GENERAL ASSEMBLY

INTRO TO DATA SCIENCE
LECTURE 15: ADVANCED UNSUPERVISED LEARNING

I. LDA
II. DEEP LEARNING
III. AUTOENCODERS

EXERCISE:
IV. DIMENSIONALITY REDUCTION IN SCIKIT-LEARN

REVIEW: DIMENSIONALITY REDUCTION

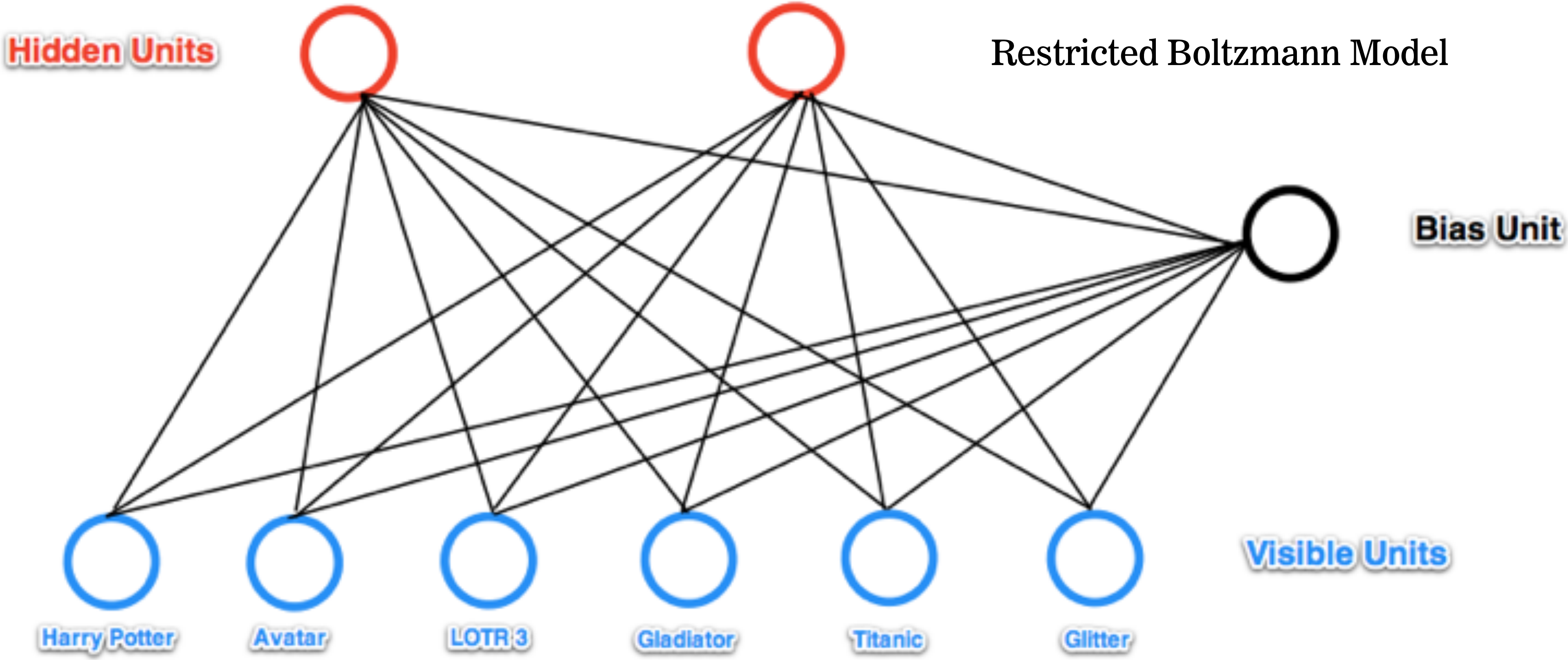
Q: What is dimensionality reduction?

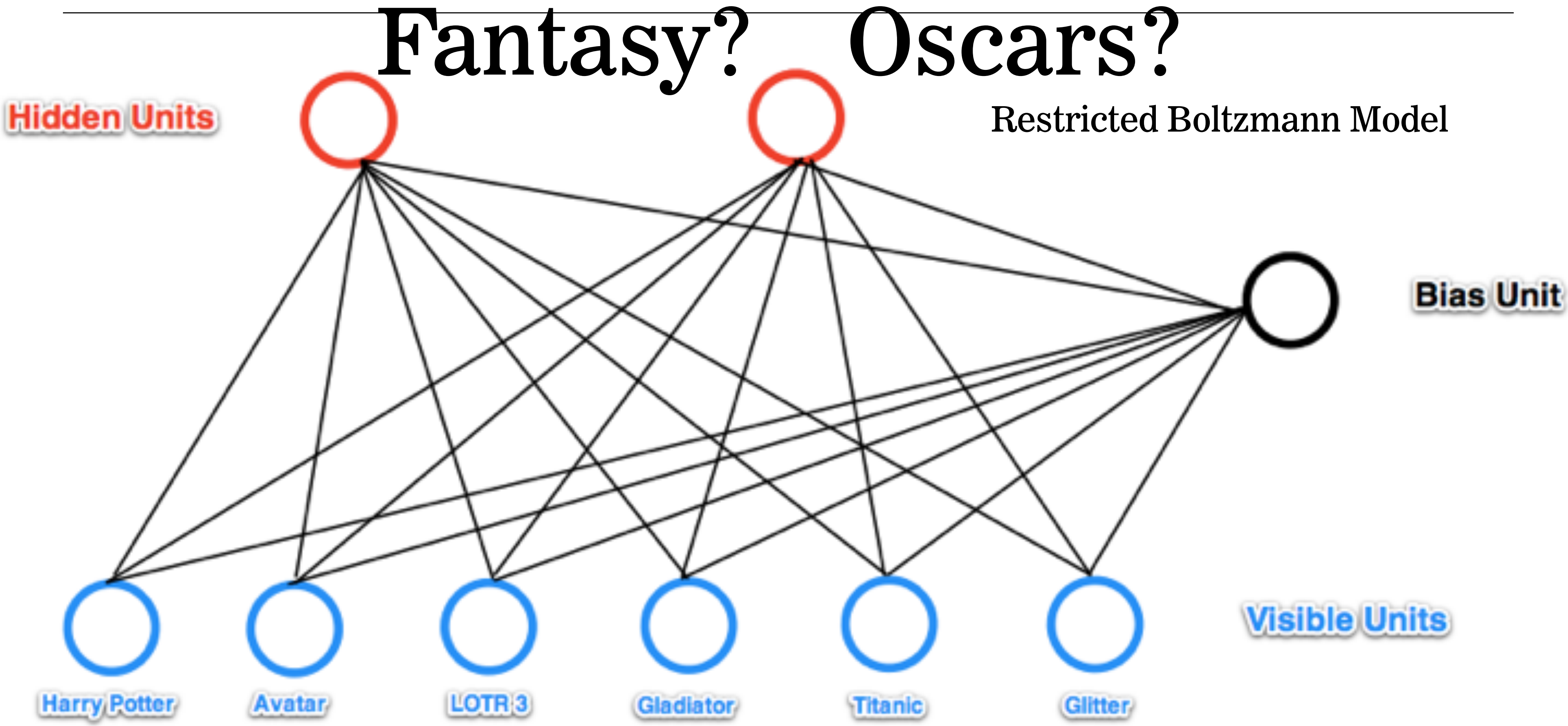
A: A set of techniques for reducing the size (in terms of features, records, and/or bytes) of the dataset under examination.

In general, the idea is to regard the dataset as a matrix and to decompose the matrix into simpler, meaningful pieces.

Dimensionality reduction is frequently performed as a pre-processing step before another learning algorithm is applied.

Restricted Boltzmann Model




Fantasy?
Oscars?
Hidden Units
Restricted Boltzmann Model
Bias Unit
Visible Units
Harry Potter
Avatar
LOTR 3
Gladiator
Titanic
Glitter

REVIEW: SINGULAR VALUE DECOMPOSITION

Consider a matrix A with n rows and d features.

The **singular value decomposition** *of A is given by:*

$$\underset{(n \times d)}{A} = \underset{(n \times k)}{U} \; \underset{(k \times k)}{\Sigma} \; \underset{(k \times d)}{V^T}$$

The **singular value decomposition** *of A is given by:*

$$\underset{(n \times d)}{A} = \underset{(n \times k)}{U} \; \underset{(k \times k)}{\Sigma} \; \underset{(k \times d)}{V^T}$$

The nonzero entries of Σ are the **singular values** *of A. These are real, nonnegative, and rank-ordered (decreasing from left to right).*

SINGULAR VALUE DECOMPOSITION

doc-to-concept similarity matrix

concepts strengths

term-to-concept similarity matrix

$$
\begin{array}{c}
\\
\mathrm{CS} \\
\\
\\
\\
\mathrm{MD} \\
\\
\end{array}
\begin{array}{ccccc}
\text{data} & \text{inf.} & \text{retrieval} & \text{brain} & \text{lung} \\
\end{array}
$$

$$
\begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
2 & 2 & 2 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 \\
5 & 5 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 & 2 \\
0 & 0 & 0 & 3 & 3 \\
0 & 0 & 0 & 1 & 1
\end{bmatrix}
=
\begin{bmatrix}
0.18 & 0 \\
0.36 & 0 \\
0.18 & 0 \\
0.90 & 0 \\
0 & 0.53 \\
0 & 0.80 \\
0 & 0.27
\end{bmatrix}
\times
\begin{bmatrix}
9.64 & 0 \\
0 & 5.29
\end{bmatrix}
\times
\begin{bmatrix}
0.58 & 0.58 & 0.58 & 0 & 0 \\
0 & 0 & 0 & 0.71 & 0.71
\end{bmatrix}
$$

I. LATENT DIRICHLET ALLOCATION

Bayes' theorem. *Here it is again:*

$$P(A|B) = P(B|A) * P(A) \, / \, P(B)$$

Some facts:

- This is a simple algebraic relationship using elementary definitions.

