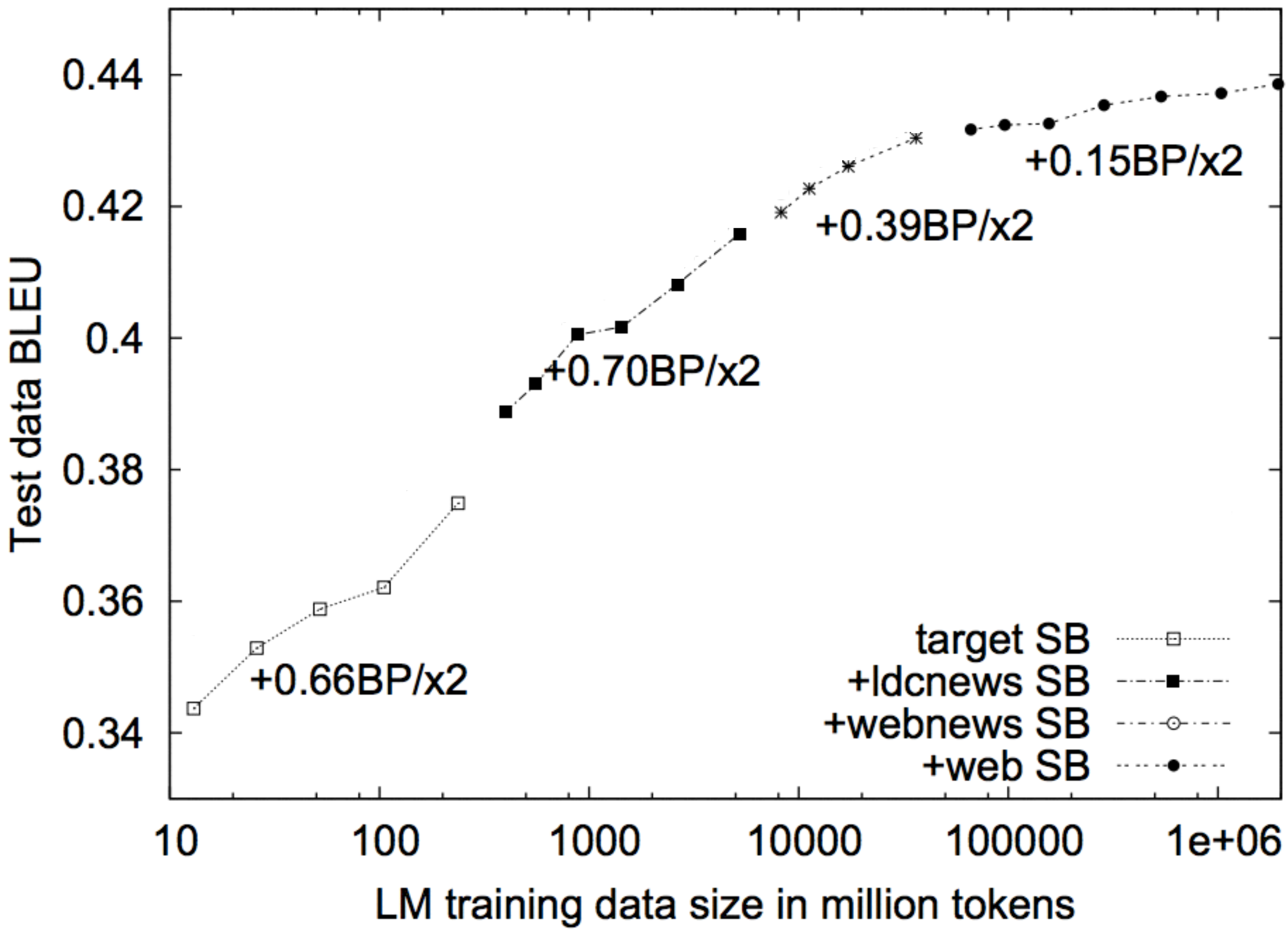
“Neural” language models



Test data BLEU
0.44
0.42
0.4
0.38
0.36
0.34
10
100
1000
10000
100000
1e+06
LM training data size in million tokens
+0.66BP/x2
+0.70BP/x2
+0.39BP/x2
+0.15BP/x2
target SB
+ldcnews SB
+webnews SB
+web SB

LMs in MT

$$p(\mathbf{e}, \mathbf{f}) = p_{LM}(\mathbf{e}) \times p_{TM}(\mathbf{f} \mid \mathbf{e})$$

What is the probability of a sentence?

- Requirements
  - Assign a probability to *every sentence* (i.e., string of words)

$$\sum_{\mathbf{e} \in \Sigma^*} p_{\mathrm{LM}}(\mathbf{e}) = 1$$

$$p_{\mathrm{LM}}(\mathbf{e}) \geq 0 \quad \forall \mathbf{e} \in \Sigma^*$$

n-gram LMs

$$p_{\mathrm{LM}}(\mathbf{e})$$

Vector-valued random variable

Whence parameters?

Whence parameters?

Estimation.

$$
\begin{aligned}
p(x \mid y) &= \frac{p(x, y)}{p(y)} \\
\hat{p}_{\mathrm{MLE}}(x) &= \frac{\mathrm{count}(x)}{N} \\
\hat{p}_{\mathrm{MLE}}(x, y) &= \frac{\mathrm{count}(x, y)}{N} \\
\hat{p}_{\mathrm{MLE}}(x \mid y) &= \frac{\mathrm{count}(x, y)}{N} \times \frac{N}{\mathrm{count}(y)} \\
&= \frac{\mathrm{count}(x, y)}{\mathrm{count}(y)}
\end{aligned}
$$

$$p(x \mid y) = \frac{p(x, y)}{p(y)}$$

$$\hat{p}_{\mathrm{MLE}}(x) = \frac{\mathrm{count}(x)}{N}$$

$$\hat{p}_{\mathrm{MLE}}(x, y) = \frac{\mathrm{count}(x, y)}{N}$$

$$\hat{p}_{\mathrm{MLE}}(x \mid y) = \frac{\mathrm{count}(x, y)}{N} \times \frac{N}{\mathrm{count}(y)}$$

$$\boxed{= \frac{\mathrm{count}(x, y)}{\mathrm{count}(y)}}$$

$$
\begin{aligned}
p(x \mid y) &= \frac{p(x, y)}{p(y)} \\
\hat{p}_{\mathrm{MLE}}(x) &= \frac{\mathrm{count}(x)}{N} \\
\hat{p}_{\mathrm{MLE}}(x, y) &= \frac{\mathrm{count}(x, y)}{N} \\
\hat{p}_{\mathrm{MLE}}(x \mid y) &= \frac{\mathrm{count}(x, y)}{N} \times \frac{N}{\mathrm{count}(y)} \\
&= \frac{\mathrm{count}(x, y)}{\mathrm{count}(y)}
\end{aligned}
$$

$$
\hat{p}_{\mathrm{MLE}}(\texttt{call} \mid \texttt{friends}) = \frac{\mathrm{count}(\texttt{friends call})}{\mathrm{count}(\texttt{friends})}
$$

LM Evaluation

- Extrinsic evaluation: build a new language model, use it for some task (MT, ASR, etc.)
- Intrinsic: measure how good we are at modeling language

We will use **perplexity** to evaluate models

Given: $\mathbf{w}, p_{\mathrm{LM}}$

$$\mathrm{PPL} = 2^{\frac{1}{|\mathbf{w}|} \log_2 p_{\mathrm{LM}}(\mathbf{w})}$$

$$0 \leq \mathrm{PPL} \leq \infty$$

Perplexity

- Generally fairly good correlations with BLEU for *n*-gram models
- Perplexity is a generalization of the notion of branching factor
  - How many choices do I have at each position?
- State-of-the-art English LMs have PPL of ~100 word choices per position
- A uniform LM has a perplexity of $|\Sigma|$
- Humans do much better
- ... and bad models can do even *worse* than uniform!

MLE & Perplexity

- What is the **lowest (best) perplexity possible** for your model class?
- Compute the MLE!
- Well, that's easy...

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE -3.65172

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE -3.65172

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE	-3.65172	-2.07101

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE	-3.65172	-2.07101

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE -3.65172 -2.07101 -3.32231

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE -3.65172 -2.07101 $-\infty$

START my friends call me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE -3.65172 -2.07101 -3.32231 -0.271271

START my friends dub me Alex STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE -3.65172 -2.07101 $-\infty$ -2.54562

START	my	friends	call	me	Alex	STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE	-3.65172	-2.07101	-3.32231	-0.271271	-4.961

START	my	friends	dub	me	Alex	STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE	-3.65172	-2.07101	$-\infty$	-2.54562	-4.961

START	my	friends	call	me	Alex	STOP
	$p(\texttt{my} \mid \texttt{START})\times$	$p(\texttt{friends} \mid \texttt{my})\times$	$p(\texttt{call} \mid \texttt{friends})\times$	$p(\texttt{me} \mid \texttt{call})\times$	$p(\texttt{Alex} \mid \texttt{me})\times$	$p(\texttt{STOP} \mid \texttt{Alex})$
MLE	-3.65172	-2.07101	-3.32231	-0.271271	-4.961	-1.96773

START	my	friends	dub	me	Alex	STOP
	$p(\texttt{my} \mid \texttt{START})\times$	$p(\texttt{friends} \mid \texttt{my})\times$	$p(\texttt{dub} \mid \texttt{friends})\times$	$p(\texttt{me} \mid \texttt{dub})\times$	$p(\texttt{Alex} \mid \texttt{me})\times$	$p(\texttt{STOP} \mid \texttt{Alex})$
MLE	-3.65172	-2.07101	$-\infty$	-2.54562	-4.961	-1.96773

START	my	friends	call	me	Alex	STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{call} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{call}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE	-3.65172	-2.07101	-3.32231	-0.271271	-4.961	-1.96773

START	my	friends	dub	me	Alex	STOP

$$p(\texttt{my} \mid \texttt{START}) \times p(\texttt{friends} \mid \texttt{my}) \times p(\texttt{dub} \mid \texttt{friends}) \times p(\texttt{me} \mid \texttt{dub}) \times p(\texttt{Alex} \mid \texttt{me}) \times p(\texttt{STOP} \mid \texttt{Alex})$$

MLE	-3.65172	-2.07101	$-\infty$	-2.54562	-4.961	-1.96773

MLE assigns probability zero to unseen events

Zeros

- Two kinds of zero probs:
  - **Sampling zeros**: zeros in the MLE due to impoverished observations
  - **Structural zeros**: zeros that should be there. *Do these really exist?*
- Just because you haven't seen something, doesn't mean it doesn't exist.
- In practice, we don't like probability zero, even if there is an argument that it is a structural zero.

Smoothing

Smoothing an refers to a family of *estimation techniques* that seek to model important general patterns in data while avoiding modeling noise or sampling artifacts. In particular, for language modeling, we seek

$$p(\mathbf{e}) > 0 \quad \forall \mathbf{e} \in \Sigma^*$$

We will assume that Σ is known and finite.

Add-α Smoothing

$$\mathbf{p} \sim \mathrm{Dirichlet}(\boldsymbol{\alpha})$$
$$x_i \sim \mathrm{Categorical}(\mathbf{p}) \quad \forall 1 \leq i \leq |\mathbf{x}|$$

Assuming this model, what is the most probable value of **p**, having observed training data **x**?

(bunch of calculus - read about it on Wikipedia)

$$p_x^* = \frac{\mathrm{count}(x) + \alpha_x - 1}{N + \sum_{x'} (\alpha_{x'} - 1)} \quad \forall \alpha_x > 1$$

Add-α Smoothing

- Simplest possible smoother
- Surprisingly effective in many models
- Does not work well for language models
- There are procedures for dealing with 0 < alpha < 1
- When might these be useful?

Interpolation

- "Mixture of MLEs"

$$\begin{aligned}\hat{p}(\texttt{dub} \mid \texttt{my friends}) =& \lambda_3 \hat{p}_{\mathrm{MLE}}(\texttt{dub} \mid \texttt{my friends}) \\ &+ \lambda_2 \hat{p}_{\mathrm{MLE}}(\texttt{dub} \mid \texttt{friends}) \\ &+ \lambda_1 \hat{p}_{\mathrm{MLE}}(\texttt{dub}) \\ &+ \lambda_0 \frac{1}{|\Sigma|}\end{aligned}$$

Where do the lambdas come from?

Discounting

Discounting adjusts the frequencies of observed events downward to reserve probability for the things that have not been observed.

