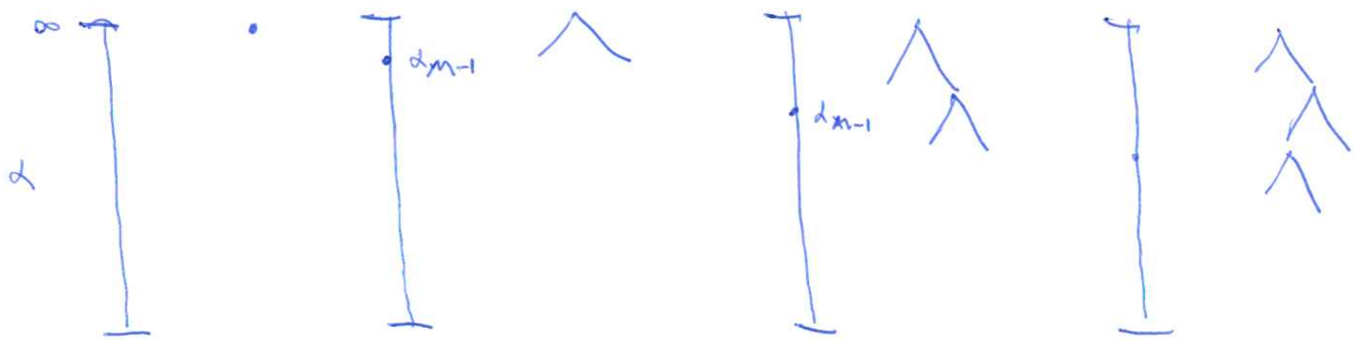
$$T_\alpha = \arg\min_{T \in \text{Root}} R_\alpha(T)$$

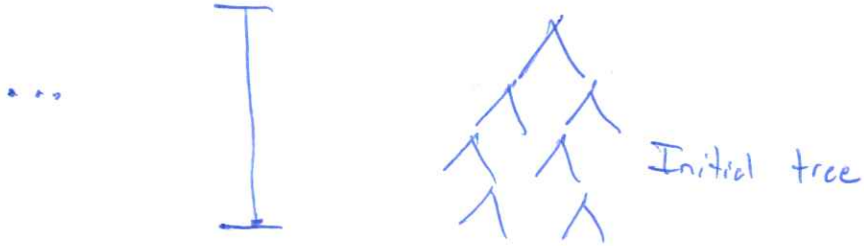
Note

① T_α is easy + cheap to construct

② The $\{T_\alpha\}$ are nested with T_0 = initial tree

T_∞ = null tree





Choose candidate complexity params

$\beta_1 = 0$

$\beta_2 = (\alpha_1 \alpha_2)^{1/2}$

$\beta_3 = (\alpha_2 \alpha_3)^{1/2}$

$\vdots$

$\beta_M = \infty$

Select β by cross validation

Cross Validation

Have method for training classification or regression algorithm.

Divide training set G, randomly into $G_1 \cdots G_N$ equal pieces.

Let $\bar{G}_n = \bigcup_{j \neq n} G_j$

CV estimates error rate of training method.

$$\widehat{\text{Error Rate}} = \frac{1}{|G|} \sum_{n=1}^{N} \text{Errors from } G_n \text{ when train on } \bar{G}_n$$

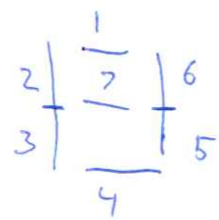
For CART let $\hat{ER}_{\beta_m}$ be estimated ER using CV with complexity parm β_m.

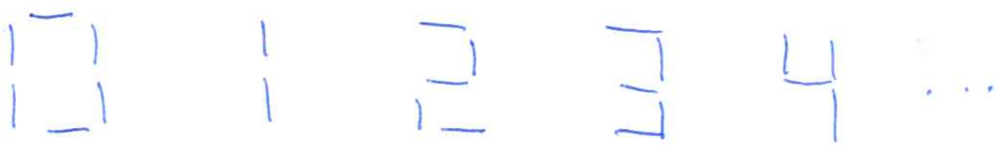
$$\beta^* = \arg\min_{\beta_m \in \{\beta_1 \cdots \beta_n\}} \hat{ER}_{\beta_m}$$

Now using all of G train tree with complexity parm β^*.

Digit Recognition

Can get 10 digits $0, \dots, 9$ with 7 possible "strokes"





$$z_0 = (1111110)$$

$$z_1 = (0000110)$$

$$z_2 = (1011011)$$

Create data as follows:

$c_i \in \{0, \dots, 9\}$ at random

$x_{i1} \cdots x_{i7} = z_{c_i}$ with randomly flipped bits

$x_{i8} \cdots x_{i24} =$ noisy bits.

Classify with digit_recognition.r

HW 5 Problem 1

"cosine" "trig" - "brontosaurus"

$$\begin{array}{c} \text{class } 1 \\ \text{class } 2 \\ \vdots \\ \text{class } k \end{array} \begin{pmatrix} q_{11} & \cdots & q_{1D} \\ q_{21} & \cdots & q_{2D} \\ & & \\ q_{k1} & \cdots & q_{kD} \end{pmatrix} = \underset{k \times D}{Q}$$

$$\begin{array}{c} \text{obs } 1 \\ \text{obs } 2 \\ \vdots \\ \text{obs } n \end{array} \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ \vdots & & & \\ x_{n1} & x_{n2} & \cdots & x_{nD} \end{pmatrix} = \underset{n \times D}{X}$$

a) $$\prod_{i=1}^{n} \sum_{k=1}^{k} \pi_k \prod_{d=1}^{D} q_{kd}^{x_{id}} (1 - q_{kd})^{1-x_{id}} = P(X \mid \pi, Q)$$