- It's interesting because it's kind of a "wormhole" between two different "interpretations" of probability.

- It's a very powerful computational tool.

This term is the **likelihood function**. *It represents the joint probability of observing features* $\{x_i\}$ *given that that record belongs to class* C.

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

This term is the **prior probability** *of* C. *It represents the probability of a record belonging to class* C *before the data is taken into account.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

This term is the **normalization constant.** *It doesn't depend on* C*, and is generally ignored until the end of the computation.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

This term is the **posterior probability** *of C. It represents the probability of a record belonging to class C after the data is taken into account.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

*This term is the **posterior probability** of C. It represents the probability of a record belonging to class C after the data is taken into account.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

The goal of any Bayesian computation is to find ("learn") the posterior distribution of a particular variable.

Maximum likelihood estimator (MLE):

What parameters ***maximize*** *the likelihood function?*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

Maximum a posteriori estimate (MAP):

*What parameters **maximize** the likelihood function **AND** prior?*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

We observe the following coin flips:

HTHHTHT

Maximum likelihood estimator (MLE):

*What parameters **maximize** the likelihood function?*

Let P(X = Heads) = q, and write Bayes Theorem

P(q | observations) = P (observations | q) * P (q) / constant

Maximum likelihood estimator (MLE):

What parameters ***maximize*** *the likelihood function?*

Let P(X = Heads) = q, and write Bayes Theorem

P(q | observations) = P (observations | q) * P (q) / constant

P(observations | q) = ?

P(q) = ?

Maximum likelihood estimator (MLE):

What parameters ***maximize*** *the likelihood function?*

Let P(X = Heads) = q, and write Bayes Theorem

P(q | observations) = P (observations | q) * P (q) / constant

P(observations | q) = Binomial Distribution

P(q) = ????

A prior distribution is known as **conjugate prior** if its from the same family as the posterior for a certain likelihood function

For the binomial distribution, the conjugate prior is the **Beta distribution**

$$= \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

$$= \frac{1}{\mathrm{B}(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}$$

The **MAP estimate** is the value that maximizes both the likelihood function and prior - the product of the two.

In the coin flip setting is the value that optimizes

$$P(\text{HTHHTHT} \mid q) * P(q)$$

The **MAP estimate** is the value that maximizes both the likelihood function and prior - the product of the two.

In the coin flip setting is the value that optimizes

P (HTHHTHT | q) * P(q)

= (7 choose 4) q ^ 4 * (1 -q) ^3 * q^(a-1) * (1-a) ^(b-1)

The **MAP estimate** is the value that maximizes both the likelihood function and prior - the product of the two.

In the coin flip setting is the value that optimizes

$P (HTHHTHT | q) * P(q)$

$= (7 \text{ choose } 4) q ^ 4 * (1 -q) ^3 * q^(a-1) * (1-a) ^(b-1)$

$= q^(4 + a -1) * (1-q)^ (3 + b - 1)$

The **MAP estimate** is the value that maximizes both the likelihood function and prior - the product of the two.

In the coin flip setting is the value that optimizes

P (HTHHTHT | q) * P(q)

= (7 choose 4) q ^ 4 * (1 -q) ^3 * q^(a-1) * (1-a) ^(b-1)

= q^(4 + a -1) * (1-q)^ (3 + b - 1)

After optimizing, the **MAP is (4 + a -1) / (7 + a + b - 2)**

Why do you care?

Why do you care?

Many problems are binary and are estimated using counts...

Suppose we have a dataset with features $x_1, \ldots, x_n$ and a class label c. What can we say about classification using Bayes' theorem?

Suppose we have a dataset with features $x_1, \ldots, x_n$ and a class label C. What can we say about classification using Bayes' theorem?

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

Bayes' theorem can help us to determine the probability of a record belonging to a class, given the data we observe.

source: Data Analysis with Open Source Tools, by Philipp K. Janert. O'Reilly Media, 2011.

The idea of Bayesian inference, then, is to **update** *our beliefs about the distribution of C using the data ("evidence") at our disposal.*

$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$

Then we can use the posterior for prediction.

Q: What piece of the puzzle we've seen so far looks like it could intractably difficult in practice?

Remember the likelihood function?

$$P(\{x_i\}|C) = P(\{x_1, x_2, \ldots, x_n\})|C)$$

Remember the likelihood function?

$$P(\{x_i\}|C) = P(\{x_1, x_2, \ldots, x_n\})|C)$$

Observing this exactly would require us to have enough data for every possible combination of features to make a reasonable estimate.

Q: What piece of the puzzle we've seen so far looks like it could intractably difficult in practice?

A: Estimating the full likelihood function.

Q: So what can we do about it?

Q: So what can we do about it?

A: Make a simplifying assumption. In particular, we assume that the features x_i are conditionally independent from each other:

Q: So what can we do about it?

A: Make a simplifying assumption. In particular, we assume that the features x_i are conditionally independent from each other:

$$P(\{x_i\}|C) = P(x_1, x_2, \ldots, x_n|C) \approx P(x_1|C) * P(x_2|C) * \ldots * P(x_n|C)$$

Q: So what can we do about it?

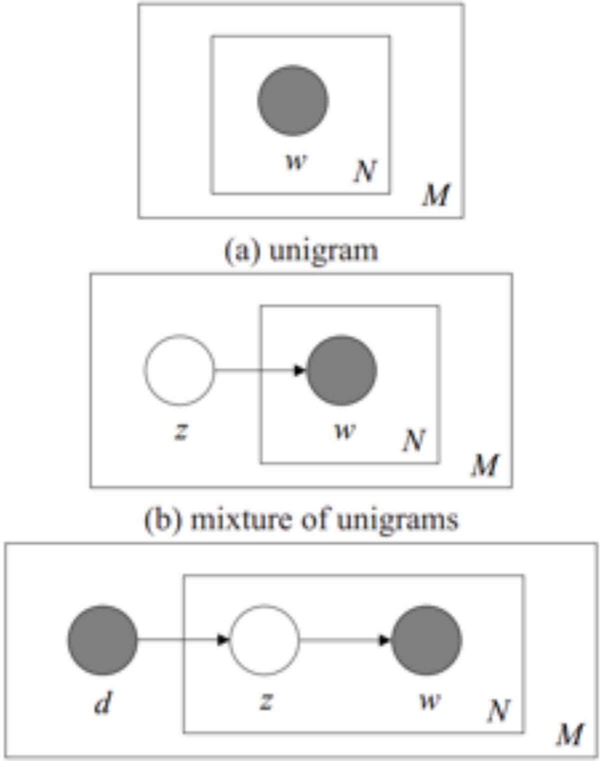
A: Make a simplifying assumption. In particular, we assume that the features x_i are conditionally independent from each other:

$$P(\{x_i\}|C) = P(x_1, x_2, \ldots, x_n|C) \approx P(x_1|C) * P(x_2|C) * \ldots * P(x_n|C)$$

This "naïve" assumption simplifies the likelihood function to make it tractable.

This term is the **posterior probability** *of* C*. It represents the probability of a record belonging to class* C *after the data is taken into account.*

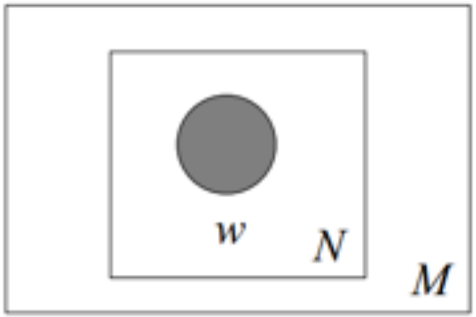
$$P(\text{class } C \mid \{x_i\}) = \frac{P(\{x_i\} \mid \text{class } C) \cdot P(\text{class } C)}{P(\{x_i\})}$$



