Feature Extraction

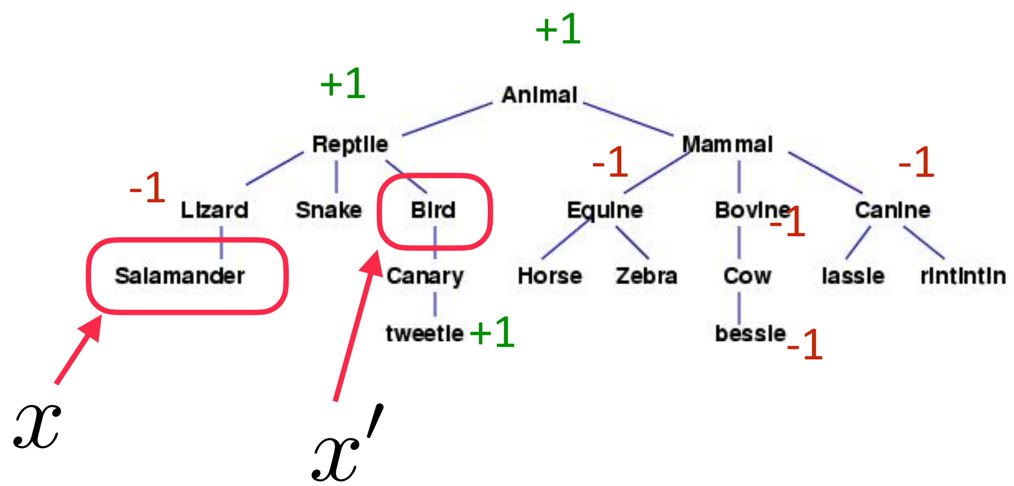
Feature extraction

Data comes in all forms:

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \ldots, 2.5]^\top$

Categorical data $x = [\texttt{Red}, 98105, \texttt{Finished basement}, , \ldots, 2.5]^\top$

Structured data



Given tree and labels are known at some nodes, how do we predict unknown labels?

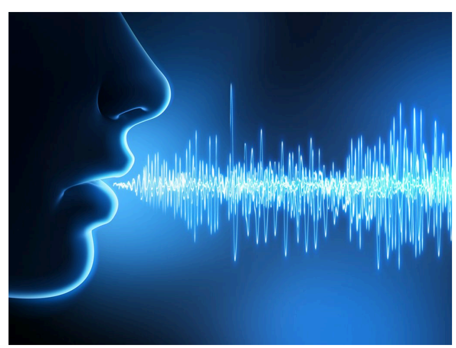
Feature extraction

Data comes in all forms:

Text data

Image data

Audio data



Time-series data

Feature Extraction given real-valued data

Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \ldots, 2.5]^\top$

Strategies if many features are **uninformative**?

Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \ldots, 2.5]^\top$

Strategies if many features are **incomparable**?

Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \ldots, 2.5]^\top$

Strategies if many features are **superfluous** or correlated with each other?

Feature extraction - real vectors

Real, continuous features $x \in \mathbb{R}^d$ $x = [0.1, 4.0, 4.3, \ldots, 2.5]^\top$

Pre-processing pipeline:

1. Standardize data (de-mean, divide by standard deviation)
2. Project down to lower dimensional representation using PCA
3. Apply exact transformation to Train and Test.

Autoencoders

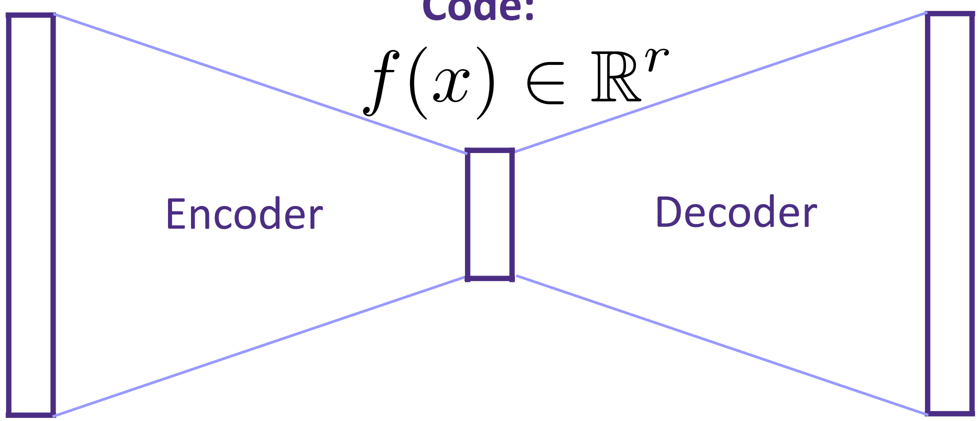
Find a low dimensional representation for your data by predicting your data

Code:

$f(x) \in \mathbb{R}^r$

Input:

