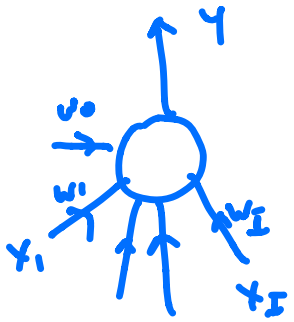
W08- Neural Networks

feed forward / neuron style networks



- inputs $x_1 \dots x_I$ $x_0 = 1$
- weights $w_1 \dots w_I$ w_0

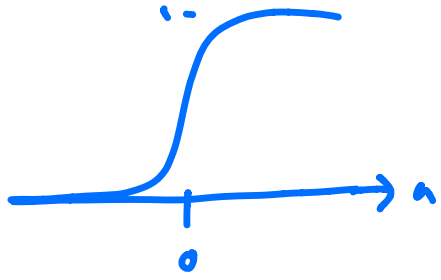
activation

$$a = w_0 + \sum_{i=1}^{I} w_i x_i$$

$$= \sum_{i=0}^{I} w_i x_i$$

- activity

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



Learning: calculate weights that minimize the error function.

Data $\{\vec{x}^{(1)}, \dots \vec{x}^{(N)}\}$

$\{t^{(1)}, \dots, t^{(N)}\}$ $\quad t^i = \begin{cases} 0 \to \text{oranges} \\ 1 \to \text{Apples} \end{cases}$

$$G(W) = -\sum_{n=1}^{N} \left[t^{(n)} \log y^{(n)} + (1-t^{(n)}) \log (1-y^{(n)}) \right]$$

Learn: minimize error function $G(w)$

Gradient descent learning.

$$\frac{\delta G(w)}{\delta w_i} = -\sum_{n=1}^{N} (t^{(n)} - y^{(n)}) x_i^{(n)}$$

$$\bar{g} = \left(\frac{\delta G}{\delta w_0} | \cdots | \frac{\delta G}{\delta w_I} \right) = -\sum_{n=1}^{N} (t^{(n)} - y^{(n)}) \bar{x}^{(n)}$$

start $\bar{w}^{old}$

(1) update $\bar{w}^{(new)} = \bar{w}^{old} - \eta \bar{g}$

$$= \bar{w}^{old} + \eta \sum_{n=1}^{N} (t^{(n)} - y_{old}^{(n)}) \bar{x}^{(n)}$$

batch update

(2) update $\bar{w}^{(new)} = \bar{w}^{old} + \eta (t^{(k)} - y_{old}^{(k)}) \bar{x}^{(k)}$

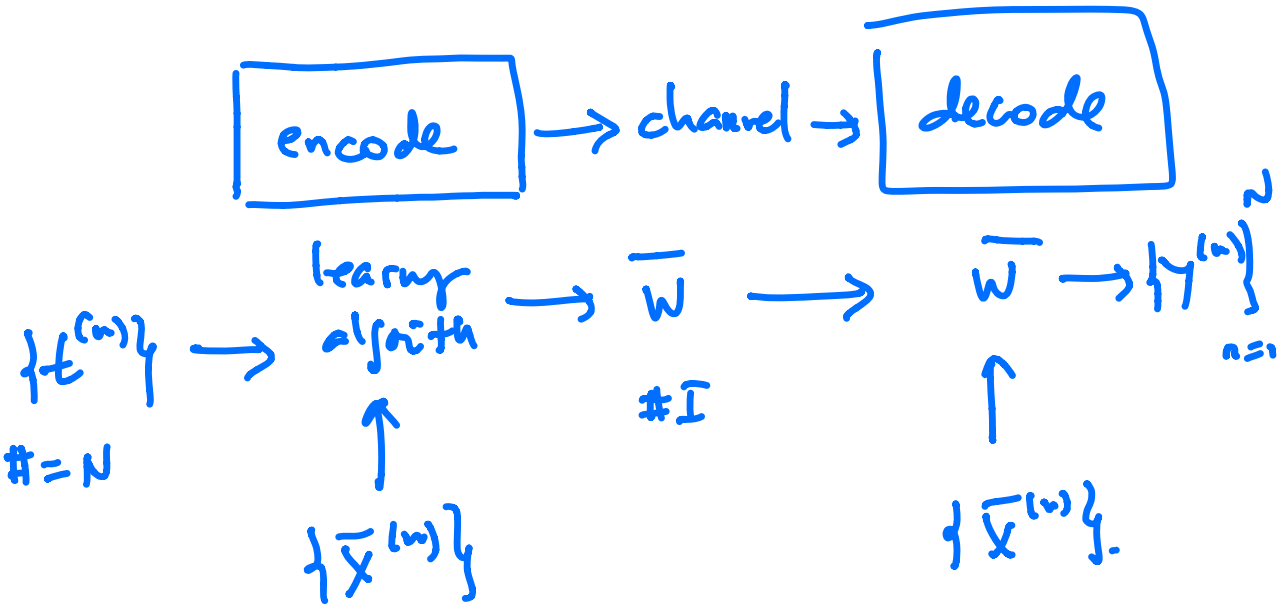
for $k \in (1, \ldots, N)$ picked at random.