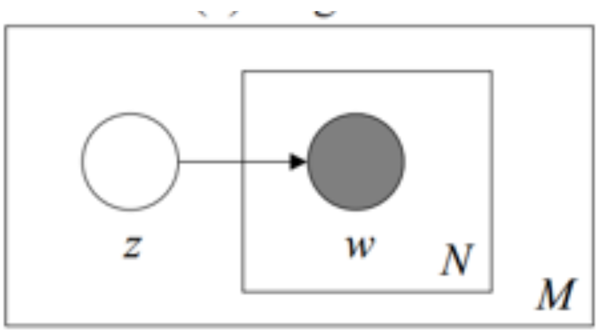
(a) unigram

(b) mixture of unigrams

(c) pLSI/aspect model



(a) unigram



(b) mixture of unigrams

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

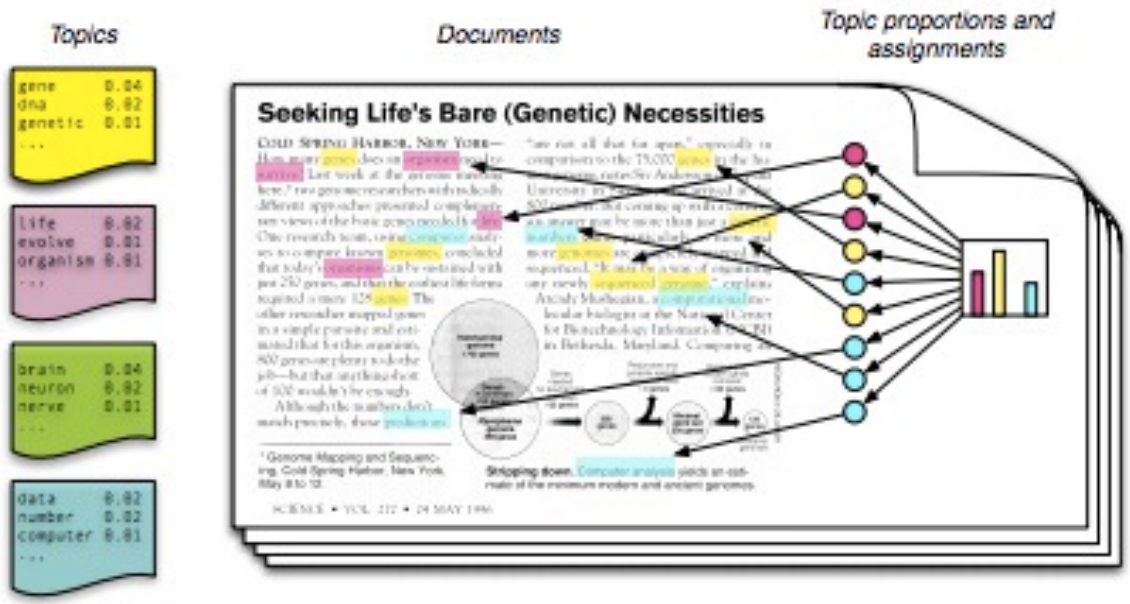
Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

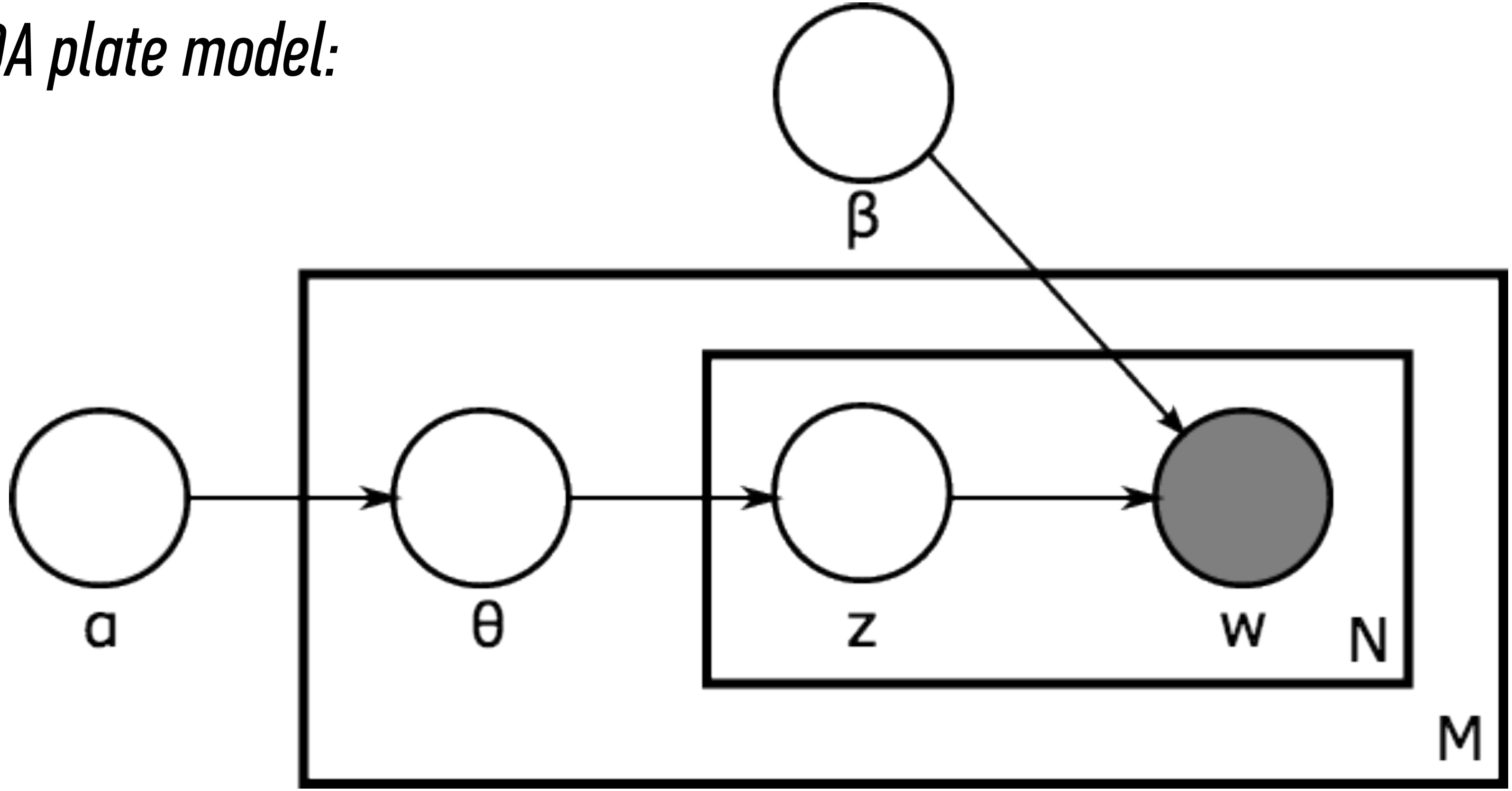
SCIENCE • VOL. 272 • 24 MAY 1996

Simple intuition: Documents exhibit multiple topics.