b) $$\gamma_{ik} = P(\text{class} = k \mid x_i) = \frac{\pi_k \prod_{d=1}^{D} q_{kd}^{x_{id}} (1 - q_{kd})^{1-x_{id}}}{\sum_{k'}}$$

c) $$\hat{\pi}_k = \frac{n_k}{n} \qquad n_k = \sum_i \gamma_{ik}$$

$$\hat{q}_{kd} = \frac{\text{count of class } k \text{ with } x_{id} = 1}{\#\text{ count of class } k} = \frac{\sum_{\{i : x_{id} = 1\}} \gamma_{ik}}{n_k} = \frac{\sum \gamma_{ik} x_{id}}{n_k}$$

d) $\log p(z, X; \pi, Q) = \log \prod_{i=1}^{n} \pi_{z_i} \prod_{d=1}^{D} q_{z_i d}^{x_{id}} (1-q_{z_i d})^{1-x_{id}}$

$$= \sum_{i=1}^{n} \log \pi_{z_i} + \sum_{d=1}^{D} x_{id} \log q_{z_i d} + (1-x_{id}) \log (1-q_{z_i d})$$

e) $\mathbb{E}_{\pi^{old}, Q^{old}} \log p(z, X; \pi, Q) =$

$$\underbrace{\sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_{ik} \log \pi_k}_{\pi_k \text{ only appear here}} + \sum_{d=1}^{D} x_{id} \sum_{k=1}^{K} \gamma_{ik} \log q_{kd} + (1-x_{id}) \sum_{k=1}^{K} \gamma_{ik} \log (1-q_{kd})$$

f) $\hat{\pi}^{new} = \underset{\substack{\pi \\ \text{s.t. } \sum \pi_k = 1}}{\arg\max} \sum_{k=1}^{n} n_k \log \pi_k$

$$\frac{n_k}{\pi_k} = \lambda \implies \frac{n_k = \lambda \pi_k}{+ \quad n = \lambda}$$

$$\frac{n_k}{\pi_k} = n \implies \pi_k^{new} = \frac{n_k}{n} \quad (\text{as always})$$

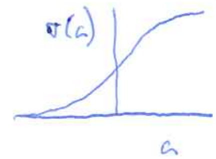
Neural Networks

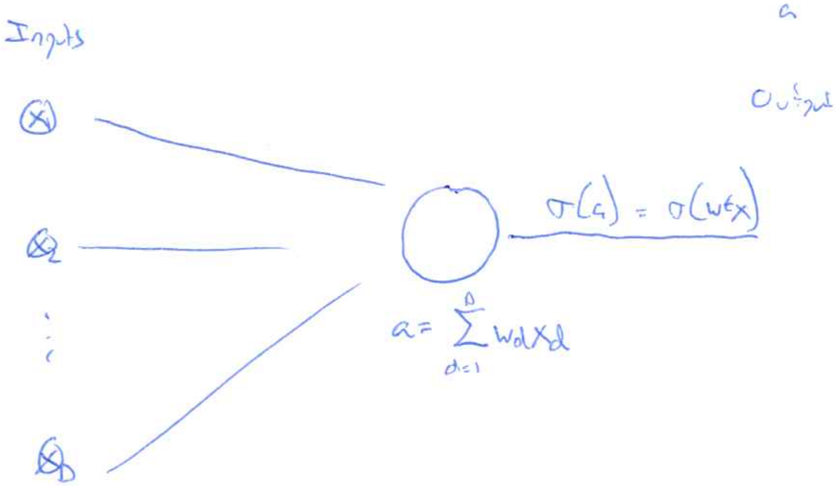
Recall 2-class logistic regression. Have multivariate ~~xxxx~~ vector $x \in \mathbb{R}^D$

Model

$$P(\text{class} = 1 \mid x) = \sigma\left(\sum_{d=1}^{D} w_d x_d\right) = \sigma(w^t x)$$

where $\sigma(a) = \dfrac{1}{1+e^{-a}}$

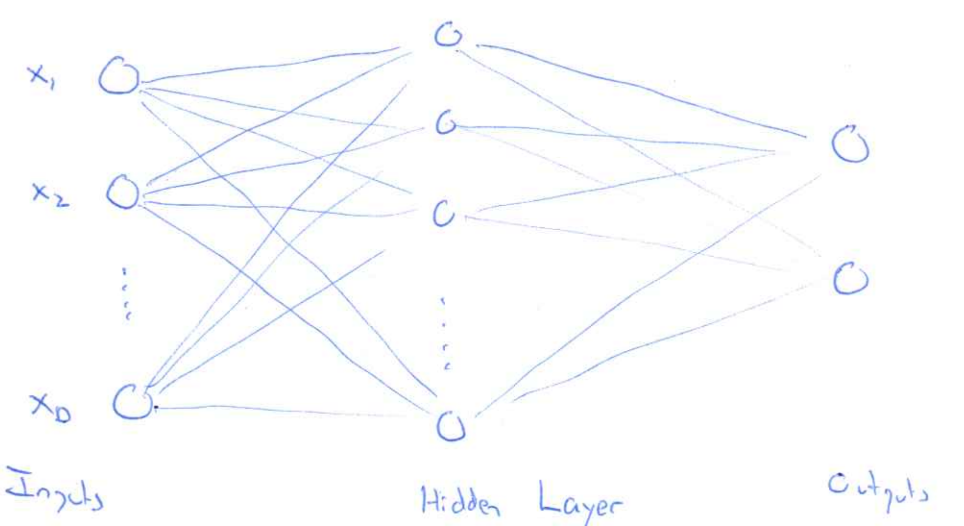




Output is nonlinear xform of linear comb. of inputs.