Regularization

$$G(w) + \alpha \frac{1}{2} \sum_i w_i^2.$$

$$\bar{w}^{new} = (1 - \alpha\eta)\, \bar{w}^{old} + \eta \sum_n (t^n - y^{(w)}_{old})\, \bar{x}^{(n)}$$

Learning as communication channel



Reproduce information in $\{t^{(n)}\}_{n=1}^{N}$ using only $\{w_i\}_{i=0}^{I} \rightarrow I \ll N$

optimal communication channel

Learning as Inference

Optimize

$$M(w) = G(w, data) + \alpha R(w)$$

Remember

$$P(\bar{w}|D) = P(D|\bar{w})\,P(\bar{w})$$

compare to

$$e^{-M(w)} = e^{-G(w,D)}\; e^{-\alpha R(w)}$$

$$\updownarrow \qquad \updownarrow \qquad \updownarrow$$

$$P(\bar{w}|D) \qquad P(D|\bar{w}) \qquad P(\bar{w})$$

$$e^{-G(w)} = e^{\sum_n \left[t^{(n)} \log y^{(n)} + (1-t^{(n)}) \log (1-y^{(n)}) \right]}$$

$$= \prod_{n=1}^{N} e^{t^{(n)} \log y^{(n)}} \cdot e^{(1-t^{(n)}) \log (1-y^{(n)})}$$

$$= \prod_{n=1}^{N} \left(y^{(n)}\right)^{t^{(n)}} \left(1-y^{(n)}\right)^{1-t^{(n)}}$$

$$P(t^{(n)}=1 \mid \bar{w}) = y^{(n)}$$

$$P(t^{(n)}=0 \mid \bar{w}) = 1-y^{(n)}$$

$$P(t^{(n)} \mid w) = y^{(n)\,t^{(n)}} \left(1-y^{(n)}\right)^{1-t^{(n)}}$$

$$e^{-G(w)} = \prod_{n=1}^{N} P(t^{(n)} \mid w) = P(D \mid w)$$

$$P(w) = e^{-\frac{\alpha}{2} R(w)} = e^{-\frac{\alpha}{2} \sum_i w_i^2}$$

a gaussian prior!

$$R(w) = \frac{1}{2} \sum_i w_i^2 \rightarrow \text{Gaussian prior.}$$

Making further inference

Using this prob interpretation of learning we can estimate

$$P(t=1 \mid D) = \int\int\int_{w_0 w_1 w_2} P(\bar{w} \mid D)\, y(\bar{w}, \bar{x})\, dw_0\, dw_1\, dw_2$$

take a number of Mobile Carlo samples

$$P(t=1 \mid D) \simeq \frac{1}{S} \sum_{s=1}^{S} \frac{1}{1 + e^{-\bar{x}\, \bar{w}^{(s)}}}$$

Use Metropolis-Hastings MC sampler.

+ modify existing weights

$$w_i' = w_i + \eta_i$$

+ calculate $M(w') = S(w') + \alpha R(w')$

$$P(w) = e^{-M(w)}$$

$$P(w') = e^{-M(w')}$$

$$\frac{P(w')}{P(w)} \begin{cases} > 1 \rightarrow \text{take sample } w' \\ < 1 \rightarrow \text{draw } r \rightarrow \text{take sample } w \end{cases}$$

Collect your samples

calculate $\tilde{y} = \frac{1}{S} \sum_s y(w^s, \bar{x})$

→ class code

Feedback networks

Content addressable memories

i) memory stays

ii) can add one memory at the time w/o losing previous memories

iii) noisy versions are identified as memories

iv) robust to memory impediments

Hopfield Network

Hopefield network

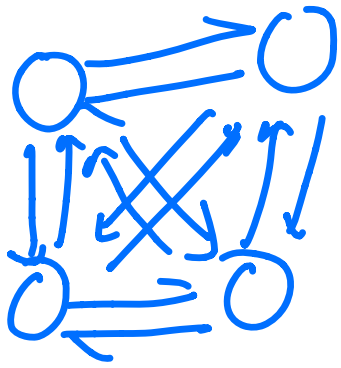
N neurons

- **Architecture**

  all neurons fully connected w/ bidirectional weights

  $w_{nm} = w_{mn}, m \neq n$

  $w_{nn} = 0$



- **Activation**

  each neurons output y_n is input to all others

  $a_n = \sum_{m \neq n} w_{mn} y_m$

- Activity

$$\gamma(a) = \begin{cases} +1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

- learning rule

Hebb's rule (1949)

if you want to learn K memories

$$w_{nm} = \sum_{k=1}^{K} y_n^{(k)} y_m^{(k)}$$

the memory challenge

Can a network trained on "A"
require imperfect "A"s?

→ class code