Note $f(w_3 \mid w_1, w_2) > 0$ only when $\mathrm{count}(w_1, w_2, w_3) > 0$

We introduce a **discounted frequency**:

$$0 \leq f^*(w_3 \mid w_1, w_2) \leq f(w_3 \mid w_1, w_2)$$

The total discount is the zero-frequency probability:

$$\lambda(w_1, w_2) = 1 - \sum_{w'} f^*(w' \mid w_1, w_2)$$

Back-off

Recursive formulation of probability:

$$\hat{p}_{\mathrm{BO}}(w_3 \mid w_1, w_2) = \begin{cases} f^*(w_3 \mid w_1, w_2) & \text{if } f^*(w_3 \mid w_1, w_2) > 0 \\ \alpha_{w_1,w_2} \times \lambda(w_1, w_2) \times \hat{p}_{\mathrm{BO}}(w_3 \mid \cancel{w_1}, w_2) & \text{otherwise} \end{cases}$$

"Back-off weight"

Question 1: how do we discount?

Kneser-Ney Discounting

- State-of-the-art in language modeling for 15 years
- Two major intuitions
  - Some contexts have lots of new words
  - Some words appear in lots of contexts
- Procedure
  - Only register a lower-order count the *first time* it is seen in a backoff context
  - Example: bigram model
    - “San Francisco” is a common bigram
    - But, we only count the unigram “Francisco” the first time we see the bigram “San Francisco” - **we change its unigram probability**

Back-off

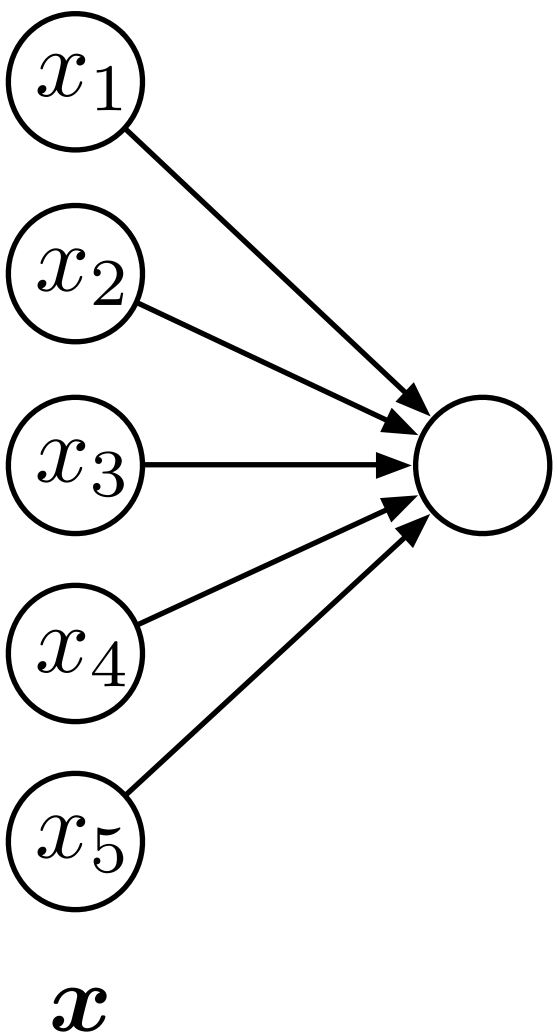
Recursive formulation of probability:

$$\hat{p}_{\mathrm{BO}}(w_3 \mid w_1, w_2) = \begin{cases} f^*(w_3 \mid w_1, w_2) & \text{if } f^*(w_3 \mid w_1, w_2) > 0 \\ \underbrace{\alpha_{w_1,w_2} \times \lambda(w_1, w_2)}_{\text{“Back-off weight”}} \times \hat{p}_{\mathrm{BO}}(w_3 \mid \cancel{w_1}, w_2) & \text{otherwise} \end{cases}$$

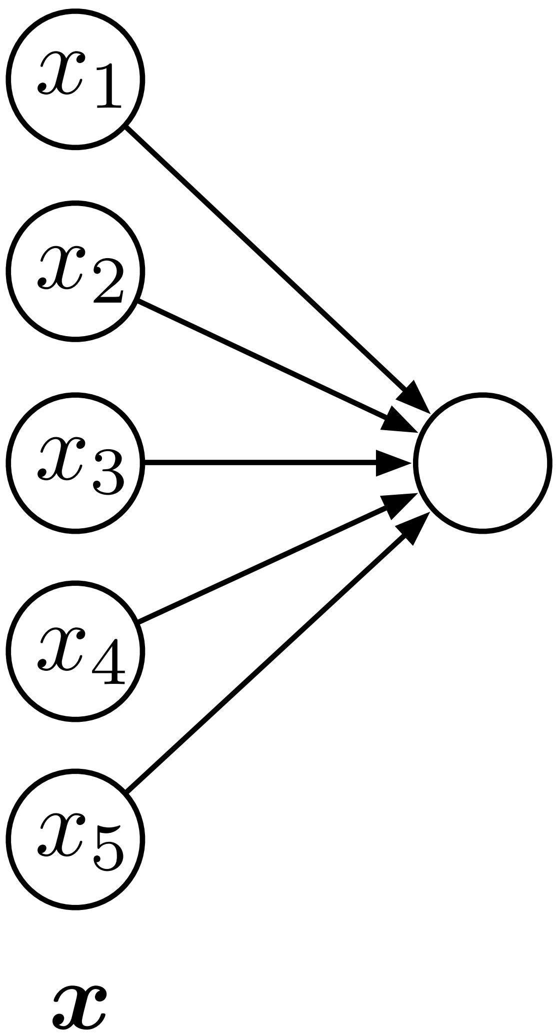
Question 1: how do we discount?

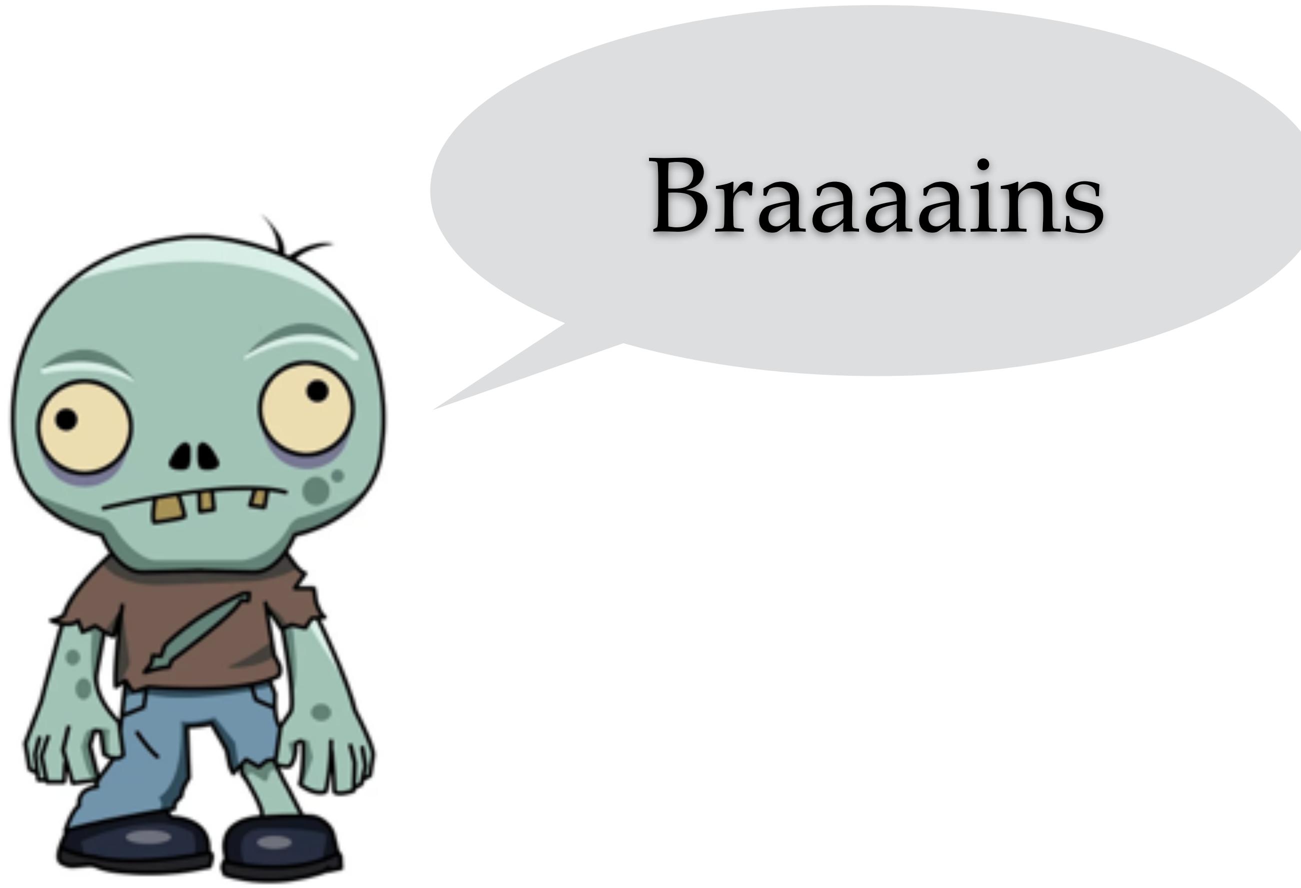
Question 2: how many parameters?

“Neurons”

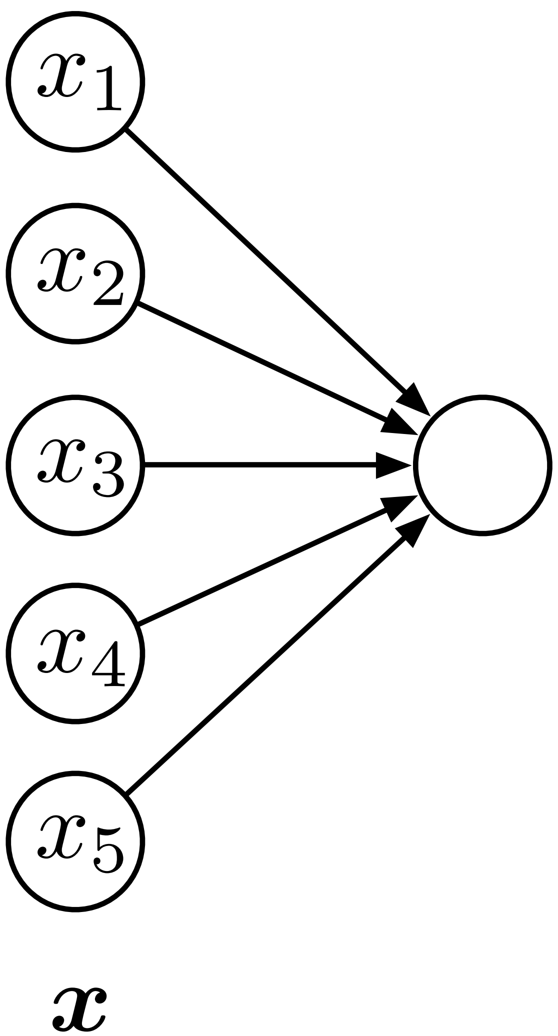


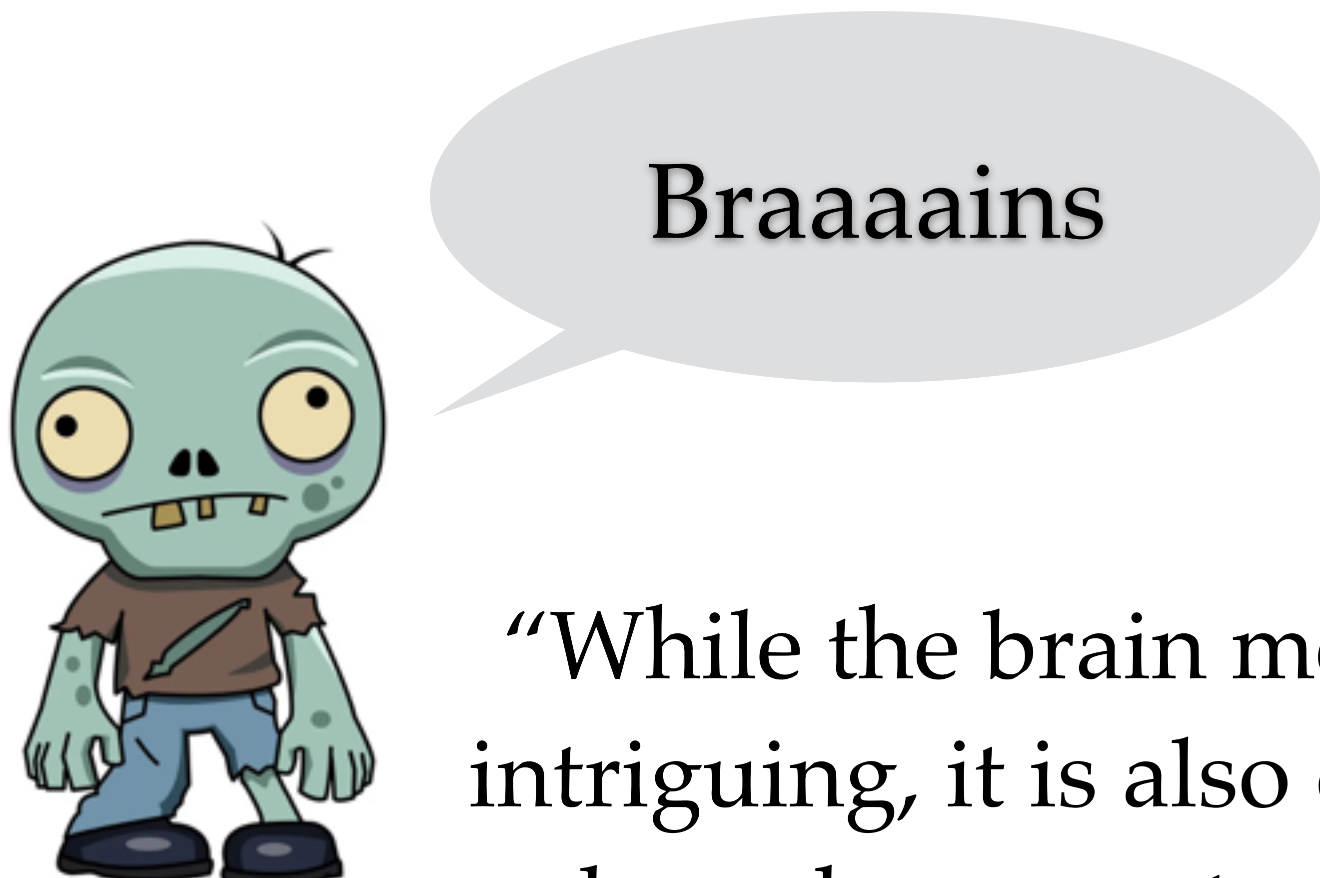
“Neurons”