Neural Network generalizes to multiple logistic-type nodes

Typical Network



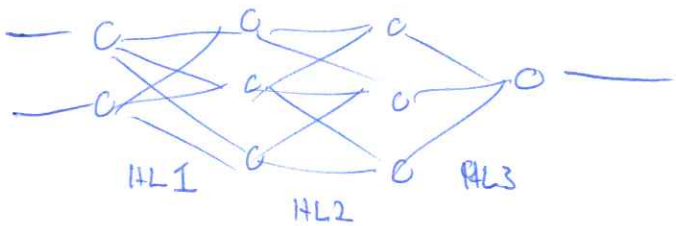
~~Let $z_1, \ldots, z_M$ be inputs to any node~~

Network Topology

Can have

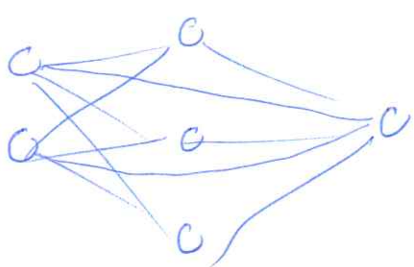
1) Any # input nodes, hidden nodes, outputs

2) Any # of hidden layers



3) Layers need not be completely connected, but must be "feed forward". That is, if $D(v_i)$ is the depth of vertex (node) v_i and $v_i \to v_j$ in network, then $D(v_j) > D(v_i)$

[Graph cannot have cycles]



Connections can skip over layers.

Activation

At each hidden node v_j, the activation $a_j = \sum_{i=1}^{D} w_{ji} z_i$ where $\{z_i\}$ are inputs. (First layer has inputs $\{x_i\}$)

The activation is governed by "weights" w_{ji}. These are parameters of NN. (Learned during training)

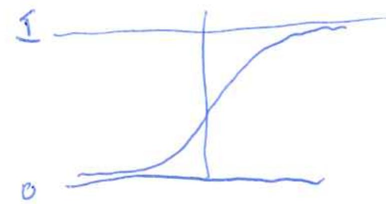
Nonlinearity

At node v_j have output $z_j = h(a_j) = h\left(\sum_{i=1}^{D} w_{ji} z_i\right) = h(w_j^t z)$

Common choices for h:

(1) $h(a) = \sigma(a) = \dfrac{1}{1+e^{-a}}$

(2) $h(a) = \tanh(a) = \dfrac{e^a - e^{-a}}{e^a + e^{-a}}$

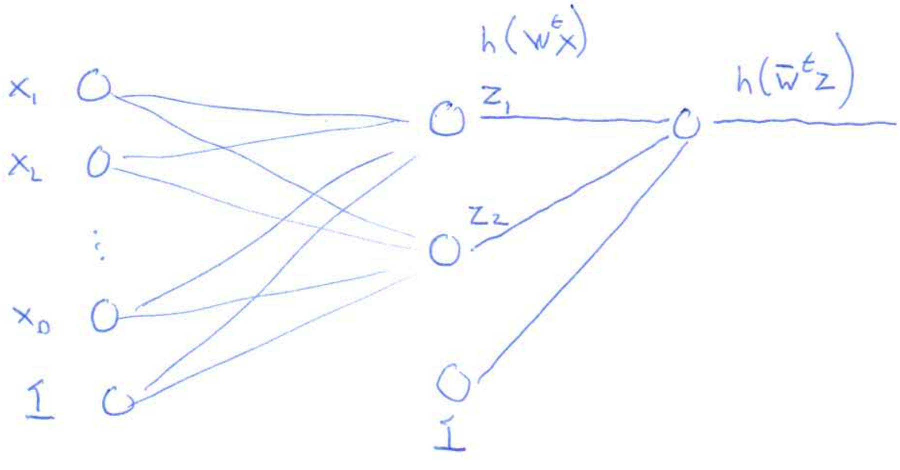


Bias

Sometimes activation written as

$$a_j = w_j^t z + \underbrace{w_{j0}}_{\text{bias term}}$$

As in regression can add 1 to input



Then outputs can be written $h(w^t x)$ or $h(w^t z)$

Output of Neural Network

Form of output depends on goal of network

3 cases: Regression (continuous output)

2-class classification

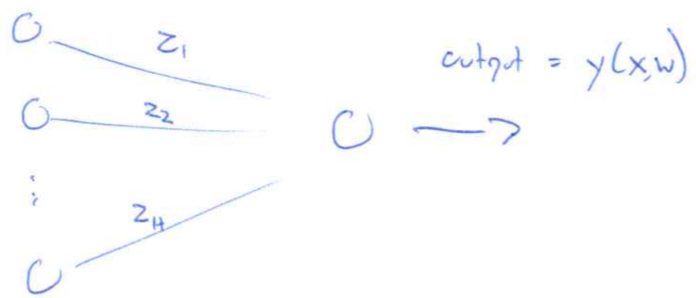
K-class classification

① Regression (Continuous output [predict lifetime of object])

Denote output of NN by $y(x, w)$

inputs weights

Let $z_1 \cdots z_H$ be inputs to final node



Model $y(x,w) = \sum_{i=1}^{H} w_i z_i = w^t z$

NB No non linear (h) trans on output.

(2) 2-class classification

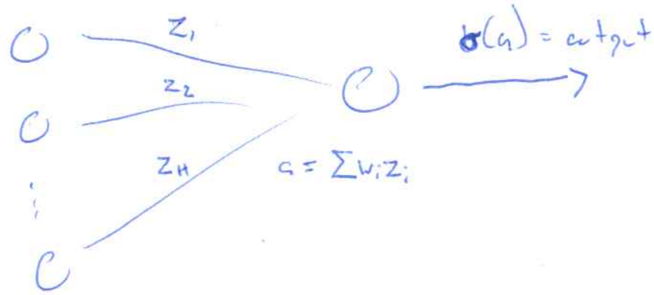
$z_1 \cdots z_H$ inputs to final node

Output: $y(x,w) = P(C=1 \mid x,w)$

<u>Model</u>

$$y(x,w) = \sigma\left(\sum_i w_i z_i\right) = \sigma(w^t z)$$

$$\sigma(a) = \frac{1}{1+e^{-a}}$$ as in logistic regression.