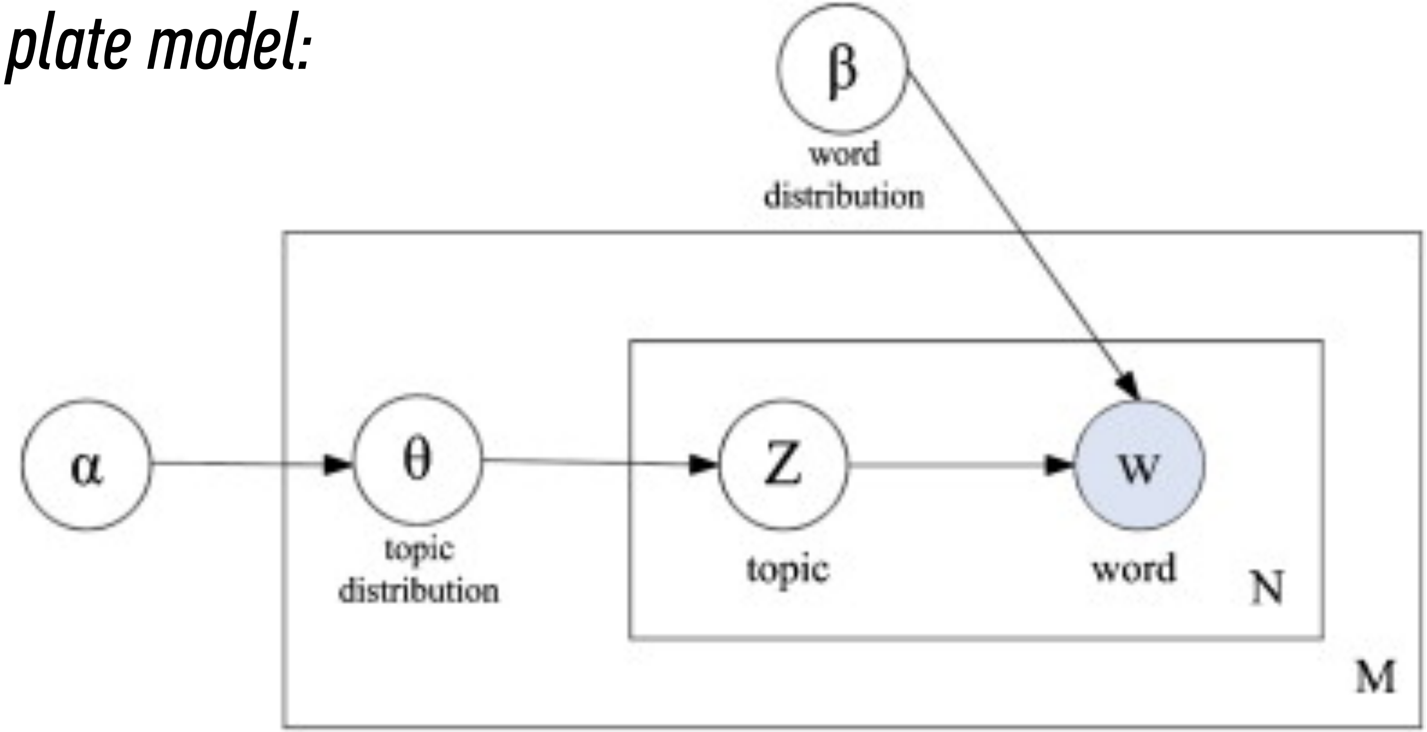


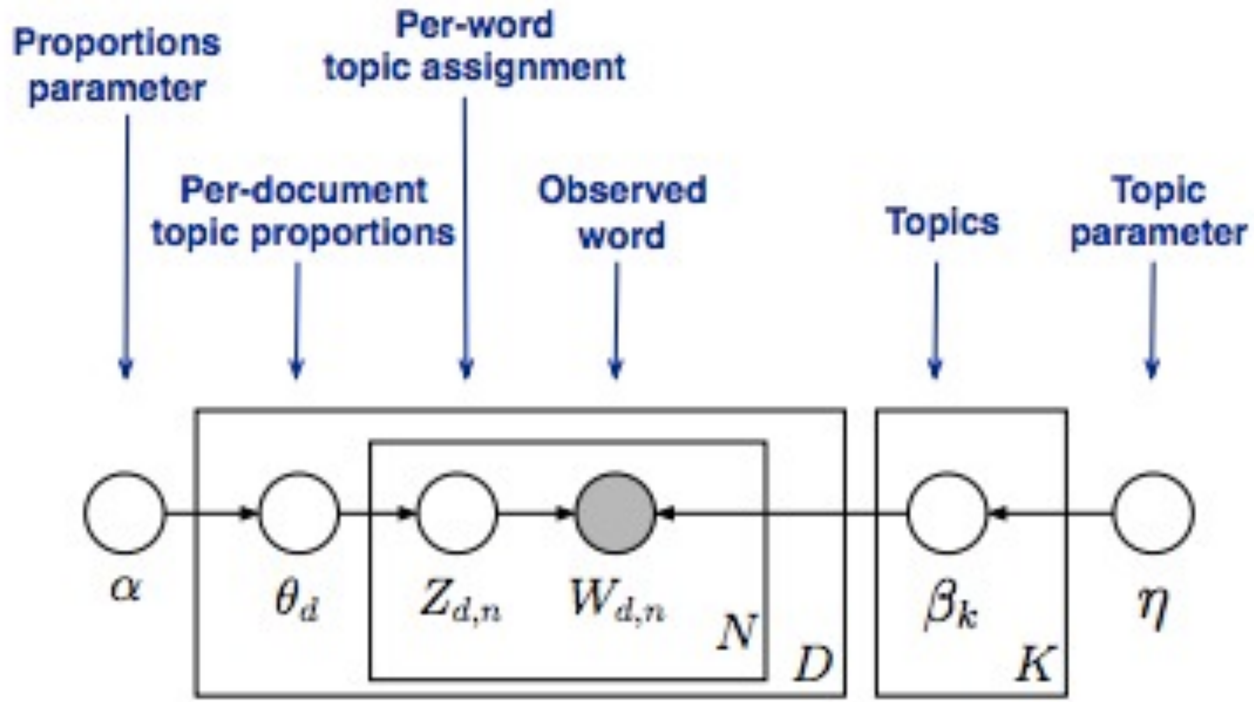
- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics
- Each **word** is drawn from one of those topics

LDA plate model:



LDA plate model:




Proportions parameter
Per-word topic assignment
Per-document topic proportions
Observed word
Topics
Topic parameter
α
θ_d
$Z_{d,n}$
$W_{d,n}$
N
D
β_k
K
η

LDA as a generative process:

1. Choose $\theta_i \sim \mathrm{Dir}(\alpha)$, where $i \in \{1, \dots, M\}$ and $\mathrm{Dir}(\alpha)$ is the Dirichlet distribution for parameter α
2. Choose $\phi_k \sim \mathrm{Dir}(\beta)$, where $k \in \{1, \dots, K\}$
3. For each of the word positions i, j, where $j \in \{1, \dots, N_i\}$, and $i \in \{1, \dots, M\}$
   (a) Choose a topic $z_{i,j} \sim \mathrm{Multinomial}(\theta_i)$.
   (b) Choose a word $w_{i,j} \sim \mathrm{Multinomial}(\phi_{z_{i,j}})$.

LDA in Bayes Formula:

$$p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta) = p(\theta \mid \alpha) \prod_{n=1}^{N} p(z_n \mid \theta) p(w_n \mid z_n, \beta)$$

II. UNSUPERVISED FEATURE EXTRACTION

Deep learning *is*

Deep learning *is*

"... a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence"

-- deeplearning.net

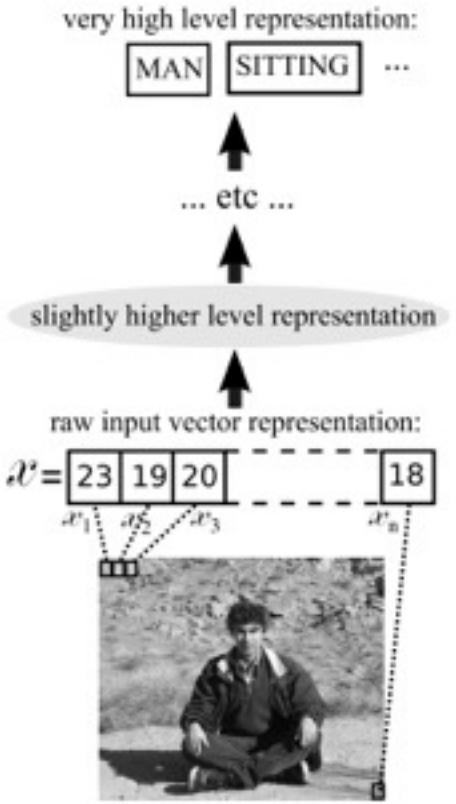
Deep learning *is*

"*...* composition of multiple non-linear transformations of the data"

-- Yoshua Bengio

From Yoshua Bengio's *Representation Learning: A Review and New Perspectives*

"The performance of machine learning methods is **heavily dependent on the choice of data representation (or features)** *... much of the actual effort in deploying machine learning algorithms goes into the* **design of preprocessing pipelines and data transformations** *that result in a representation.*


very high level representation:
MAN
SITTING
...
... etc ...
slightly higher level representation
raw input vector representation:
x=
23
19
20
18
x_1
x_2
x_3
x_n

Yoshua Bengio (Montreal)

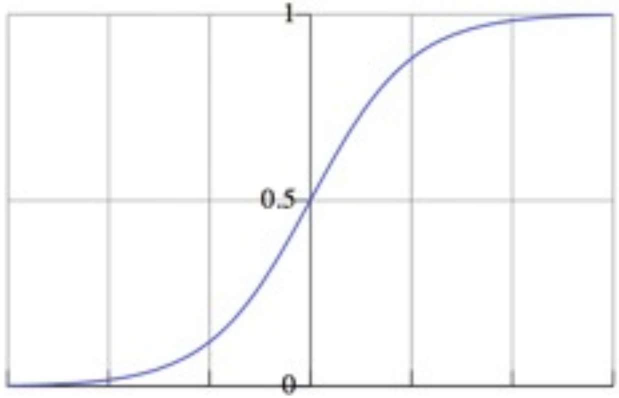
Yann LeCun (NYU, now Facebook)

Geoff Hinton (Toronto, now Google)

Andrew Ng (Stanford)

The **logistic function***:*

$$E(y|x) = \pi(x) = \frac{e^{\alpha+\beta x}}{1+e^{\alpha+\beta x}}$$



The **logit function** *is an important transformation of the logistic function. Notice that it returns the linear model!*

$$g(x) = ln(\frac{\pi(x)}{1-\pi(x)}) = \alpha + \beta x$$

Q: What do the terms in this model mean?