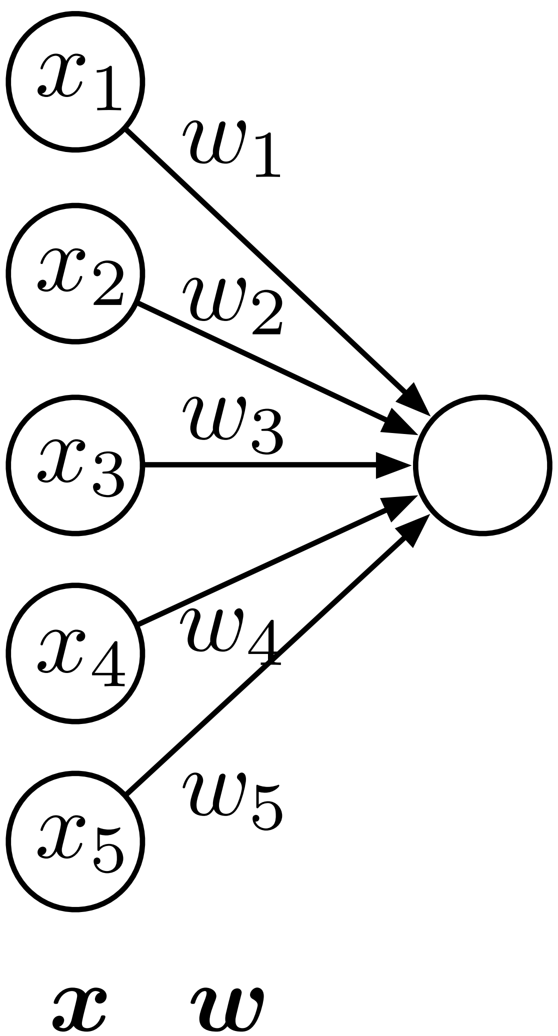
“Neurons”



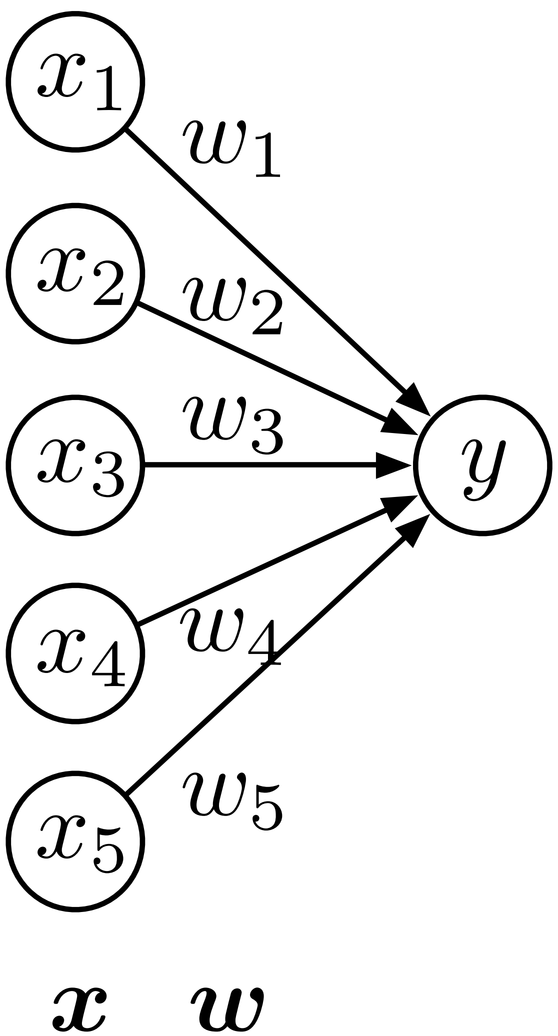


“While the brain metaphor is intriguing, it is also distracting and cumbersome to manipulate mathematically. We therefore switch to using more concise mathematical notation.” (Goldberg, 2015)

“Neurons”

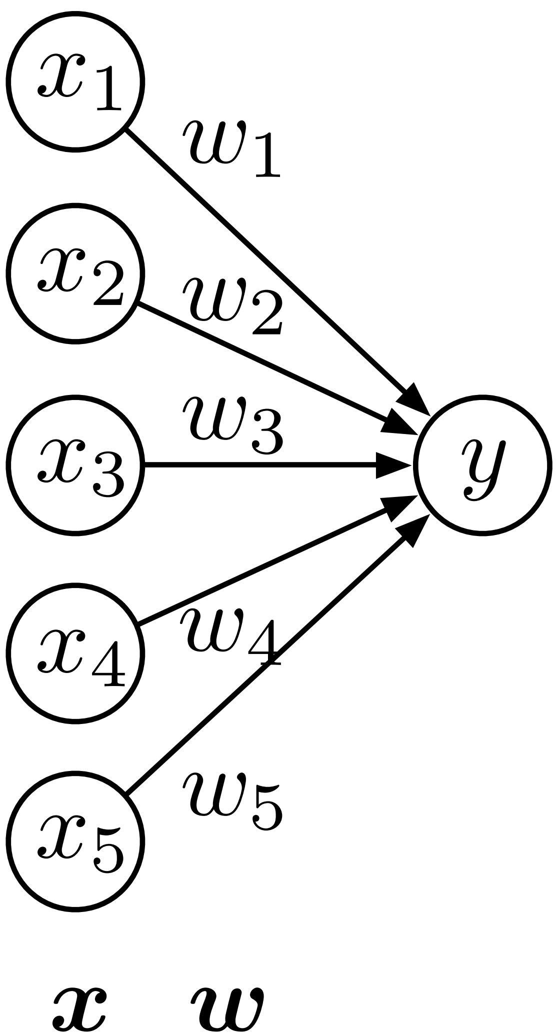


“Neurons”

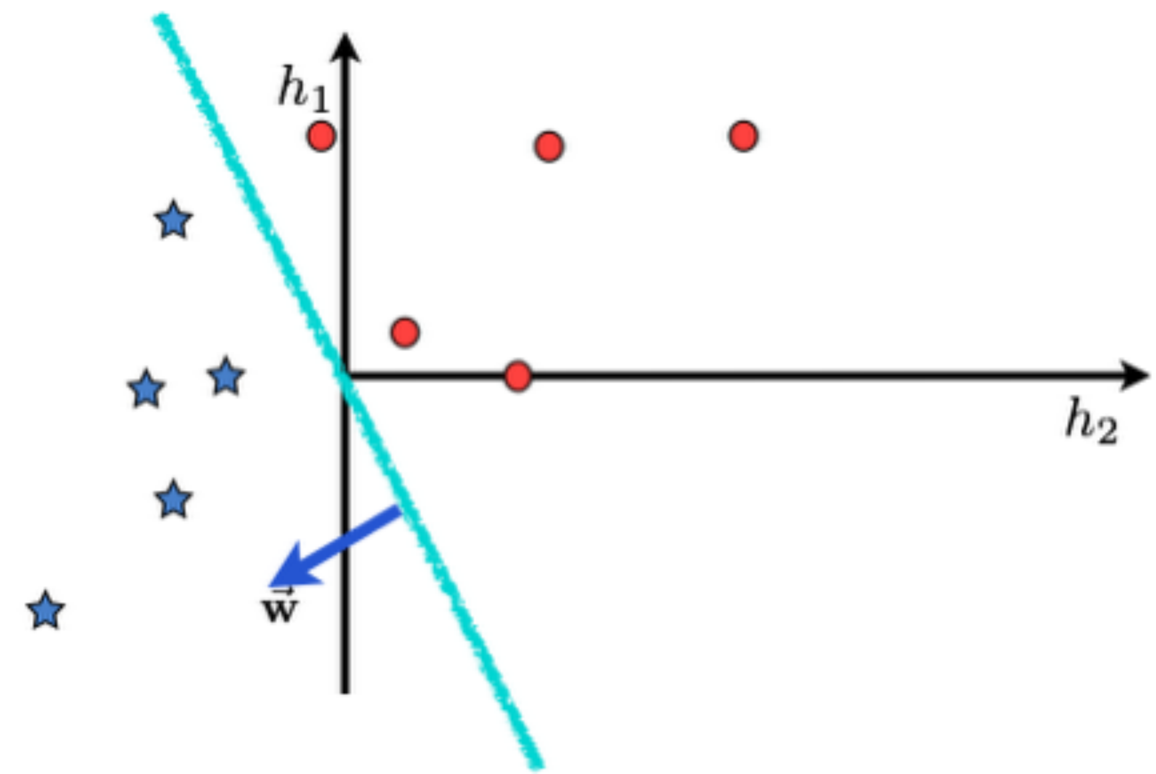


$$y = \boldsymbol{x} \cdot \boldsymbol{w}$$

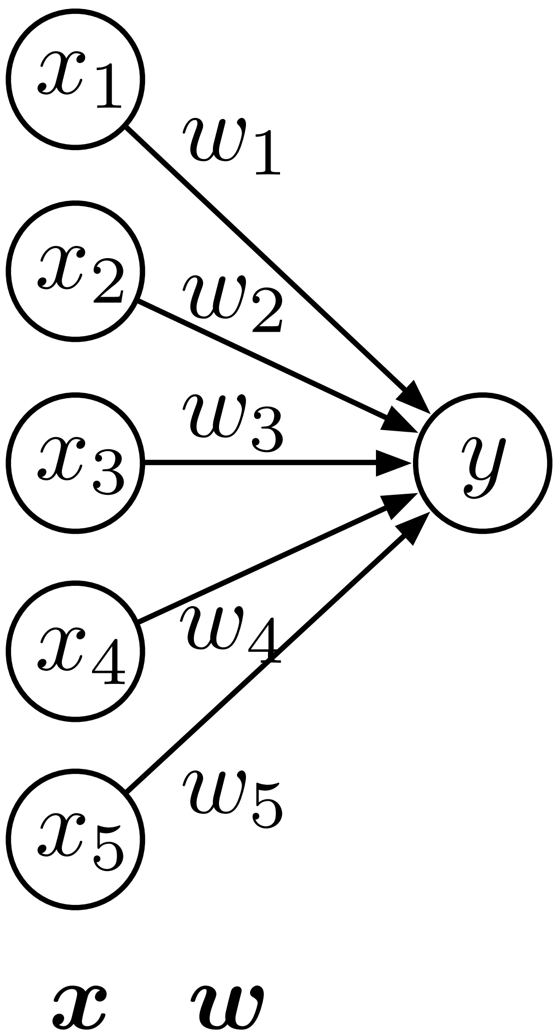
"Neurons"