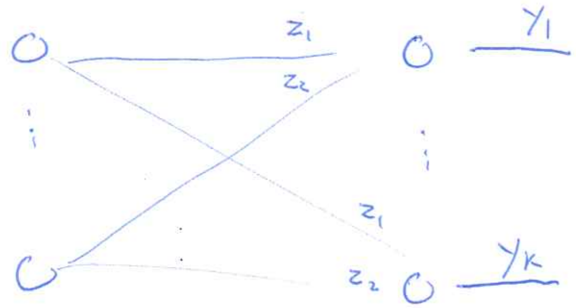
(3) k-class classification

Output $y_k(x,w) = P(C=k \mid x,w)$

$\sum_k y_k(x,w) = 1$

$0 \le y_k(x,w) \le 1$

$$y_k(x,w) = \frac{e^{w_k^t z}}{\sum_{k'=1}^{k} e^{w_{k'}^t z}}$$ (softmax)



Universal Approximation

Neural Net $y(x, w)$ represents function.

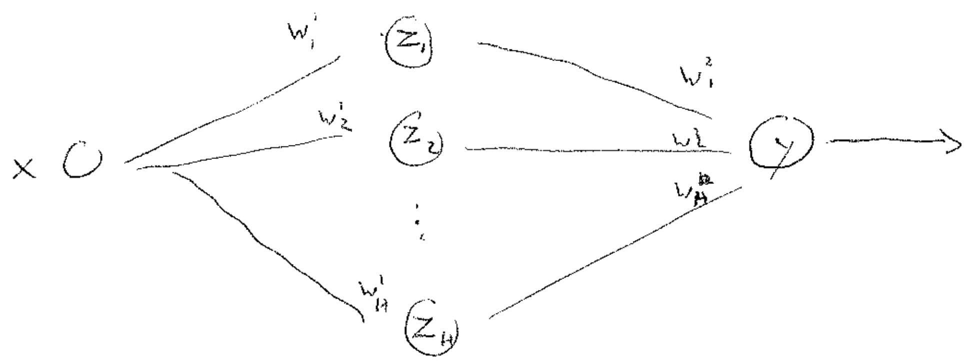
input

output

Loosely speaking, NN can approx any function with arbitrary precision.

More precisely, suppose $x \in [0, 1]$ with $t(x)$ "target" fn (what we want to approx.)

Let $y(x, w)$ be NN with H hidden units as follows.



$$y(x, w) = y(x, w_1^1 \cdots w_H^1, w_1^2 \cdots w_H^2) = \sum_{j=1}^{H} w_j^2 \, h(w_j^1 x)$$

Then

$$y(x, w) \xrightarrow[H \to \infty]{} t(x)$$

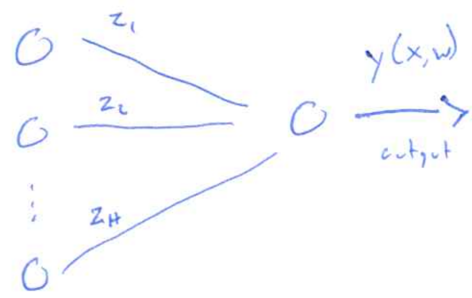
if weights $\{w_1^1 \cdots w_H^1, w_1^2 \cdots w_H^2\}$ chosen properly.

Note Other methods are also universal approximators

1. GMM can approx any dist as # of mixture comps (k) increases
2. CART used for regression (as on exam) can approx any fn.

Training the Network

Consider 2-class classification



$$y(x,w) = P(\text{Class} = 1 \mid x) = \sigma\left(\sum w_h z_h\right) = \sigma(a)$$

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

Have data $(x_1, t_1), (x_2, t_2) \dots (x_n, t_n)$ $\quad t_i \in \{0, 1\}$

$$P(t_1 \dots t_n \mid x_1 \dots x_n, w) = \prod_{i=1}^{n} P(\text{Class} = 1 \mid x_i, w)^{t_i} \left(1 - P(\text{Class} = 1 \mid x_i, w)\right)^{1-t_i}$$

$$= \prod_{i=1}^{n} y(x_i, w)^{t_i} \left(1 - y(x_i, w)\right)^{1-t_i}$$

$$E(w) = -\log P(t_1 \dots t_n \mid x_1 \dots x_n, w) = -\sum_{i=1}^{n} t_i \log y(x_i, w) + (1-t_i) \log(1 - y(x_i, w))$$

$$\doteq \sum_{i=1}^{n} E_i(w)$$

$$\frac{dE_i(w)}{da} = \frac{-t_i}{y(x_i,w)} \frac{dy(x_i,w)}{da} + \frac{1-t_i}{1-y(x_i,w)} \frac{dy(x_i,w)}{da}$$

$$= \left(\frac{1-t_i}{1-y(x_i,w)} - \frac{t_i}{y(x_i,w)}\right) y(x_i,w)\left(1 - y(x_i,w)\right)$$

$$= (1-\cancel{t_i})\, y(x_i,w) - t_i\left(1 - y(\cancel{x_i},w)\right)$$

$$= y(x_i,w) - t_i$$

$$= y_i - t_i$$

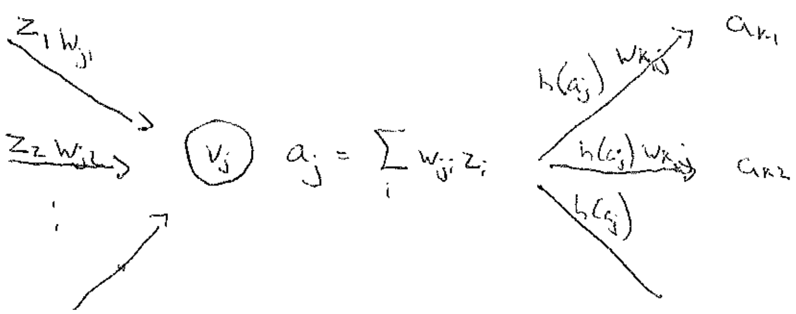
Gradient Descent by Backpropagation

NN parametrized by weights : w $\quad y(x, w)$

Want to minimize error $E(w)$

$$E(w) = \sum_{n=1}^{N} \underbrace{E_n(w)}_{\text{error for single observation}} \Longrightarrow \nabla_w E(w) = \sum_{n=1}^{N} E_n(w)$$