$$y = \alpha + \beta x + \varepsilon$$

A: y = **response variable** *(the one we want to predict)*

x = **input variable** *(the one we use to train the model)*

α = **intercept** *(where the line crosses the y-axis)*

β = **regression coefficient** *(the model "parameter")*

ε = **residual** *(the prediction error)*

Multiple Linear Regression *model:*

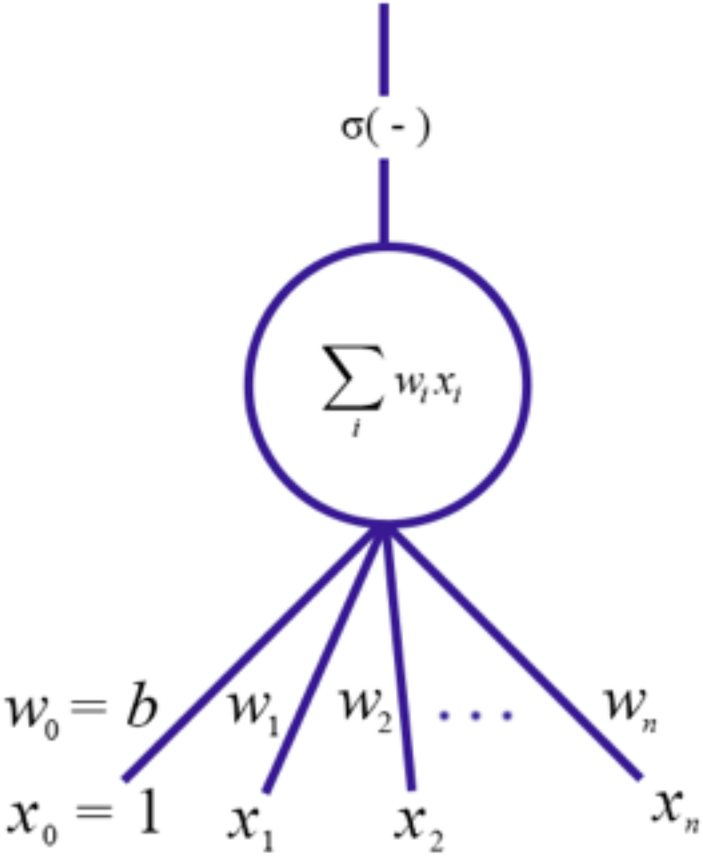
$$y = \alpha + w_1 x_1 + \ldots + w_n x_n + \varepsilon$$

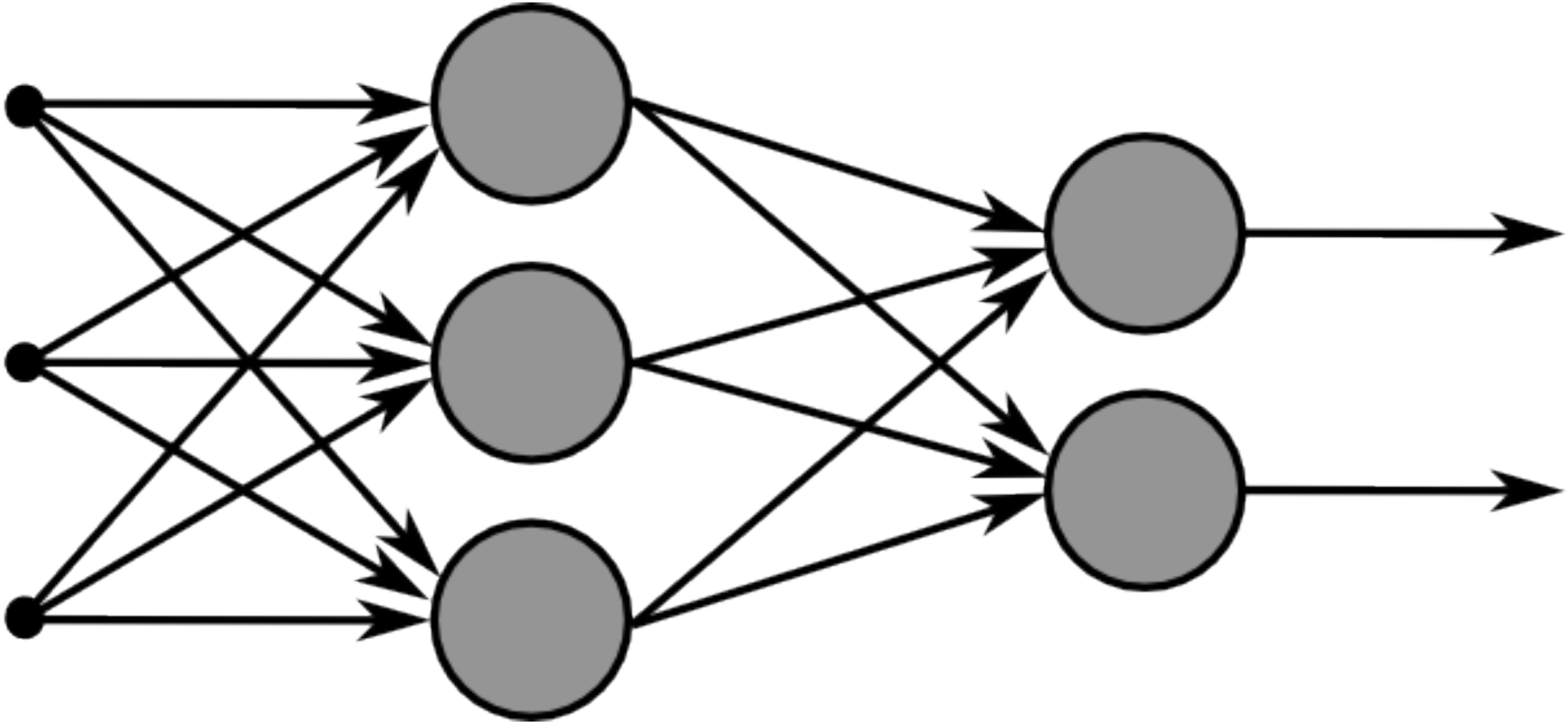
Multiple Linear Regression *model:*

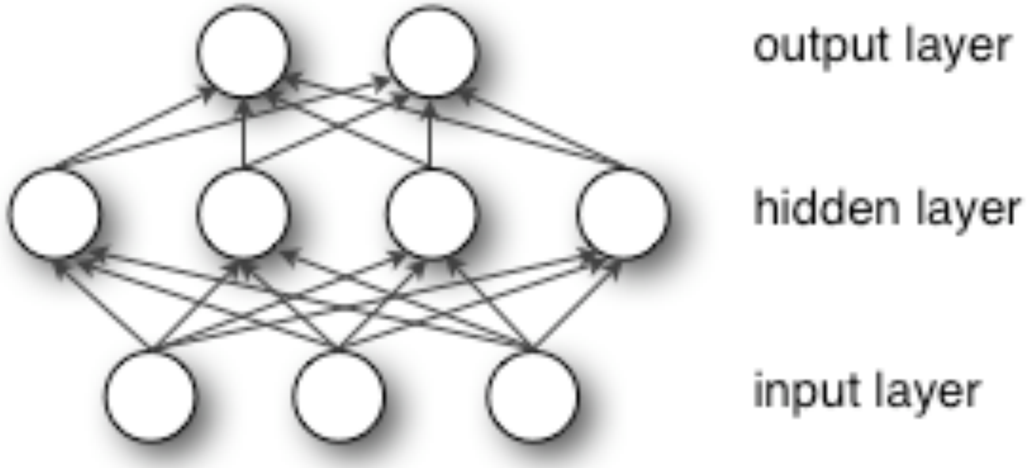
$$y = \alpha + w_1 x_1 + \ldots + w_n x_n + \varepsilon$$

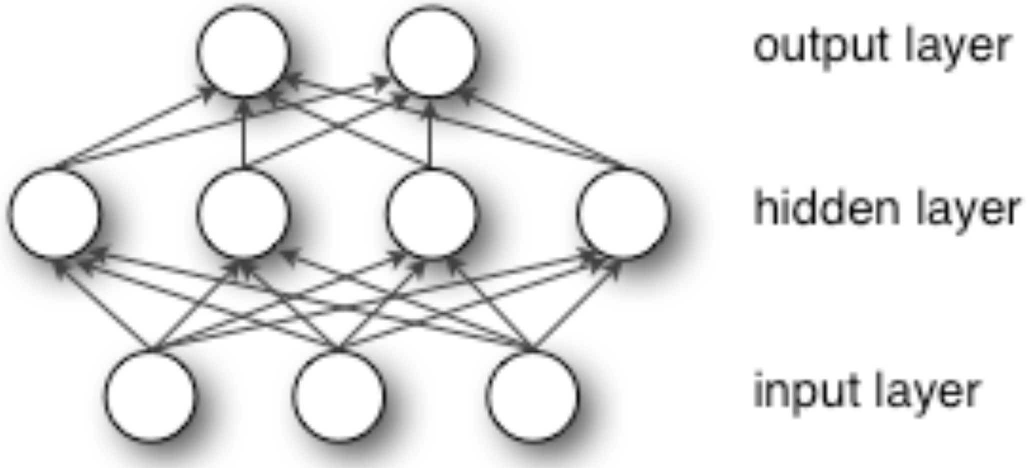
equivalent to:

SUM w_i * x_i = dot_product(w, x)