$$y = \boldsymbol{x} \cdot \boldsymbol{w}$$

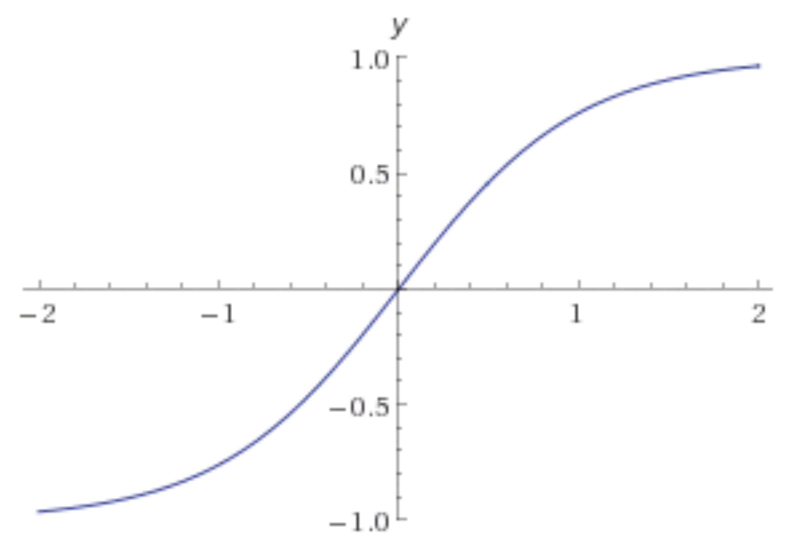


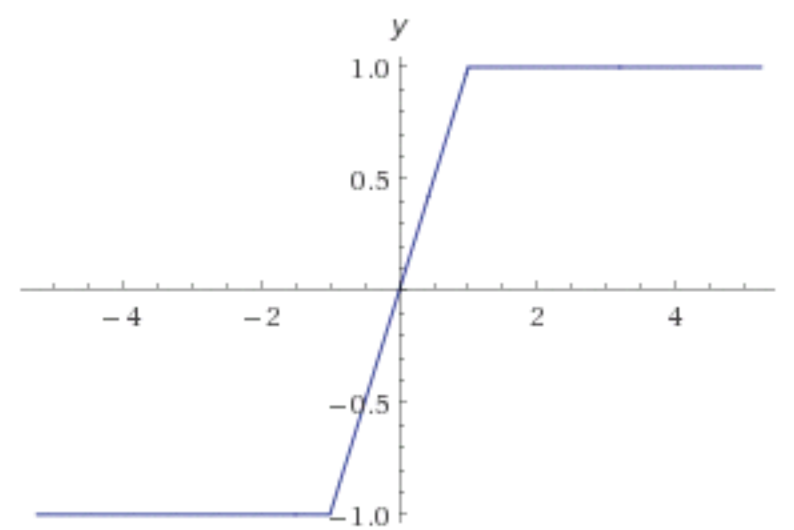
“Neurons”



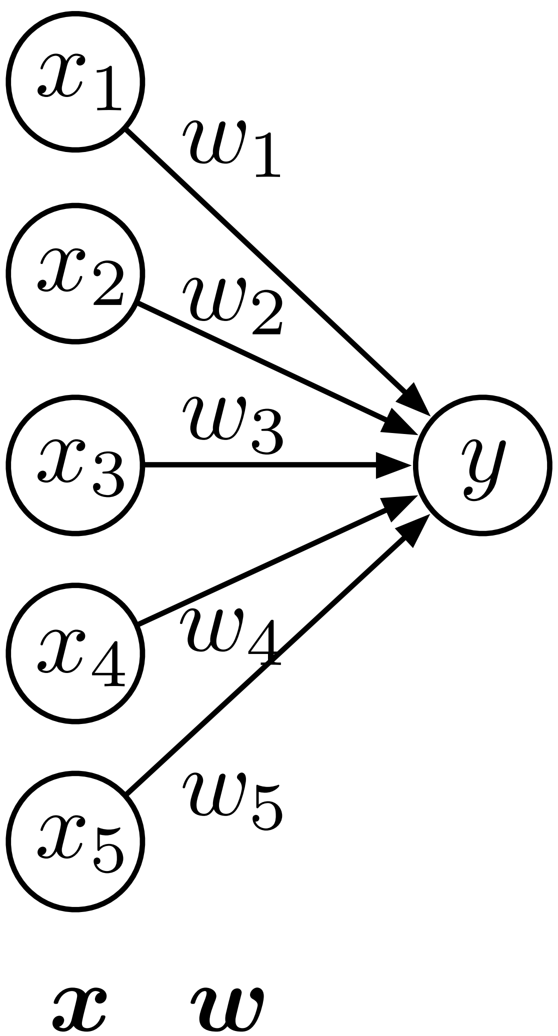
$$\cancel{y = \boldsymbol{x} \cdot \boldsymbol{w}}$$

$$y = g(\boldsymbol{x} \cdot \boldsymbol{w})$$

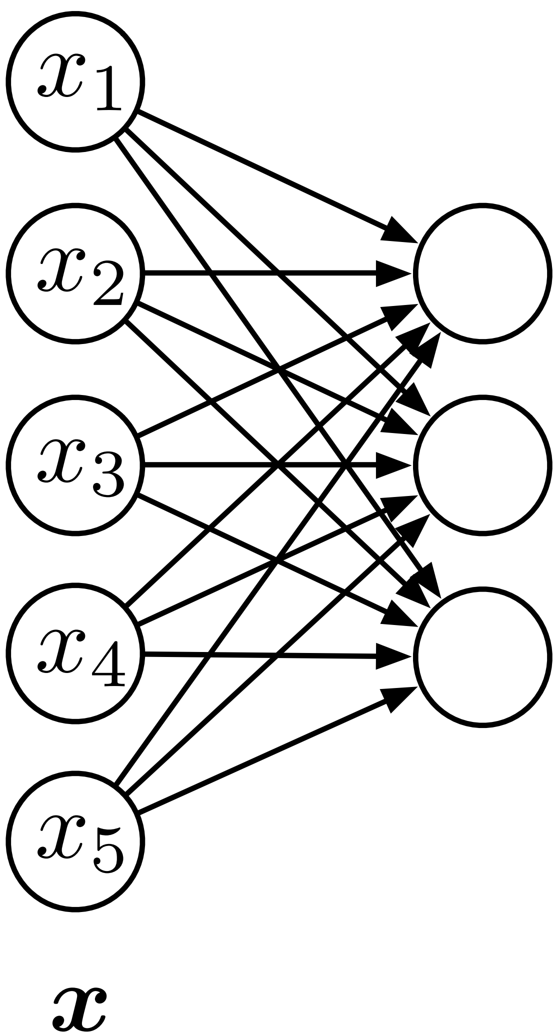




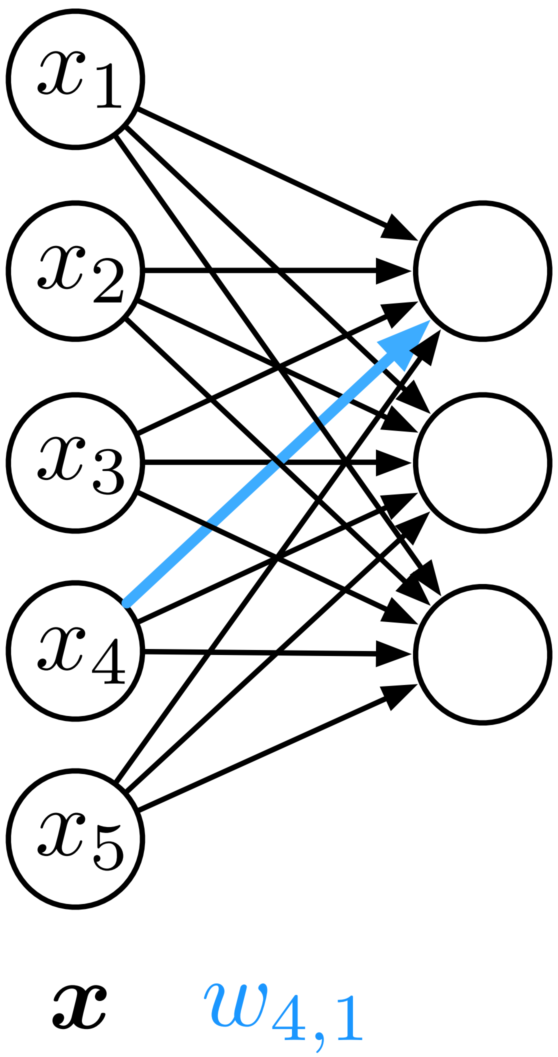
“Neurons”



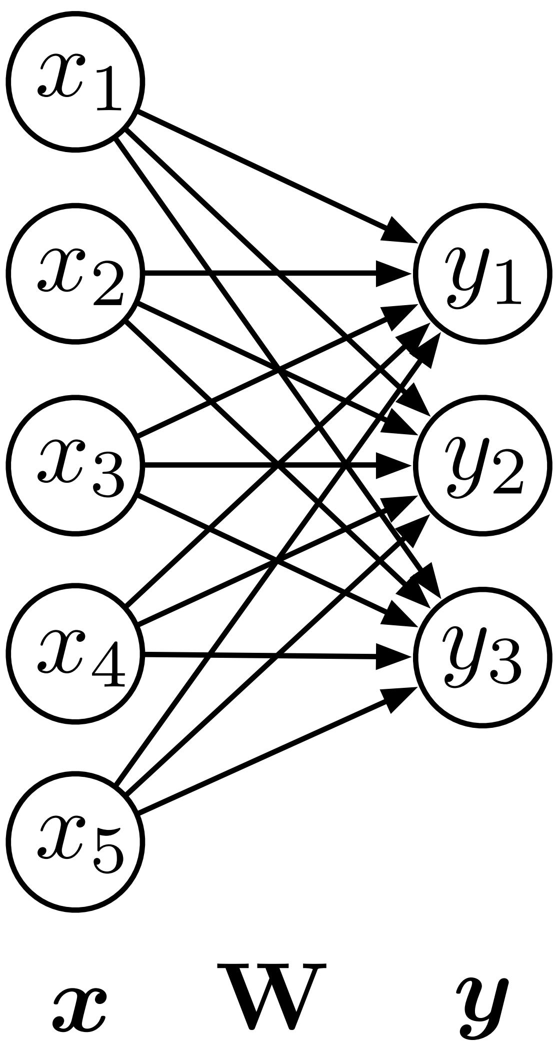
“Neural” Networks



“Neural” Networks

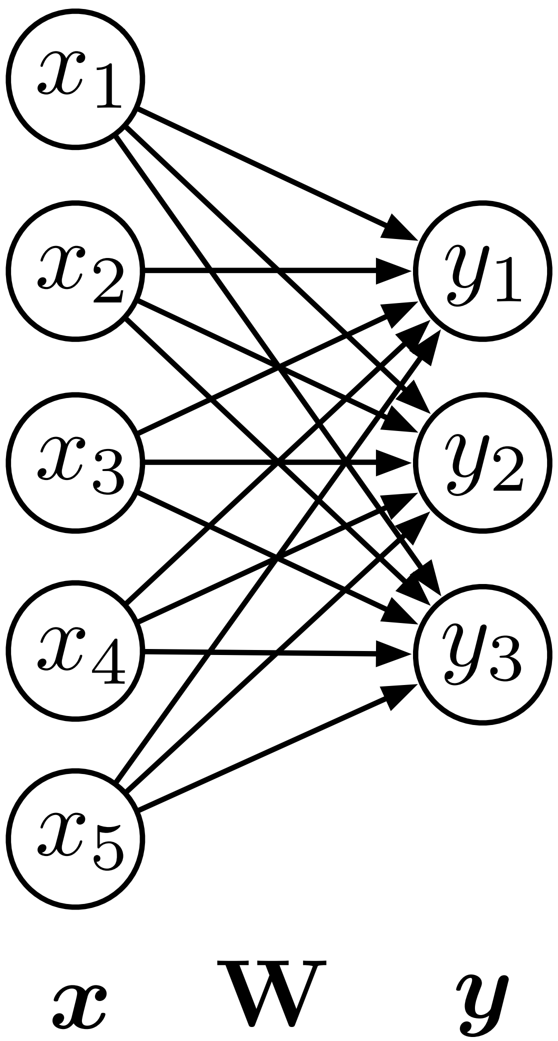


“Neural” Networks



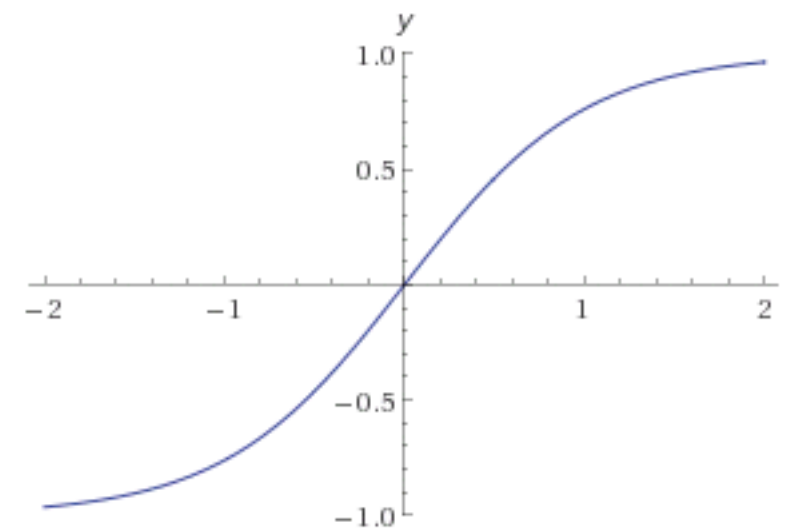
$$\boldsymbol{y} = \boldsymbol{x}^\top \mathbf{W}$$