Will focus on $\nabla E_n(w) = \left\{ \dfrac{\partial E_n}{\partial w_{ji}} \right\}_{\text{all connections } i \to j}$



E_n depends on w_{ji} only through a_j

$$E_n(w_{ji}, \dots) = E_n(a_j(w_{ji} \dots), \dots)$$

$$\frac{\partial E_n}{\partial w_{ji}} = \underbrace{\frac{\partial E_n}{\partial a_j}}_{\delta_j} \frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i$$

$E_n(a_j, \dots) = \cancel{E_n}$

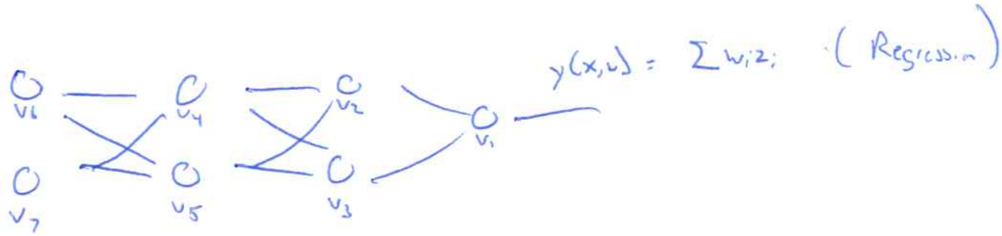
$= E_n(a_{k_1}(a_j), a_{k_2}(a_j), \dots)$

E_n depends on a_j only through $\{a_k : j \to k\}$

$$\delta_j = \frac{\partial E_n}{\partial a_j} = \sum_{k: j \to k} \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} = \sum_{k: j \to k} \delta_k w_{kj} h'(a_j) = h'(a_j) \sum_{k: j \to k} \delta_k w_{kj}$$

NB We develop δ_j in terms nodes "downstream" of v_j. This is backpropagation.

An Example



$$\delta_1 = \frac{\partial E_n}{\partial a_1} = (y_n - t_n)$$

$$\delta_2 = w_{12}\,\delta_1\, h'(a_2)$$

$$\delta_3 = w_{13}\,\delta_1\, h'(a_3)$$

$$\delta_4 = (w_{24}\,\delta_2 + w_{34}\,\delta_3)\, h'(a_4)$$

$$\delta_5 = (w_{25}\,\delta_2 + w_{35}\,\delta_3)\, h'(a_5)$$

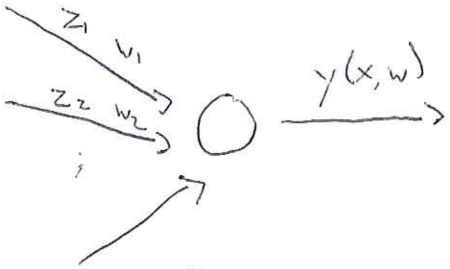
$$\frac{\partial E_n}{w_{12}} = z_2\,\delta_1$$

$$\frac{\partial E_n}{\partial w_{13}} = z_3\,\delta_1$$

$$\frac{\partial E_n}{\partial w_{24}} = z_4\,\delta_2$$

$$\frac{\partial E_n}{\partial w_{46}} = x_6\,\delta_4$$

Regression Case



$y(x,w) = \sum w_j z_j = a$ (no nonlinear h)

final node

For regression use sq error as ~~fitness~~ objective fn. Have $(x_1, t_1) \dots (x_N, t_N)$

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} \left(y(x_n, w) - t_n \right)^2$$

$$E_n(w) = \tfrac{1}{2}(y_n - t_n)^2$$

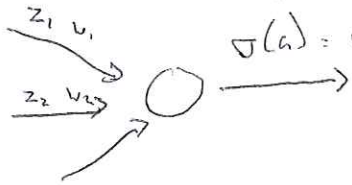
$y_n = y(x_n, w)$

$$= \frac{1}{2}\sum_{n=1}^{N} (y_n - t_n)^2 = \sum_{n=1}^{N} E_n(w)$$

As before differentiate E_n wrt. final activation a

$$\frac{dE_n}{da} = y_n - t_n$$

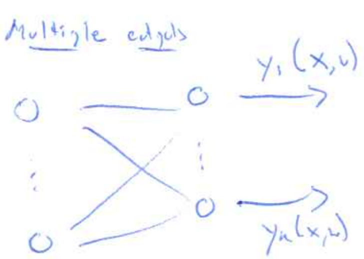
Note This is exactly what we get for 2-class classification



$\sigma(a) = \sigma(\sum w_j z_j)$

Even though had different objective fn. (log like) and different h fn.