σ(-)
$\sum_i w_i x_i$
$w_0 = b$
w_1
w_2
$\cdots$
w_n
$x_0 = 1$
x_1
x_2
x_n

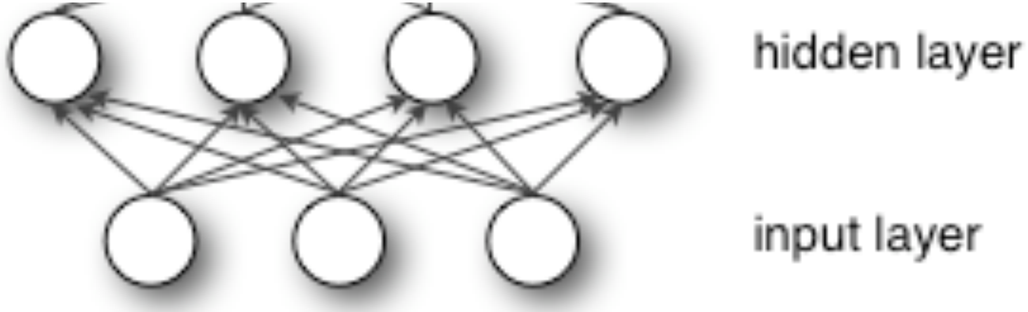



output layer
hidden layer
input layer

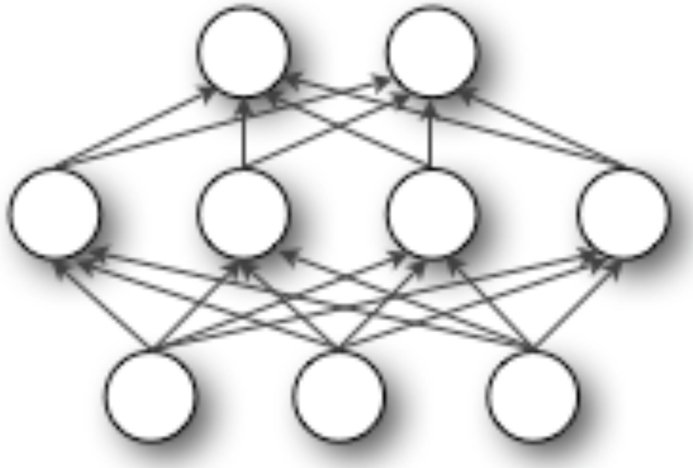


$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

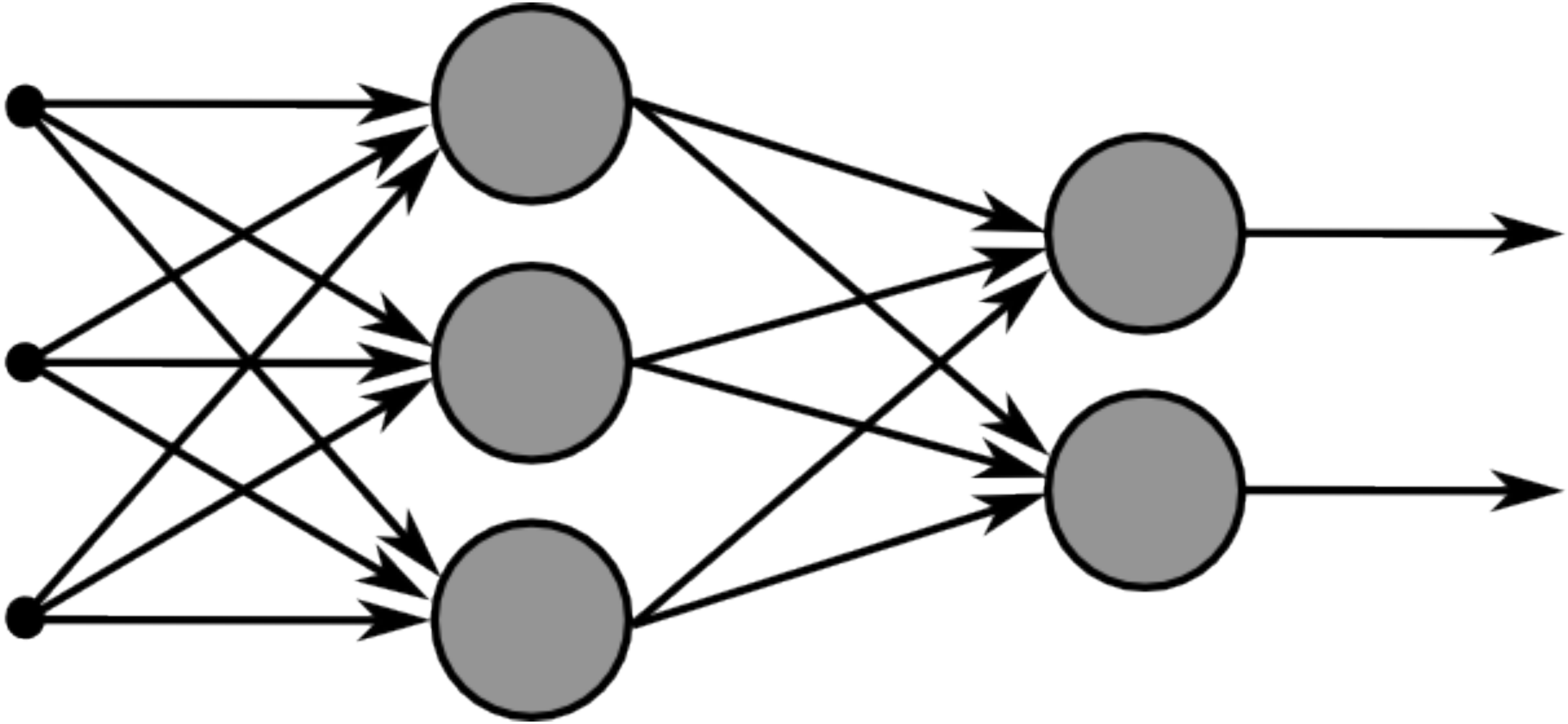
$$y = sigmoid(\ \alpha + w_1 x_1 + \ \ldots\ + w_n x_n + \varepsilon\ )$$