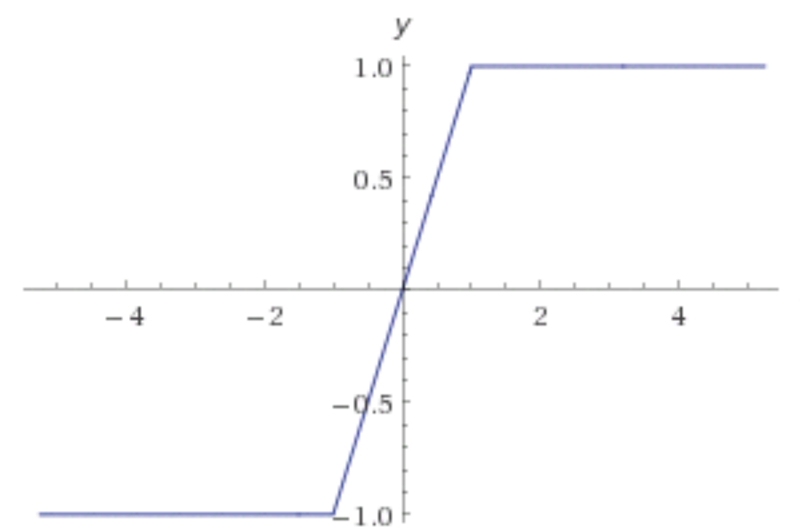
“Neural” Networks



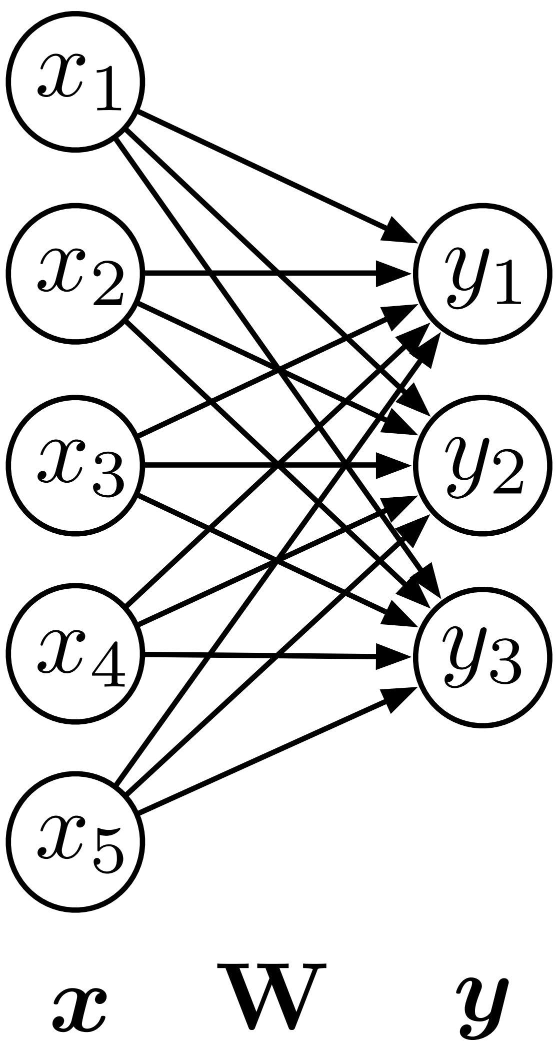
$$\cancel{\boldsymbol{y} = \boldsymbol{x}^\top \mathbf{W}}$$

$$\boldsymbol{y} = g(\boldsymbol{x}^\top \mathbf{W})$$



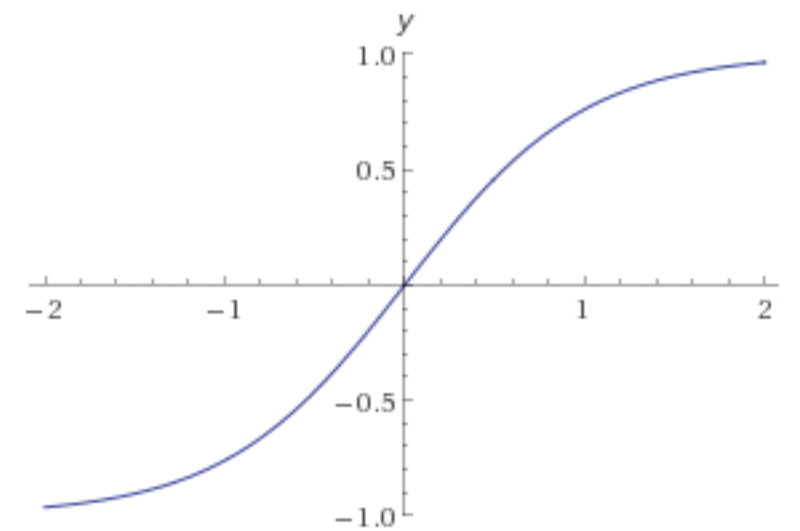


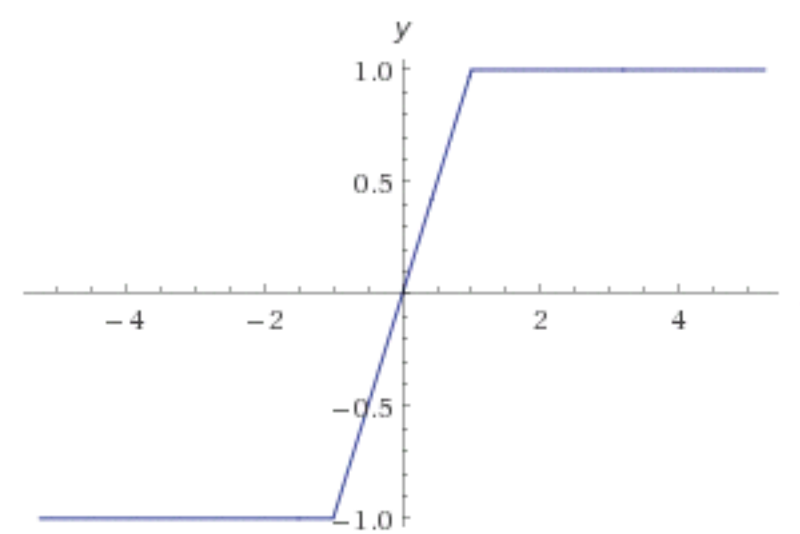
"Neural" Networks



$$\cancel{\boldsymbol{y} = \boldsymbol{x}^\top \mathbf{W}}$$

$$\boldsymbol{y} = g(\boldsymbol{x}^\top \mathbf{W})$$

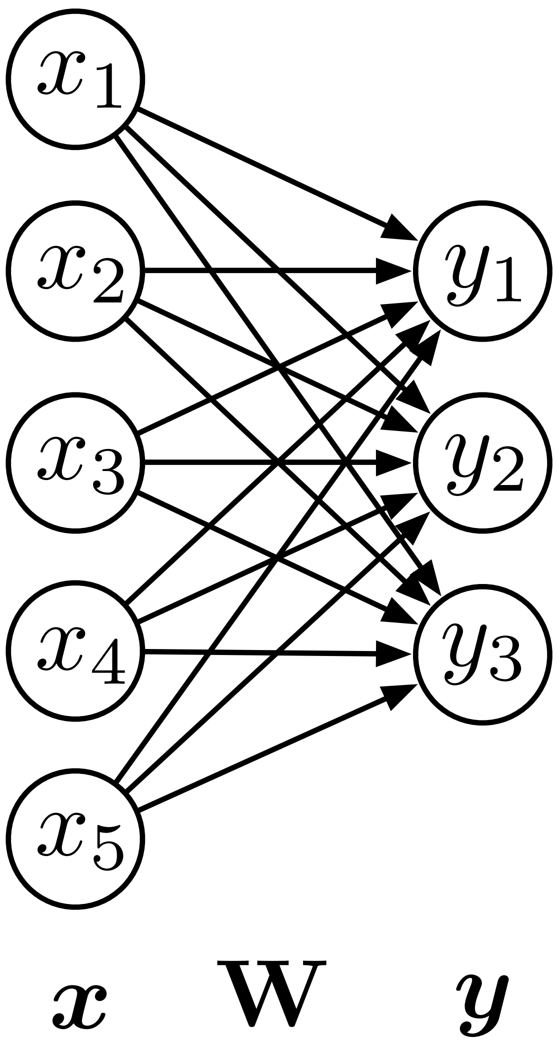




What about probabilities?

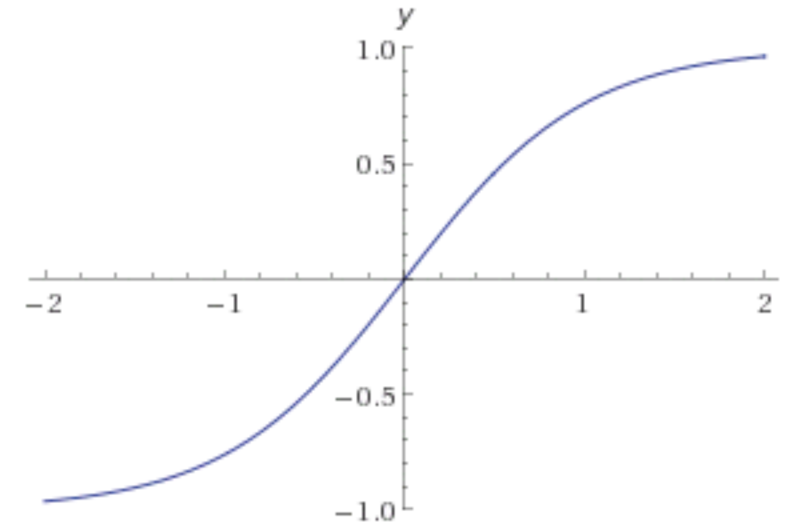
Let each y_i be an outcome.

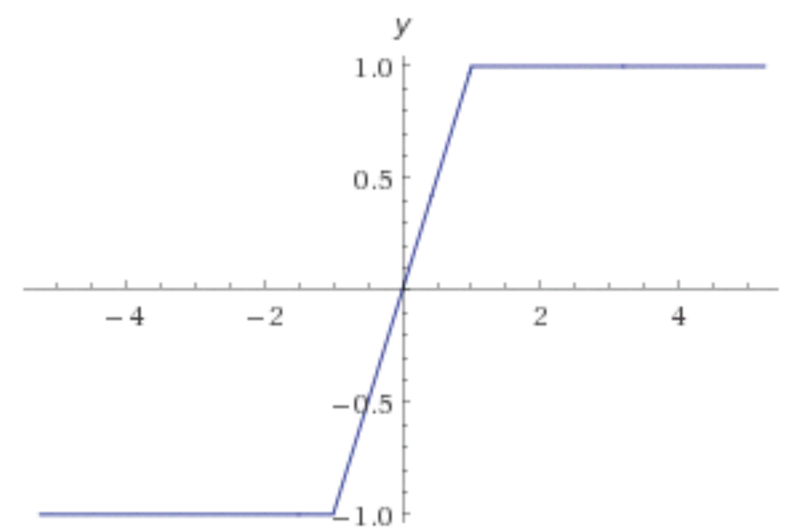
“Neural” Networks



$$\cancel{\boldsymbol{y} = \boldsymbol{x}^\top \mathbf{W}}$$

$$\boldsymbol{y} = g(\boldsymbol{x}^\top \mathbf{W})$$



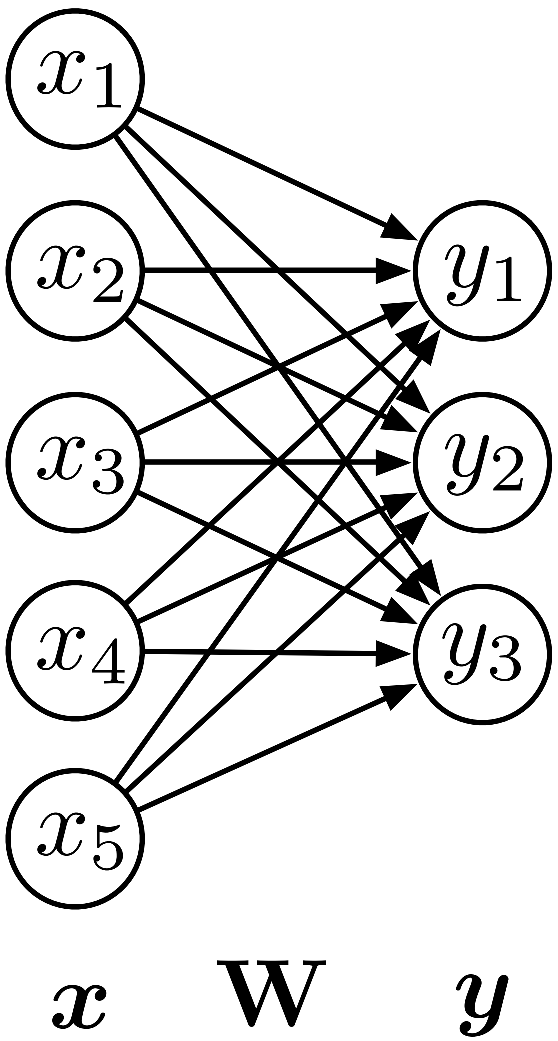


What about probabilities?
Let each y_i be an outcome.