Multiple outputs



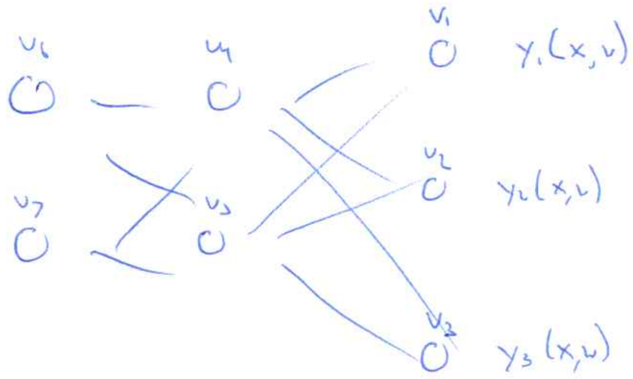
$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K} \left(y_k(x_n, w) - t_{nk} \right)^2 = \sum_{n=1}^{N} E_n(w)$$

$$E_n(w) = \frac{1}{2}\sum_{k=1}^{K} \left(y_k(x_n, w) - t_{nk} \right)^2$$

$$\cancel{\tfrac{1}{2}\sum_k (y_k(x_n,w) - t_{nk})^2}$$

$$\frac{\partial E_n}{\partial a_k} = y_k(x_n, w) - t_{nk}$$

Regression Ex



$$\delta_1 = y_1(x_n, w) - t_{1n}$$

$$\delta_2 = y_2(x_n, w) - t_{2n}$$

$$\delta_3 = y_3(x_n, w) - t_{3n}$$

$$\delta_4 = h'(a_4) \sum_{k=1}^{3} w_{k4} \delta_k$$

$$\delta_5 = h'(a_5) \sum_{k=1}^{3} w_{k5} \delta_k$$

$$\frac{\partial E_n}{\partial w_{14}} = z_4 \delta_1$$

$$\frac{\partial E_n}{\partial_{46}} = x_6 \delta_4$$

Regularization of Neural Nets

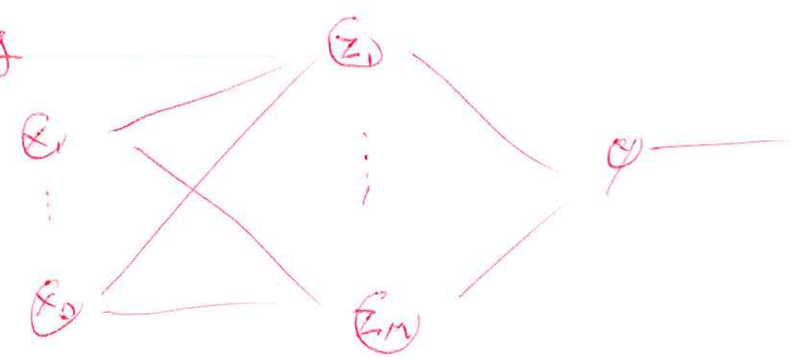
NN's can have many hidden nodes (= many parameters (= weights))
This can lead to overfitting.
How to address this?

① Cross Validation

Have separate validation set $(x_i^v, t_i^v) \dots$ not used in training.

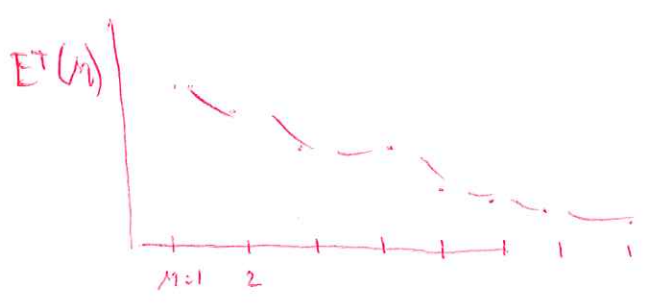
Suppose network has M hidden nodes with known topology (connectivity structure) but M unknown.

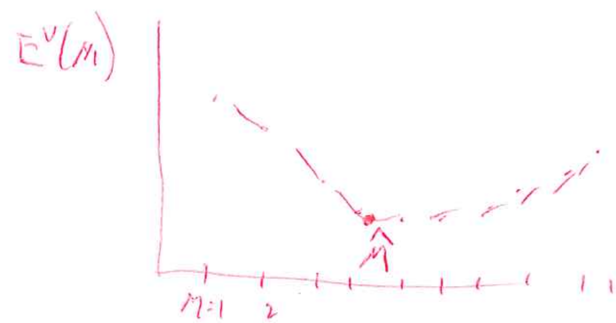
Eg



For $M = 1, 2, \dots$ train network using backprop.

Let ~~…~~ $E^T(M)$ be error on training set after training with M.





Typically $E^T(M)$ decreasing in M.

Let $E^V(M)$ be error on validation after training.

Chose $\hat{M}^*$ by $\hat{M} = \arg\min_M E^V(M)$

② Early Stopping

Since NN training is gradient descent, training iterative.

Let $y(x, w^n)$ $n = 1, 2, \ldots$

be network (function estimate) after n iterations of grad. descent.

Let $E^v(n)$ be error on separate validation set.

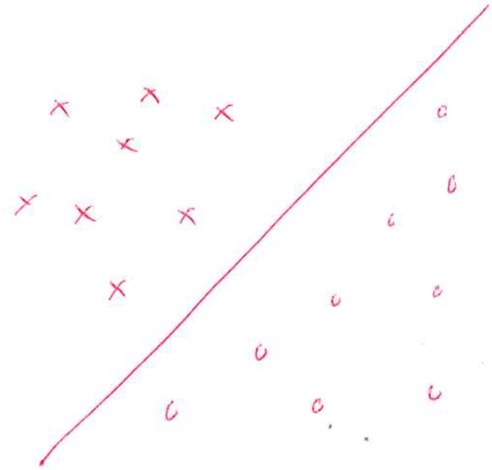
Stop training when $E^v(n)$ starts to increase.

Support Vector Machines

Consider 2-class classification problem on D-dimensional data

$x_1, x_2, \cdots x_N$

$\{x : w^t x + b = 0\}$



If $D = 2$ (as in figure) and $w \in \mathbb{R}^D$, $b \in \mathbb{R}$

$\{x : w^t x + b = 0\}$ is a line

If $D = 3$

$\{x : w^t x + b = 0\}$ is a plane

If $D > 3$

$\{x : w^t x + b = 0\}$ is "hyperplane".