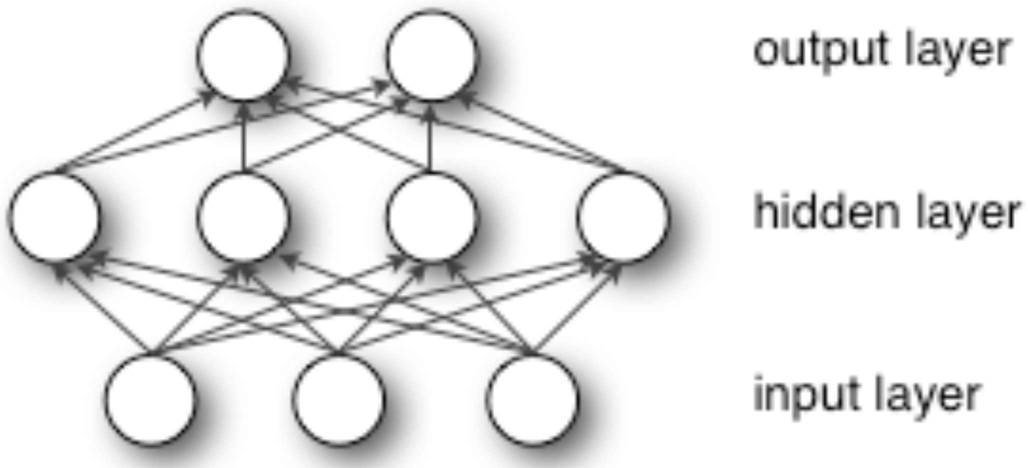


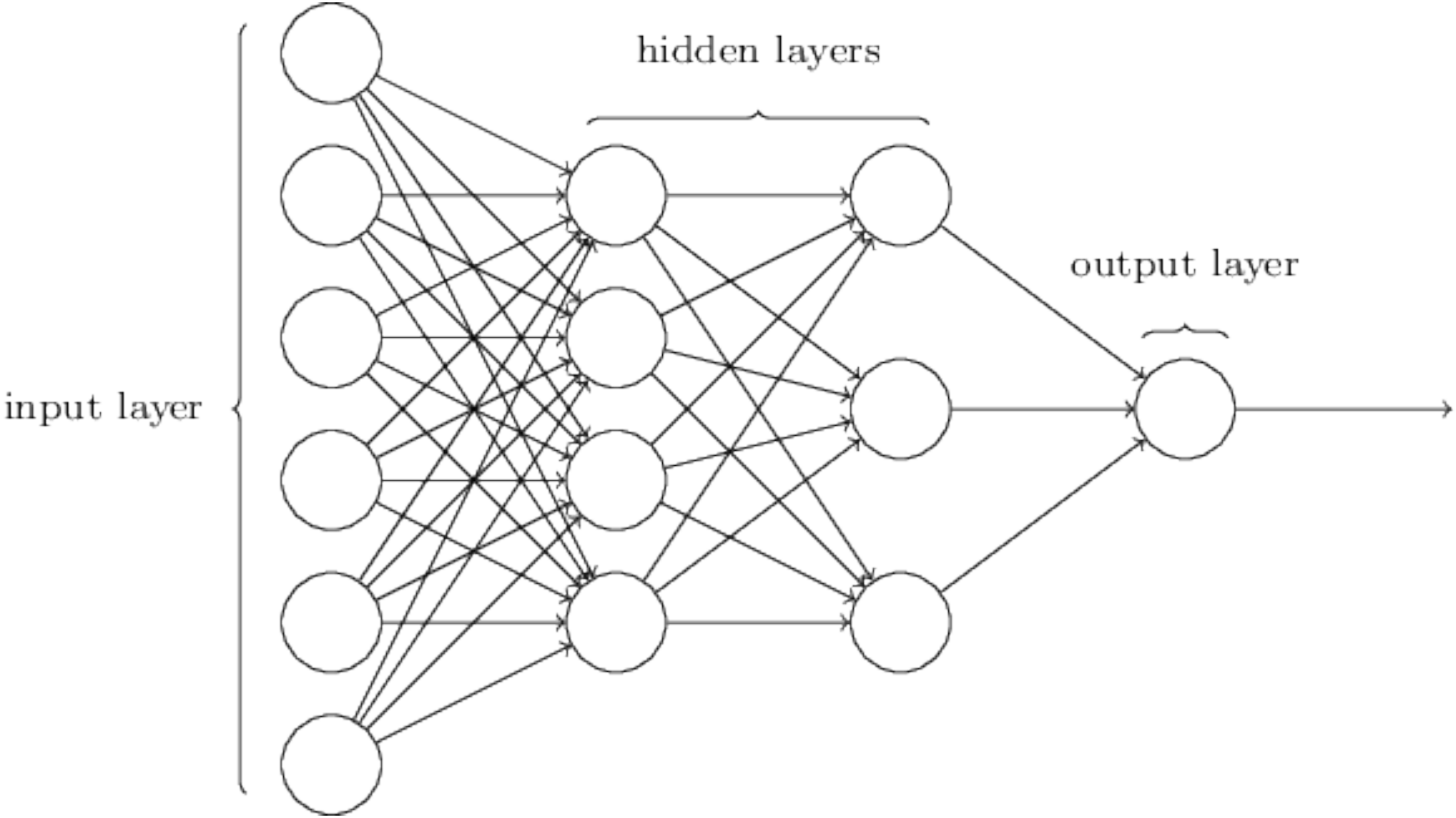
$$(s(b^{(1)} + W^{(1)}x))),$$

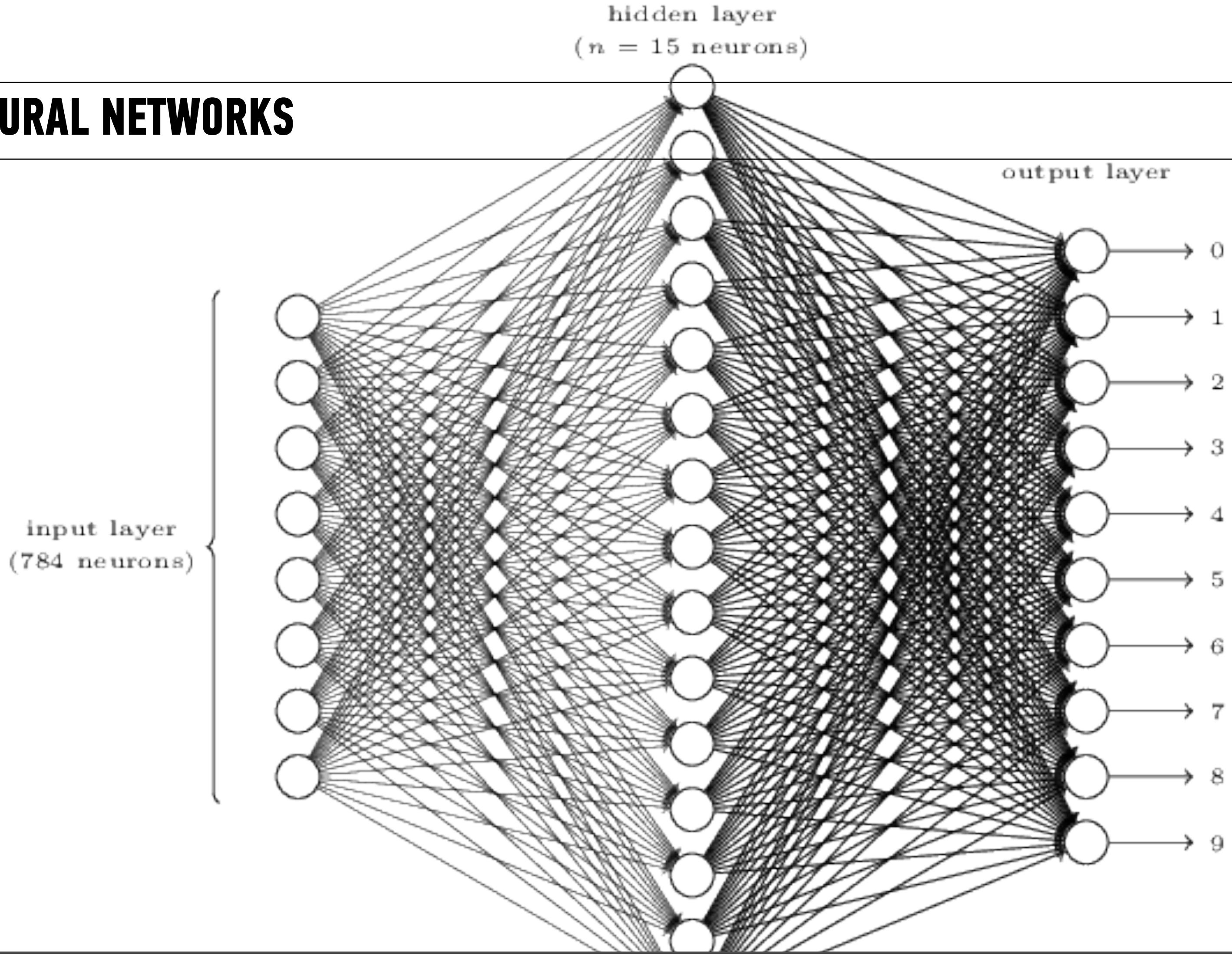


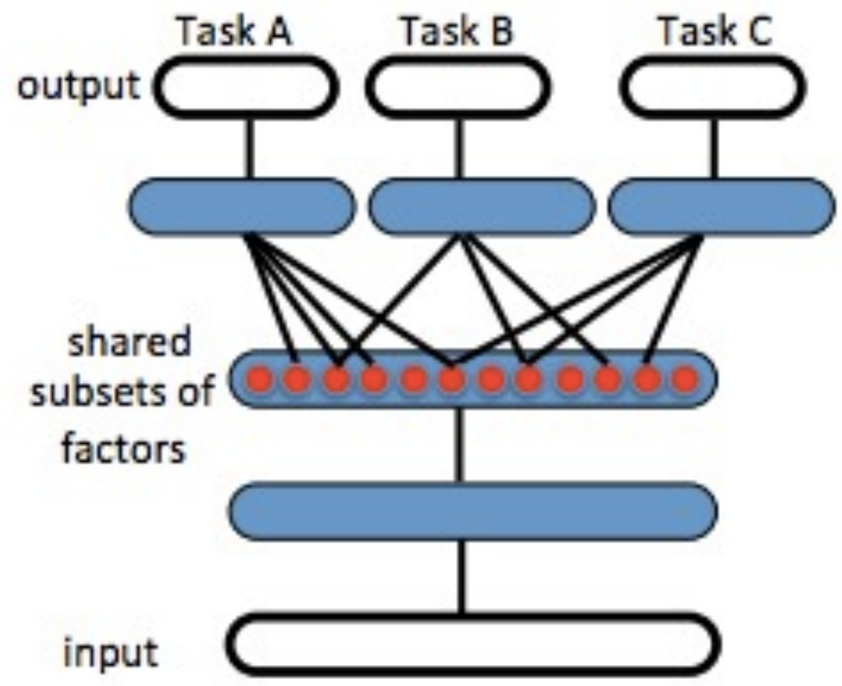
$$f(x) = G(b^{(2)} + W^{(2)}(s(b^{(1)} + W^{(1)}x))),$$

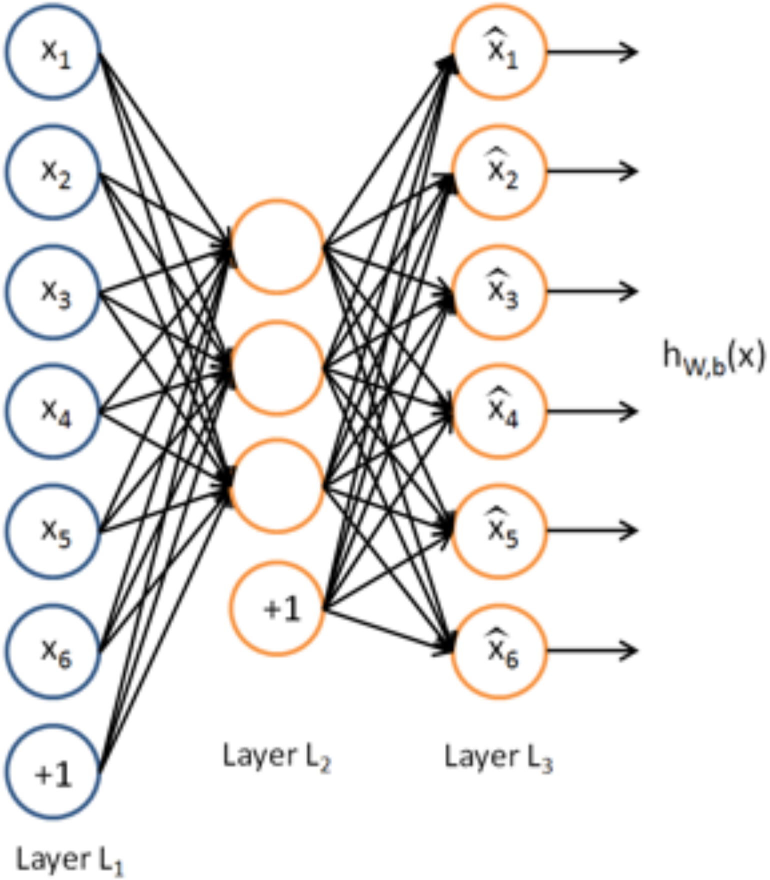



output layer
hidden layer
input layer


hidden layers
output layer
input layer


hidden layer
(n = 15 neurons)
output layer
input layer
(784 neurons)
0
1
2
3
4
5
6
7
8
9