“Soft max”

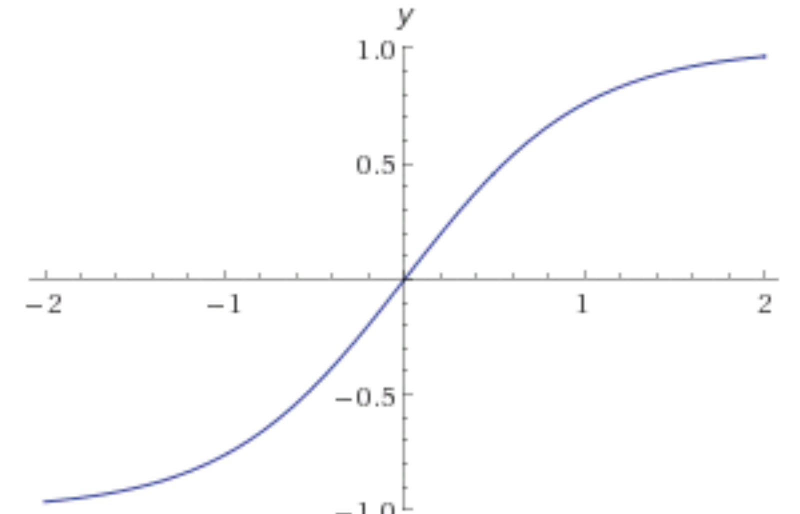
$$g(\boldsymbol{u})_i = \frac{\exp u_i}{\sum_{i'} \exp u_{i'}}$$

“Neural” Networks



$$\cancel{\boldsymbol{y} = \boldsymbol{x}^\top \mathbf{W}}$$

$$\boldsymbol{y} = g(\boldsymbol{x}^\top \mathbf{W})$$



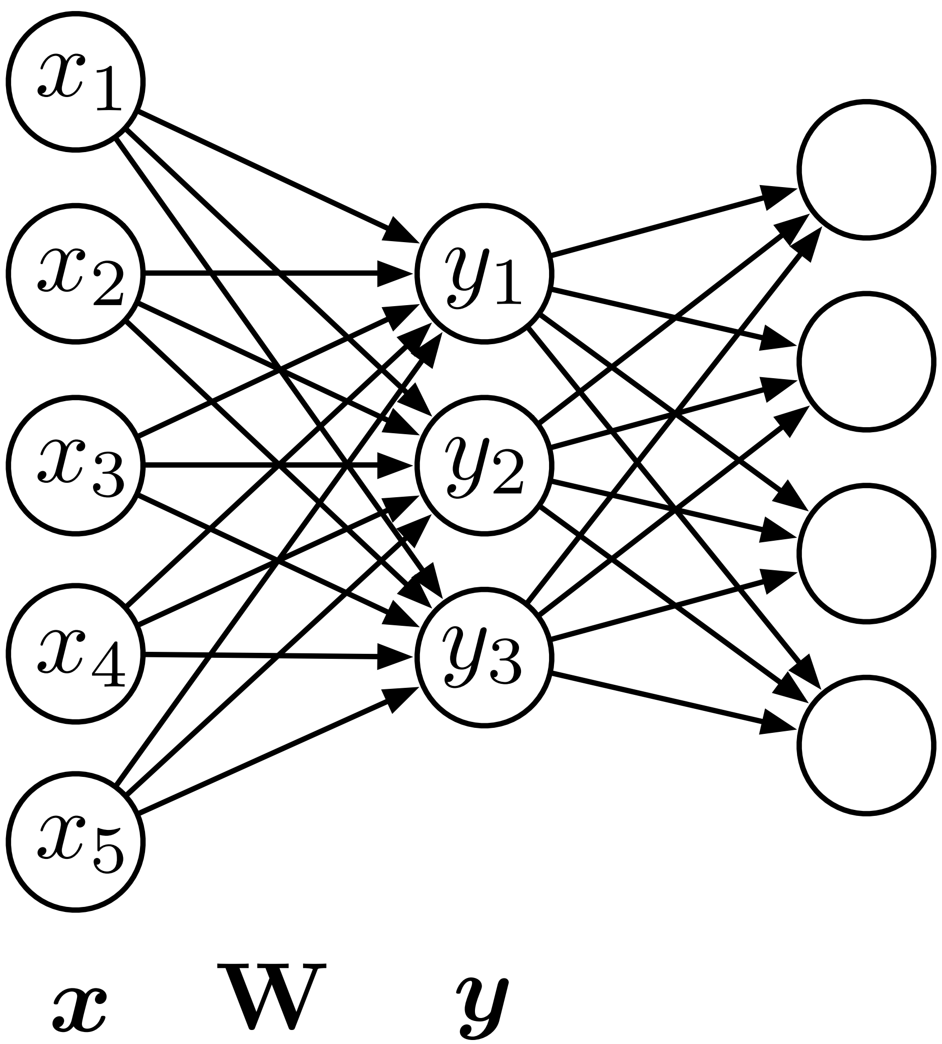
Look familiar?

What about probabilities?
Let each y_i be an outcome.

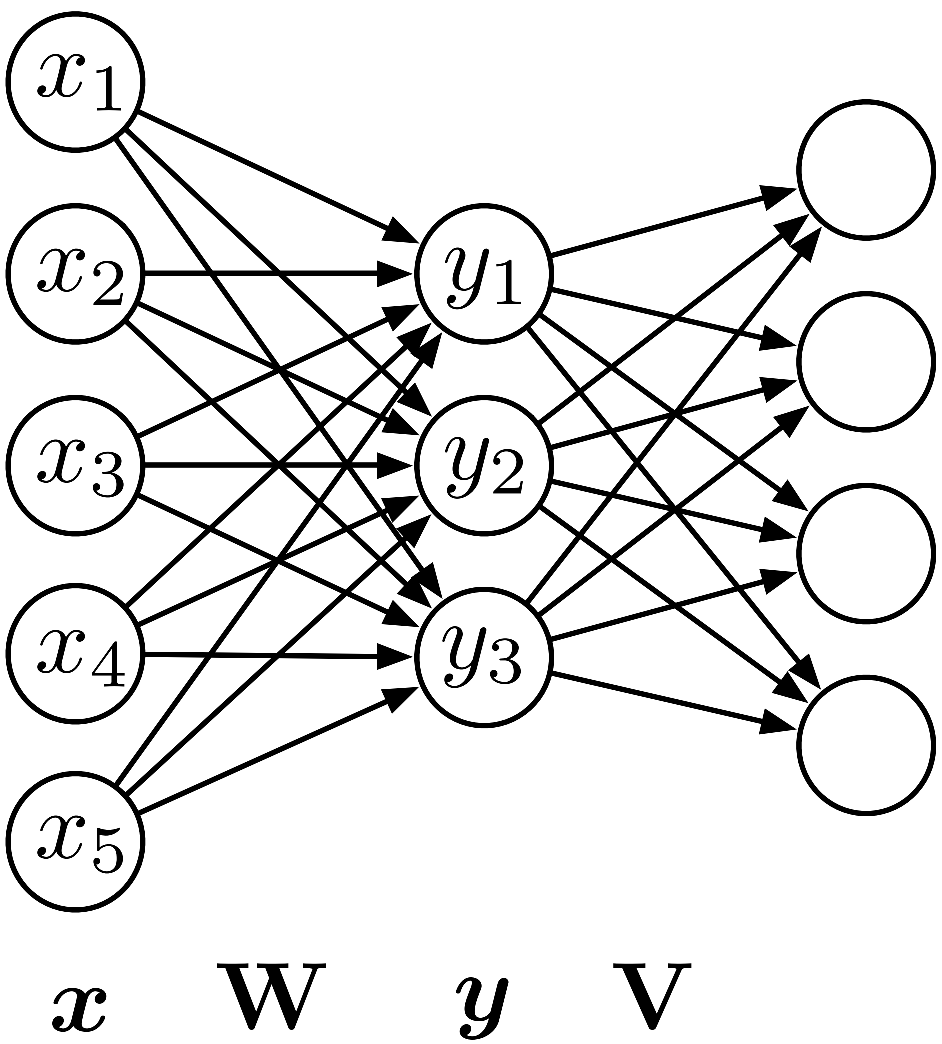
“Soft max”

$$g(\boldsymbol{u})_i = \frac{\exp u_i}{\sum_{i'} \exp u_{i'}}$$

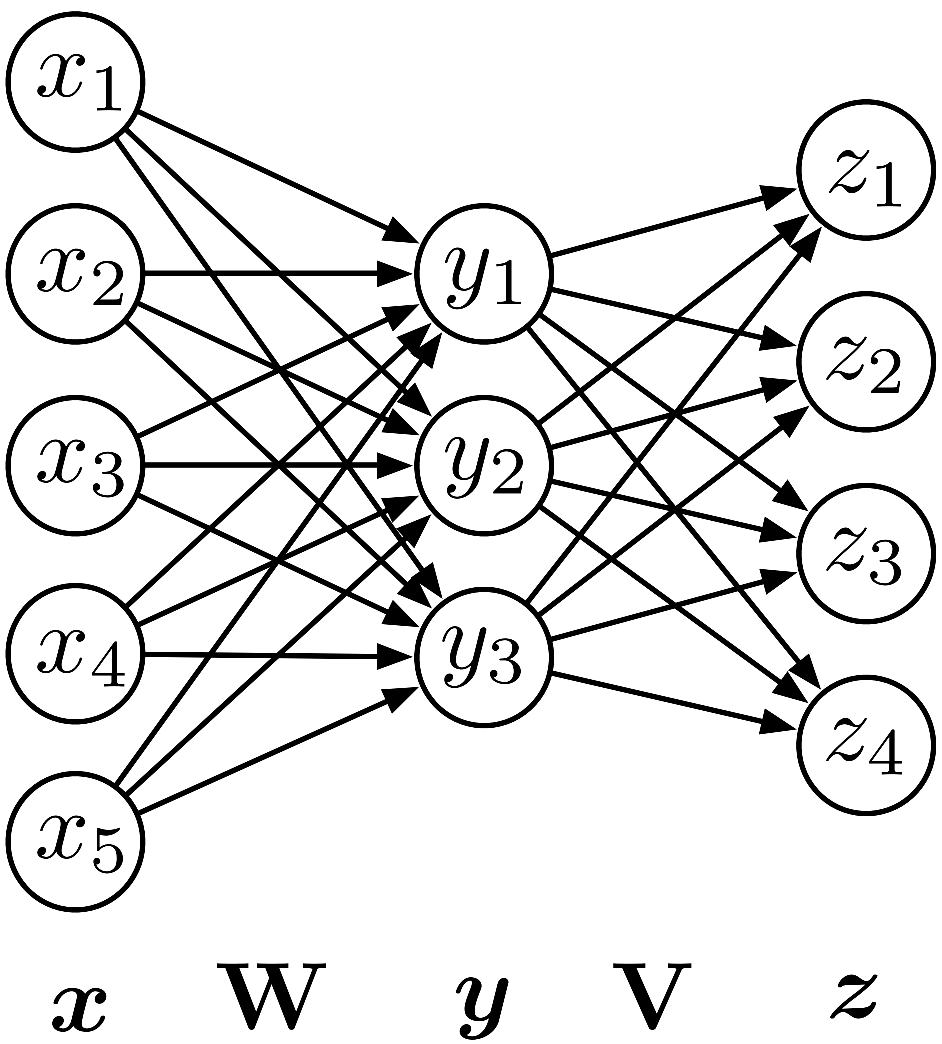
“Deep”