~~We seek $w \in \mathbb{R}^D$, $b \in \mathbb{R}$~~ s.t.

Can view w, b as classifier. Suppose classes are $\{+1, -1\}$

If $w^t x + b > 0$ classify as $+1$

If $w^t x + b < 0$ classify as -1

We seek w, b that perfectly classify $x_1, x_2 \cdots, x_N$. That is suppose classes are $t_1 \cdots t_N$ $t_n \in \{-1, +1\}$. Seek w, b s.t.

$$t_n (w^t x_n + b) > 0 \qquad n = 1 \cdots N.$$

Note:

May not be possible to classify perfectly, but can consider xform $x \longrightarrow \phi(x)$ (feature vector) s.t.

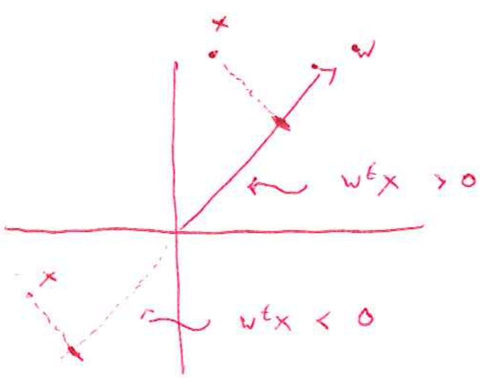
$$t_n (w^t \phi(x_n) + b) > 0 \qquad n = 1 \cdots N.$$

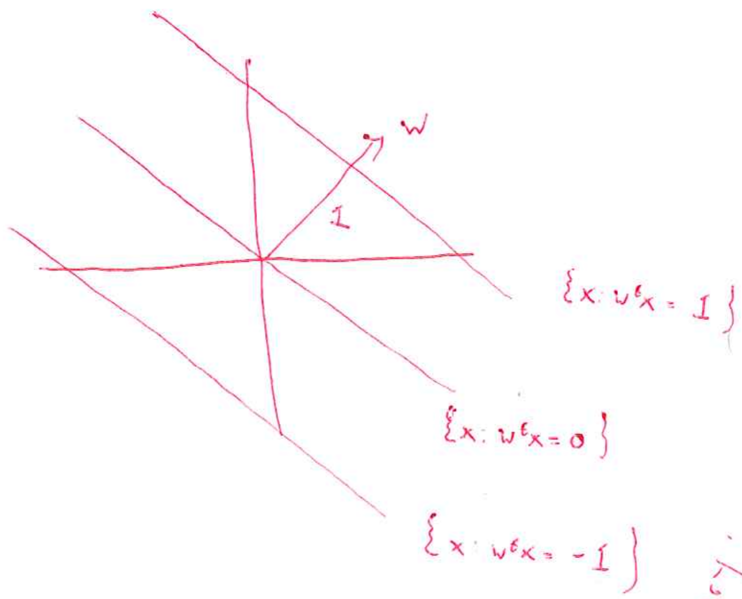
Geometry

$w^t x = \sum_{i=1}^{D} w_i x_i$ is <u>inner</u> <u>product</u> between w and x

$$(\cdots w^t \cdots) \begin{pmatrix} \vdots \\ x \\ \vdots \end{pmatrix}$$

If $\|w\| = \left(\sum w_i^2\right)^{1/2} = 1$ $w^t x$ gives signed length of proj of x onto w.





① Suppose $\{x : w^t x = 0\}$ is decision boundary

then $|w^t x|$ gives distance of x from decision boundary

② Suppose $\{x : w^t x + b = 0\}$ is decision bdry.

then $|w^t x + b|$ gives dist of x from decision bdry.

③ Have $(x_1, t_1) \dots (x_N, t_N)$ $t_n \in \{-1, +1\}$.

Suppose have w, b that perfectly classify $x_1 \cdots x_N$.

That is, $t_n = \begin{cases} +1 & \text{if } w^t x_n + b > 0 \\ -1 & \text{if } w^t x_n + b < 0 \end{cases}$

Then ~~shall~~ ~~to~~

For $x_1 \dots x_n$ $t_n (w^t x_n + b) > 0$

so $t_n (w^t x_n + b)$ gives dist. to bdry.

(note no abs value!)

What if $\|w\| \neq 1$?

Can still classify according to sign of $w^t x + b$

$$\text{Class}(x) = \begin{cases} +1 & w^t x + b > 0 \\ -1 & w^t x + b < 0 \end{cases}$$

Note that if $\alpha > 0$, scaling $w^t x + b$ by α gives equivalent classifier

$$\text{class}(x) = \begin{cases} +1 & \alpha(w^t x + b) > 0 \\ -1 & \alpha(w^t x + b) < 0 \end{cases}$$

Suppose we choose α s.t. $\|\alpha w\| = 1$. That is $\alpha = \frac{1}{\|w\|}$.