Task A
Task B
Task C
output
shared
subsets of
factors
input

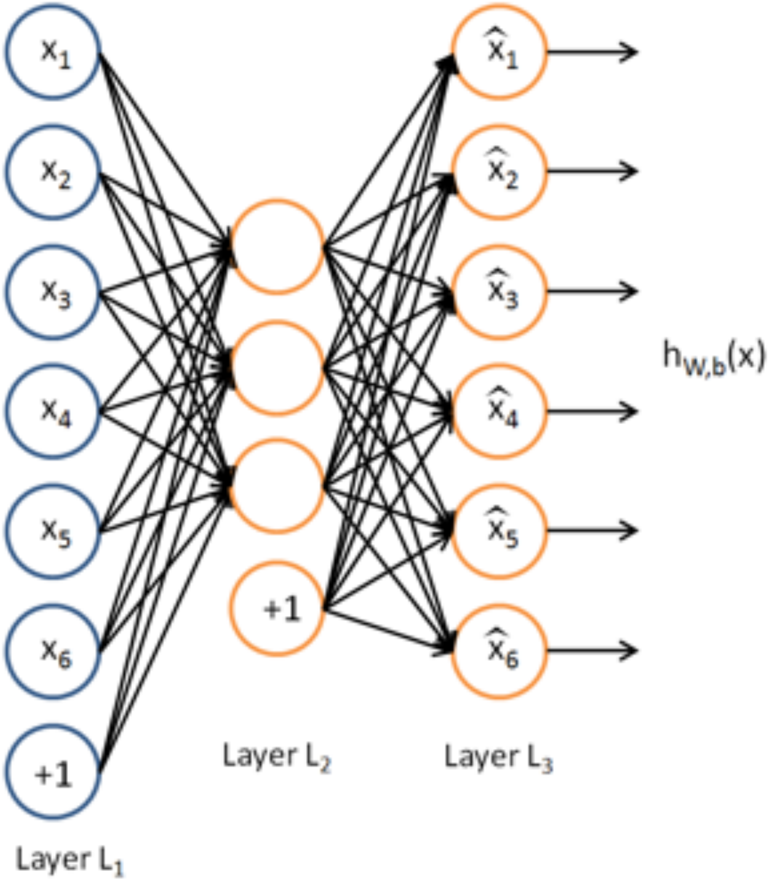


x_1
x_2
x_3
x_4
x_5
x_6
+1
+1
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$
$h_{W,b}(x)$
Layer L_1
Layer L_2
Layer L_3

Goal: *Learn parameters that can predict the input*

$$x' = g(f(x))$$

We want x' very close to x


x_1
x_2
x_3
x_4
x_5
x_6
+1
+1
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$
$h_{W,b}(x)$
Layer L_1
Layer L_2
Layer L_3

Goal: *Learn parameters that can predict the input*

$$x' = g(f(x))$$

We want x' very close to x

If g and f are linear, then we just need to learn a matrix

Goal: *Learn parameters that can predict the input*

$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$$

Goal: *Learn parameters that can predict the input*

$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$$

We need to learn two matrices of parameters:

W – n x n matrix of parameters to "encode" input x

W ` - n x n matrix of parameters to decode "encoded" x

Goal: *Learn parameters that can predict the input*

$$\mathbf{y} = s(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{z} = s(\mathbf{W}'\mathbf{y} + \mathbf{b}')$$

Optimization Criteria:

$$L(\mathbf{x}, \mathbf{z}) = ||\mathbf{x} - \mathbf{z}||^2$$

$$L_H(\mathbf{x}, \mathbf{z}) = -\sum_{k=1}^{d} [\mathbf{x}_k \log \mathbf{z}_k + (1 - \mathbf{x}_k) \log(1 - \mathbf{z}_k)]$$

Goal: *Learn parameters that can predict the input*

What's best f and g we can learn? Why is it bad?

Goal: *Learn parameters that can predict the input*

What's best f and g we can learn? Why is it bad?

How do we fix it?

Goal: *Learn parameters that can predict the input*

$$x' = g(f(x))$$

Goal: *Learn parameters that can predict the input:*

Denoising Autoencoder:

$$x' = g(f(x + \text{random error}))$$

Goal: *Learn parameters that can predict the input*

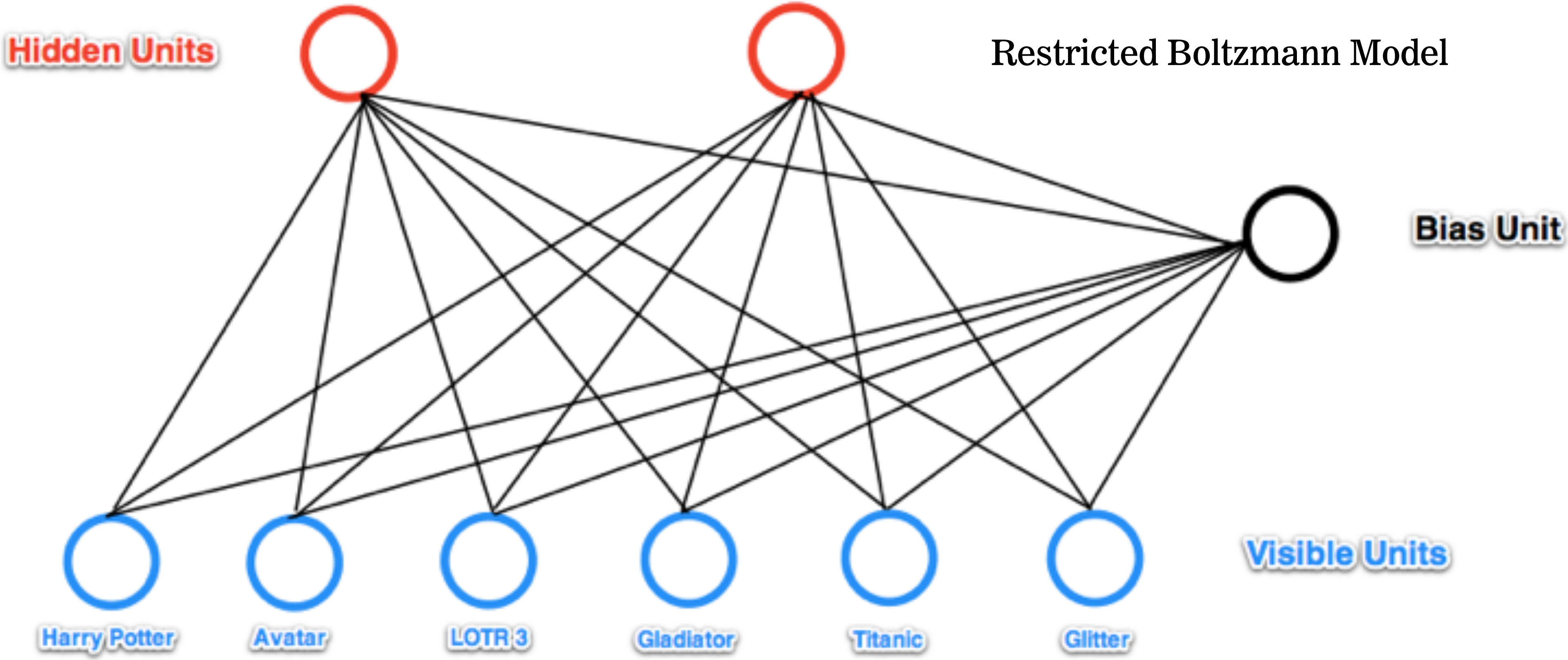
Regularized Autoencoder:

$$\mathcal{J}_{\mathrm{AE+wd}}(\theta) = \left(\sum_{x \in D_n} L(x, g(f(x))) \right) + \lambda \sum_{ij} W_{ij}^2$$

Goal: *Learn parameters that can predict the input*

Contractive Autoencoder:

$$\mathcal{J}_{\mathrm{CAE}}(\theta) = \sum_{x \in D_n} \left(L(x, g(f(x))) + \lambda \|J_f(x)\|_F^2 \right)$$


Hidden Units
Restricted Boltzmann Model
Bias Unit
Visible Units
Harry Potter
Avatar
LOTR 3
Gladiator
Titanic
Glitter

word2vec : C implementation of Recurrent NN for word representations

Theano : Advanced function optimization

Pylearn2 :

Deep learning library based on Theano

Developed at Montreal (Lisa Lab)