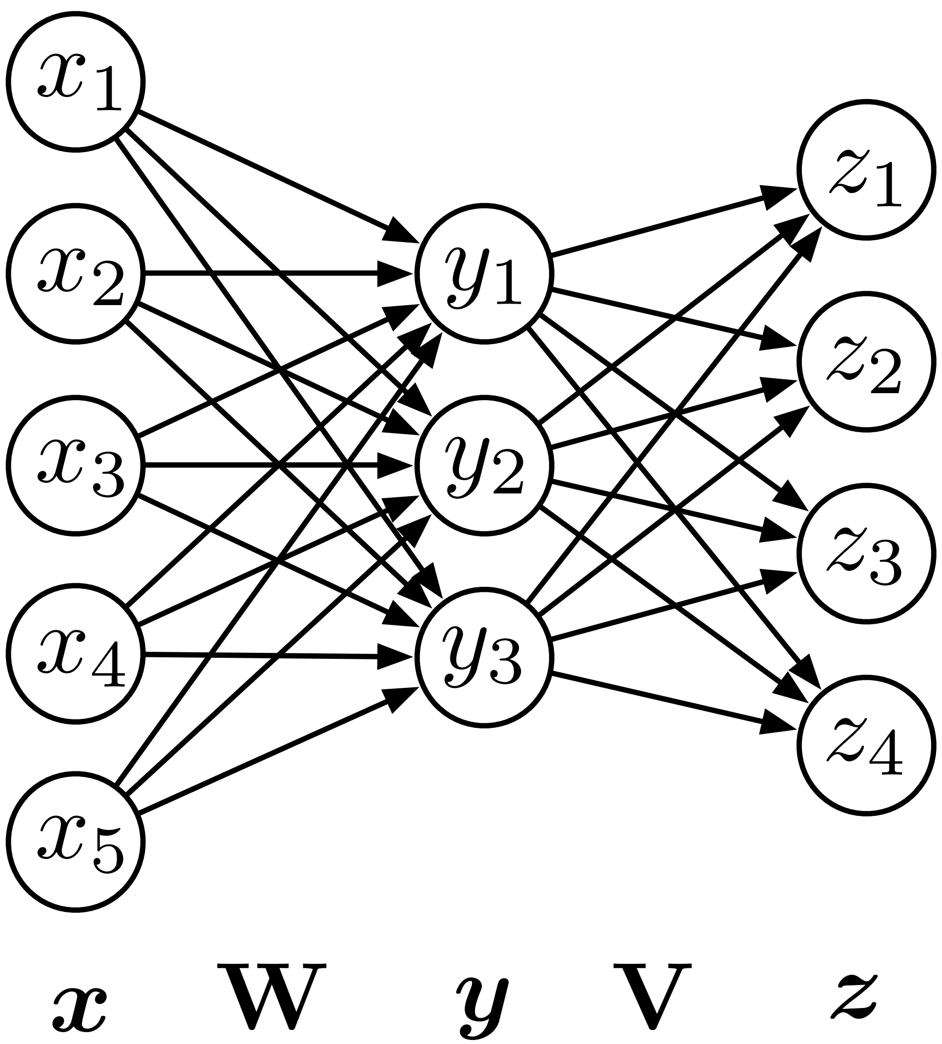
“Deep”


x_1
x_2
x_3
x_4
x_5
y_1
y_2
y_3
$\boldsymbol{x}$ $\mathbf{W}$ $\boldsymbol{y}$ $\mathbf{V}$

“Deep”

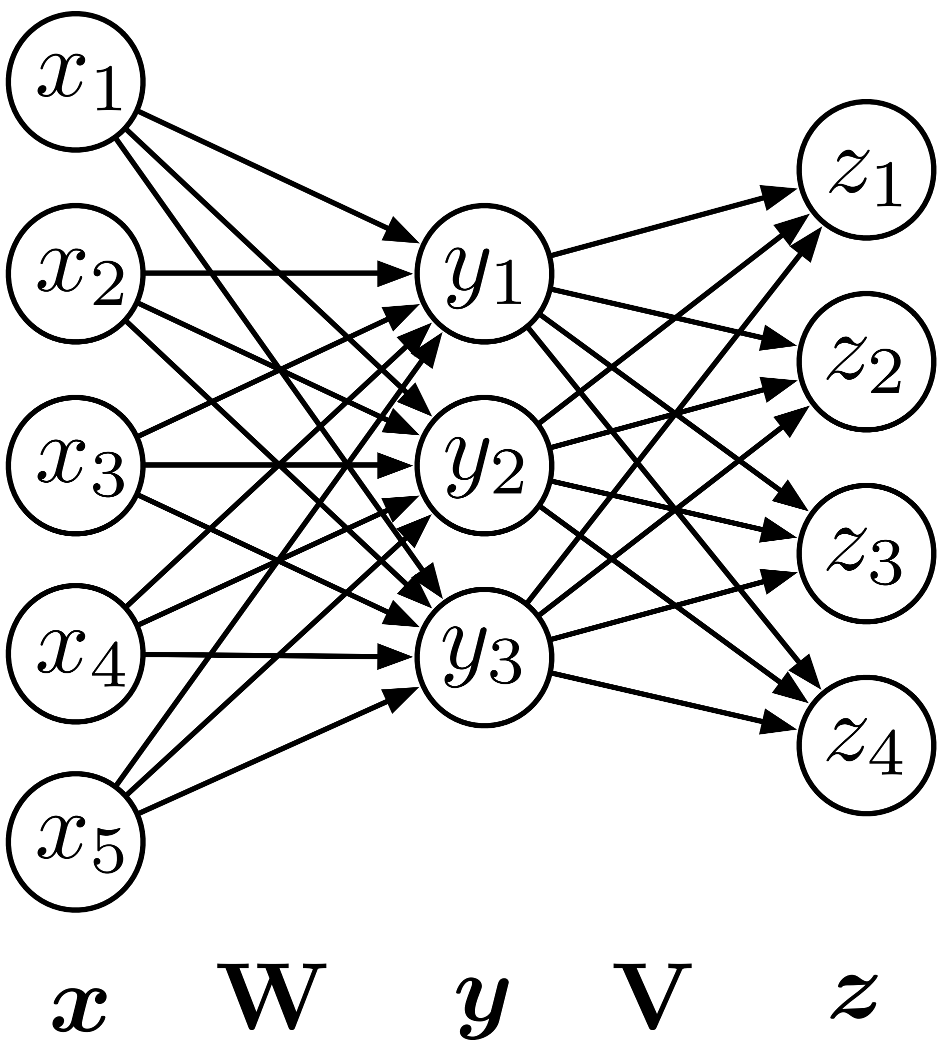

x_1
x_2
x_3
x_4
x_5
y_1
y_2
y_3
z_1
z_2
z_3
z_4
$\boldsymbol{x}$ $\mathbf{W}$ $\boldsymbol{y}$ $\mathbf{V}$ $\boldsymbol{z}$

"Deep"



$$\boldsymbol{z} = g(\boldsymbol{y}^\top \mathbf{V})$$

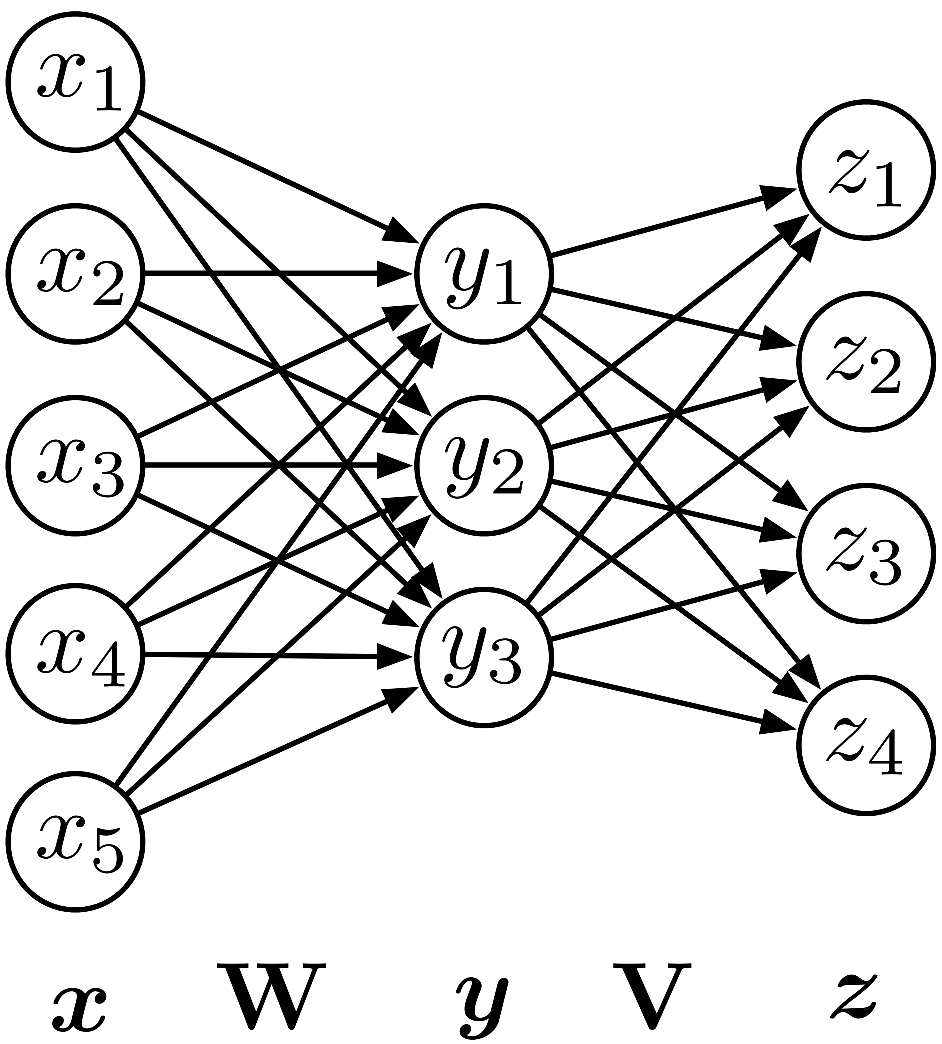
"Deep"



$$\boldsymbol{z} = g(\boldsymbol{y}^\top \mathbf{V})$$

$$\boldsymbol{z} = g(h(\boldsymbol{x}^\top \mathbf{W})^\top \mathbf{V})$$

“Deep”

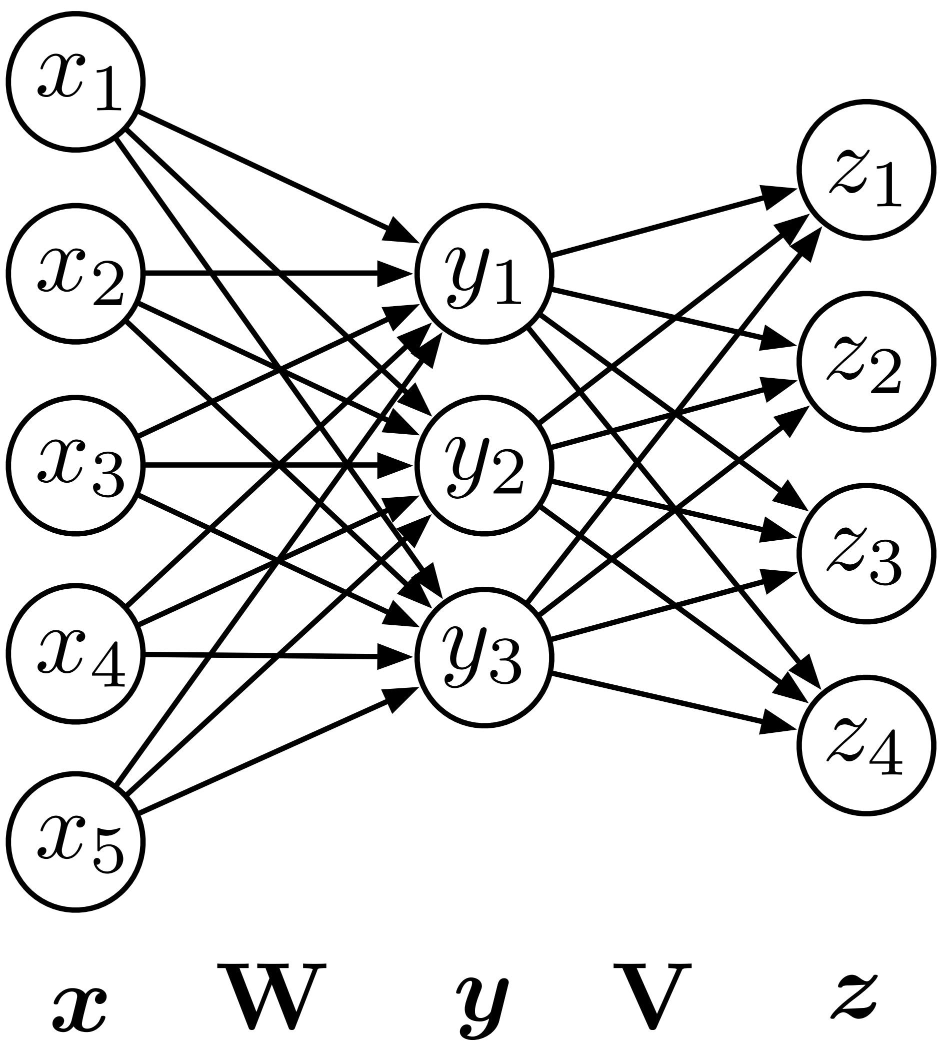


$$\boldsymbol{z} = g(\boldsymbol{y}^\top \mathbf{V})$$

$$\boldsymbol{z} = g(h(\boldsymbol{x}^\top \mathbf{W})^\top \mathbf{V})$$

$$\boldsymbol{z} = g(\mathbf{V} h(\mathbf{W} \boldsymbol{x}))$$

"Deep"



$$\boldsymbol{z} = g(\boldsymbol{y}^\top \mathbf{V})$$

$$\boldsymbol{z} = g(h(\boldsymbol{x}^\top \mathbf{W})^\top \mathbf{V})$$

$$\boldsymbol{z} = g(\mathbf{V} h(\mathbf{W}\boldsymbol{x}))$$

Note:

if $g(\boldsymbol{x}) = h(\boldsymbol{x}) = \boldsymbol{x}$

$$\boldsymbol{z} = \underbrace{(\mathbf{V}\mathbf{W})}_{\mathbf{U}} \boldsymbol{x}$$

Design Decisions

- How to represent inputs and outputs?
- Neural architecture?
  - How many layers? (Requires non-linearities to improve capacity!)
  - How many neurons?
  - Recurrent or not?
- What kind of non-linearities?