Then $\|\alpha w\| = \left\| \frac{w}{\|w\|} \right\| = \frac{1}{\|w\|} \|w\| = 1$

$$\text{Class}(x) = \begin{cases} +1 & \frac{w^t}{\|w\|} x + \frac{b}{\|w\|} > 0 \\ -1 & \frac{w^t}{\|w\|} x + \frac{b}{\|w\|} < 0 \end{cases}$$

Thus ~~, the distance of x to bdry given by~~

$$\cancel{\frac{w^t}{\|w\|} x + \frac{b}{\|w\|}}$$

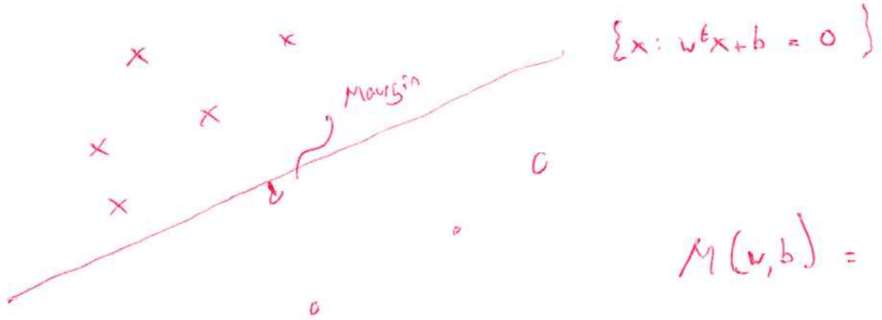
$(\|w\| \neq 1)$

Thus if w, b correctly classify all $x_1 \cdots x_n$ then dist. of x_n to bdry given by

$$t_n \left(\frac{w^t}{\|w\|} x_n + \frac{b}{\|w\|} \right) = t_n \left(\frac{w^t x_n + b}{\|w\|} \right)$$

Margin

If w, b classify all $x_1 \cdots x_N$ correctly the margin of w, b is the minimum dist to bdry



$$M(w,b) = \min_{n=1\cdots N} t_n \left(\frac{w^t x_n + b}{\|w\|} \right)$$

Support Vector Machine (SVM)

The SVM seeks w, b giving maximum margin.

$$(\hat{w}, \hat{b}) = \arg\max_{w,b} \min_n t_n \left(\frac{w^t x_n + b}{\|w\|} \right)$$

Consider xform $\begin{array}{l} w \rightarrow \alpha w \\ b \rightarrow \alpha b \end{array}$ for some $\alpha > 1$.

Note that classifier is unchanged since $w^t x + b$ unchanged.

However, xform scales $w^t x + b$ by factor of α.

For any w, b that correctly classifies, choose α to give $\min_n t_n (w^t x_n + b) = 1$. Then

$$(\hat{w}, \hat{b}) = \arg\max_{w,b} \min_n \frac{t_n (w^t x_n + b)}{\|w\|} \quad \text{s.t.} \quad \min_n t_n (w^t x_n + b) = 1$$

$$= \arg\max_{w,b} \frac{1}{\|w\|} \quad \text{s.t.} \quad \min_n t_n (w^t x_n + b) = 1$$

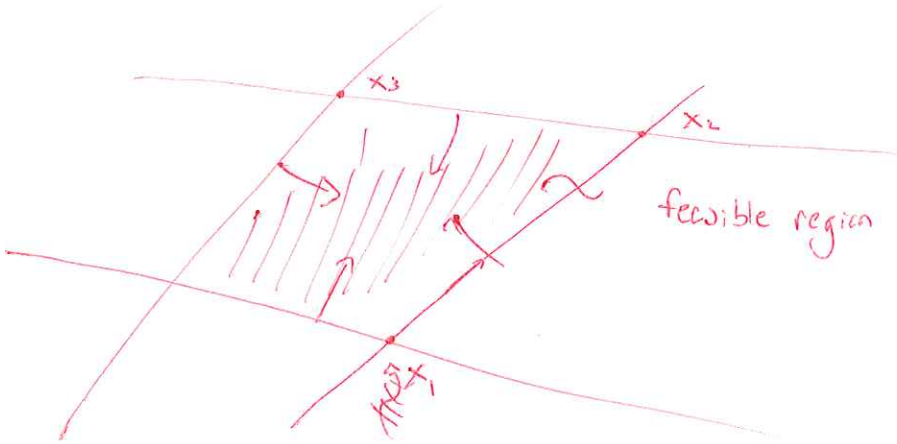
$$= \arg\max_{w,b} \frac{1}{\|w\|} \quad \text{s.t.} \quad \min_n t_n (w^t x_n + b) \geq 1$$

$$= \arg\min_{w,b} \|w\|^2 = w^t w \quad \text{s.t.} \quad \min_n t_n (w^t x_n + b) \geq 1$$

This is a quadratic programming (QP) problem.

Quadratic Programming

Linear programming seeks to maximize linear fn subject to linear constraints.



Traverse feasible corners untill adjacent corners give worse value of objective fn.

(Simplex method)

Quadratic Programming minimizes quadratic fn s.t. linear constraints.

Q.P.

Minimizes $\quad x^t D x + d^t x \quad$ s.t.

$$\begin{aligned} a_1^t x &\geq c_1 \\ a_2^t x &\geq c_2 \\ \vdots \quad & \quad \vdots \\ a_N^t x &\geq c_N \end{aligned} \quad \Longleftrightarrow \quad \underbrace{A}_{\begin{pmatrix} \text{--}\ a_1^t\ \text{--} \\ \text{--}\ a_2^t\ \text{--} \\ \vdots \\ \text{--}\ a_N^t\ \text{--} \end{pmatrix}} x \geq \underbrace{c}_{\begin{pmatrix} c_1 \\ \vdots \\ c_N \end{pmatrix}}$$

Our Problem

Minimize $\|w\|^2 = w^t w \quad$ s.t. $\quad t_n(w^t x_n + b) \geq 1 \quad n = 1 \cdots N$

$$(w \mid b) \underbrace{\left(\begin{array}{c|c} I & 0 \\ \hline 0 & 0 \end{array}\right)}_{D} \begin{pmatrix} w \\ \hline b \end{pmatrix}$$

$$\underbrace{\left(\begin{array}{c|c} t_1 x_1 \cdots & t_1 \\ t_2 x_2 \cdots & t_2 \\ \vdots & \\ t_N x_N \cdots & t_N \end{array}\right)}_{A} \begin{pmatrix} w \\ \vdots \\ \hline b \end{pmatrix} \geq \underbrace{\begin{pmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{pmatrix}}_{c}$$