Representing Language

- “One-hot” vectors
  - Each position in a vector corresponds to a word type
  - Sequence of words, sequence of vectors
  - Bag of words: multiple vectors
- Distributed representations
  - Vectors encode “features” of input words (character n-grams, morphological features, etc.)

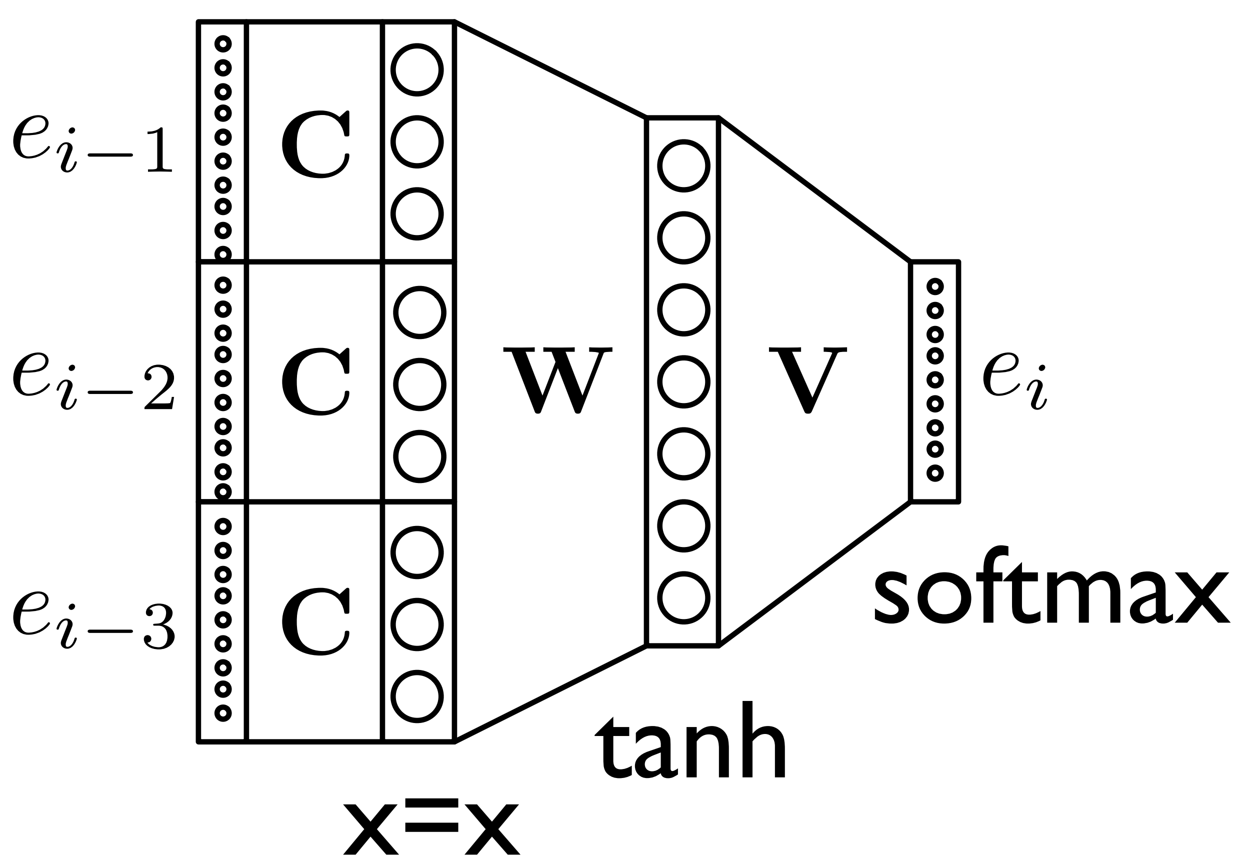
Training Neural Networks

- Neural networks are supervised models - you need a set of inputs paired with outputs
- Algorithm
  - Run for a while
    - Give input to the network, see what it predicts
    - Compute loss(y,y*) and (sub)gradient with respect to parameters. Use the chain rule, aka “back propagation”
    - Update parameters (SGD, AdaGrad, LBFGS, etc.)
- Algorithm is automated, just need to specify model structure

Bengio et al. (2003)

$$p(\mathbf{e}) = \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-n+1}, \ldots, e_{i-1})$$

$$p(e_i \mid e_{i-n+1}, \ldots, e_{i-1}) =$$



Bengio et al. (2003)

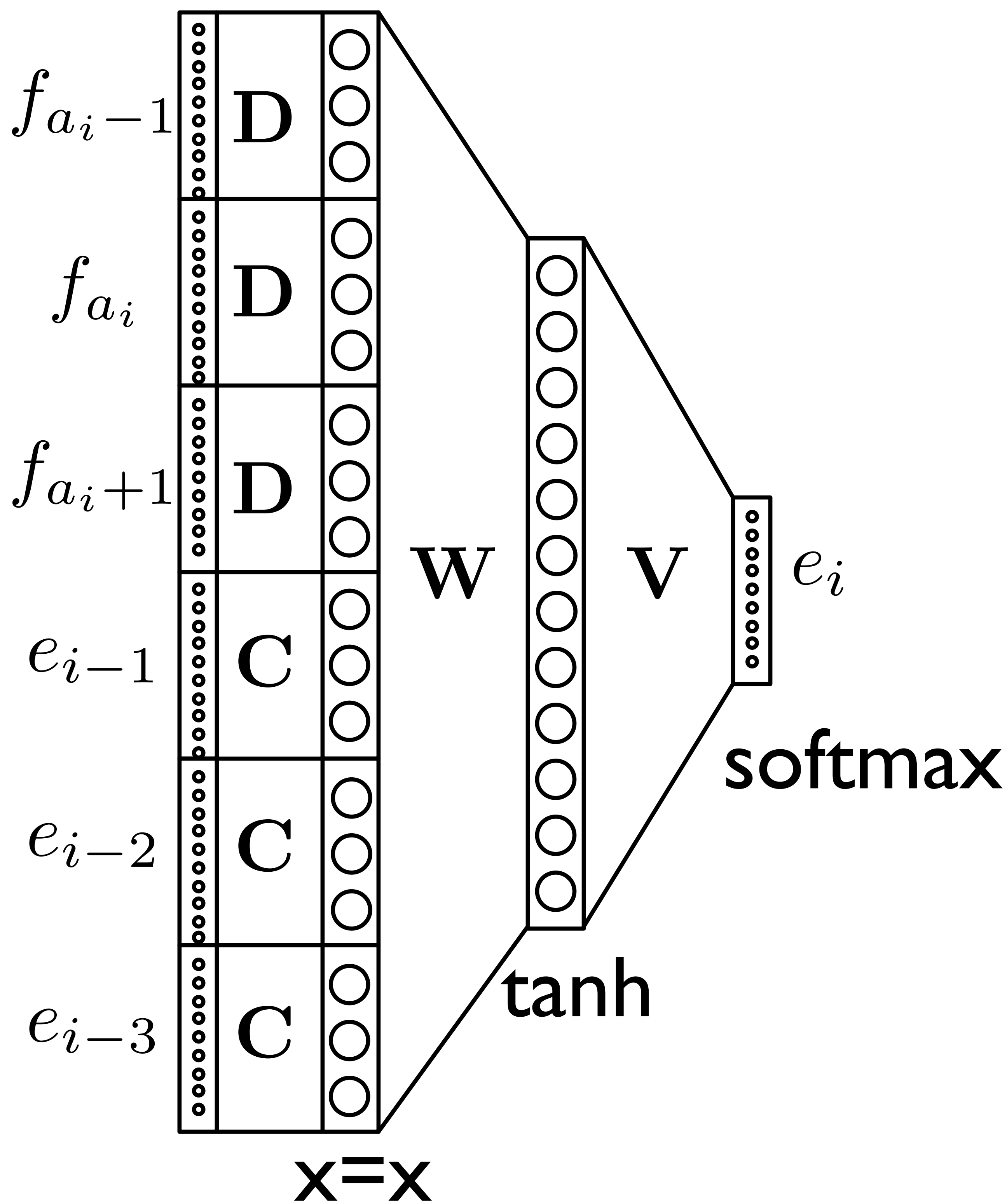
	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321

Devlin et al. (2014)

- Turn Bengio et al. (2003) into a translation model
- Conditional model; generate the next English word conditioned on
  - The previous *n* English words you generated
  - The aligned source word, and its *m* neighbors

$$p(\mathbf{e} \mid \mathbf{f}, \mathbf{a}) = \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-2}, e_{i-1}, f_{a_i-1}, f_{a_i}, f_{a_i+1})$$

$$p(e_i \mid e_{i-2}, e_{i-1}, f_{a_i-1}, f_{a_i}, f_{a_i+1}) =$$



Devlin et al. (2014)

BOLT Test		
	Ar-En	
	BLEU	**% Gain**
"Simple Hier." Baseline	33.8	-
S2T/L2R NNJM (Dec)	38.4	100%
Source Window=7	38.3	98%
Source Window=5	38.2	96%
Source Window=3	37.8	87%
Source Window=0	35.3	33%
Layers=384x768x768	38.5	102%
Layers=192x512	38.1	93%
Layers=128x128	37.1	72%
Vocab=64,000	38.5	102%
Vocab=16,000	38.1	93%
Vocab=8,000	37.3	83%
Activation=Rectified Lin.	38.5	102%
Activation=Linear	37.3	76%

Summary of neural LMs

- Two problems in standard statistical models
  - We don't condition on enough stuff
  - We don't know what features to use when we condition on lots of structure
- Neural networks let us condition on a lot of stuff without an exponential growth in parameters
- But: they are just reparameterized probability distributions. **Probability is central.**

LMs in MT

$$p(\mathbf{e}, \mathbf{f}) = p_{LM}(\mathbf{e}) \times p_{TM}(\mathbf{f} \mid \mathbf{e})$$

LMs in MT

$$
\begin{aligned}
p(\mathbf{e}, \mathbf{f}) &= p_{LM}(\mathbf{e}) \times p_{TM}(\mathbf{f} \mid \mathbf{e}) \\
&= \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-1}, \ldots, e_1) \times \prod_{j=1}^{|\mathbf{f}|} p(f_j \mid f_{j-1}, \ldots, f_1, \mathbf{e})
\end{aligned}
$$

LMs in MT

$$
\begin{aligned}
p(\mathbf{e}, \mathbf{f}) &= p_{LM}(\mathbf{e}) \times p_{TM}(\mathbf{f} \mid \mathbf{e}) \\
&= \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-1}, \ldots, e_1) \times \prod_{j=1}^{|\mathbf{f}|} p(f_j \mid f_{j-1}, \ldots, f_1, \mathbf{e})
\end{aligned}
$$

Conditional language model

LMs in MT

$$
\begin{aligned}
p(\mathbf{e}, \mathbf{f}) &= p_{LM}(\mathbf{e}) \times p_{TM}(\mathbf{f} \mid \mathbf{e}) \\
&= \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid \underbrace{e_{i-1}, \ldots, e_1}) \times \prod_{j=1}^{|\mathbf{f}|} p(f_j \mid \underbrace{f_{j-1}, \ldots, f_1}, \mathbf{e})
\end{aligned}
$$

Conditioned on entire sentence!
With direct multinomial parameterizations (n-gram models, IMB Model 1), we had to make independence assumptions for this to work.

LMs in MT

$$
\begin{aligned}
p(\mathbf{e}, \mathbf{f}) &= p_{LM}(\mathbf{e}) \times p_{TM}(\mathbf{f} \mid \mathbf{e}) \\
&= \prod_{i=1}^{|\mathbf{e}|} p(e_i \mid e_{i-1}, \ldots, e_1) \times \prod_{j=1}^{|\mathbf{f}|} p(f_j \mid f_{j-1}, \ldots, f_1, \mathbf{e})
\end{aligned}
$$