$x \in \mathbb{R}^d$

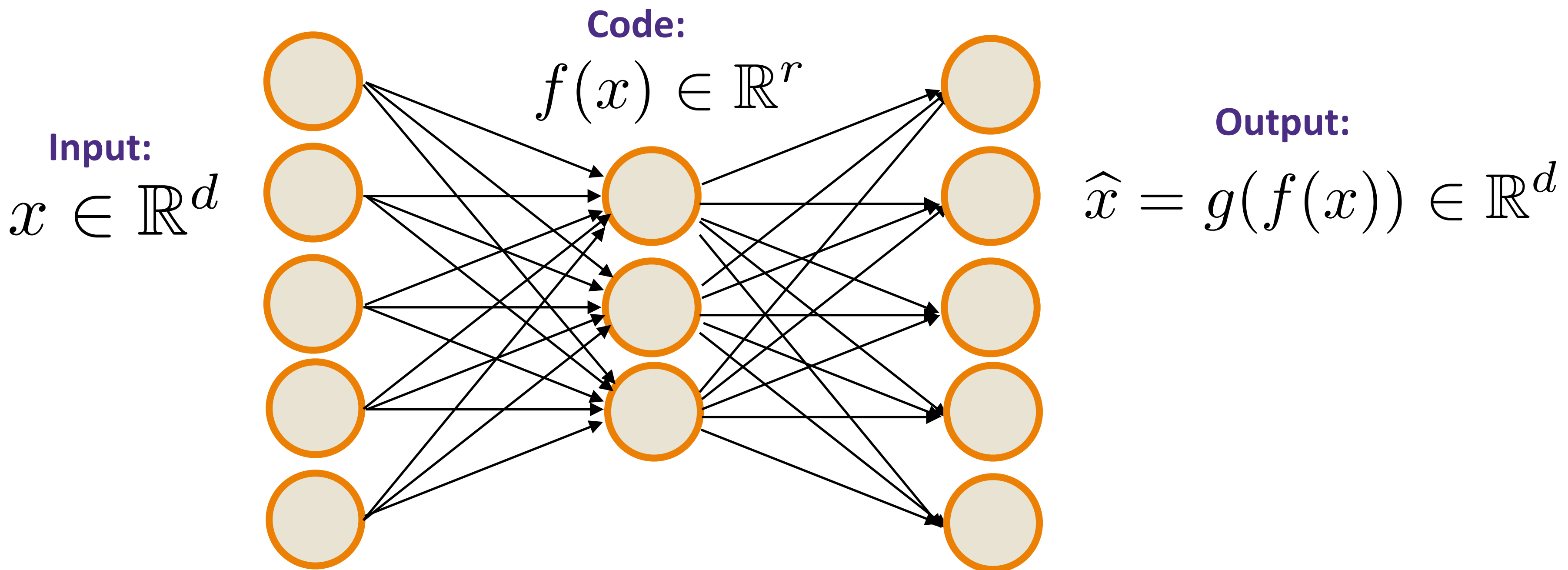


Output:

$\widehat{x} = g(f(x)) \in \mathbb{R}^d$

$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

Autoencoders



$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

What if $f(X) = Ax$ and $g(y) = By$?

Feature Extraction given categorical data

Feature extraction - categorical

Categorical data $x = [\texttt{Red}, 98105, \texttt{Finished basement}, , \ldots, 2.5]^\top$

Many machine learning algorithms (e.g., linear predictors) require **real valued-vectors** to make predictions.

And we want those real-valued numbers to be **correlated with the label**.

Feature extraction - categorical

Categorical data $x = [\texttt{Red}, 98105, \texttt{Finished basement}, , \dots, 2.5]^\top$

Many machine learning algorithms (e.g., linear predictors) require **real valued-vectors** to make predictions.

And we want those real-valued numbers to be **correlated with the label**.

One-hot encoding: Assign canonical vector to each categorical variable

$$\text{color} \in \{\texttt{red}, \texttt{green}, \texttt{blue}\}$$

Feature extraction - categorical

Categorical data $x = [\texttt{Red}, 98105, \texttt{Finished basement}, , \dots, 2.5]^\top$

Many machine learning algorithms (e.g., linear predictors) require **real valued-vectors** to make predictions.

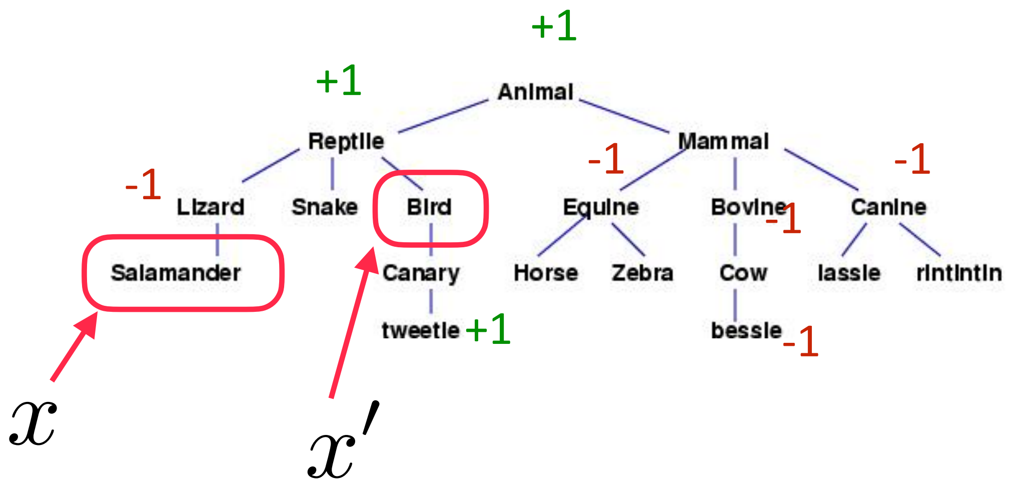
And we want those real-valued numbers to be **correlated with the label**.

Zip codes are also categorical. Is one-hot encoding appropriate?

$$\text{zip code} = 98105$$

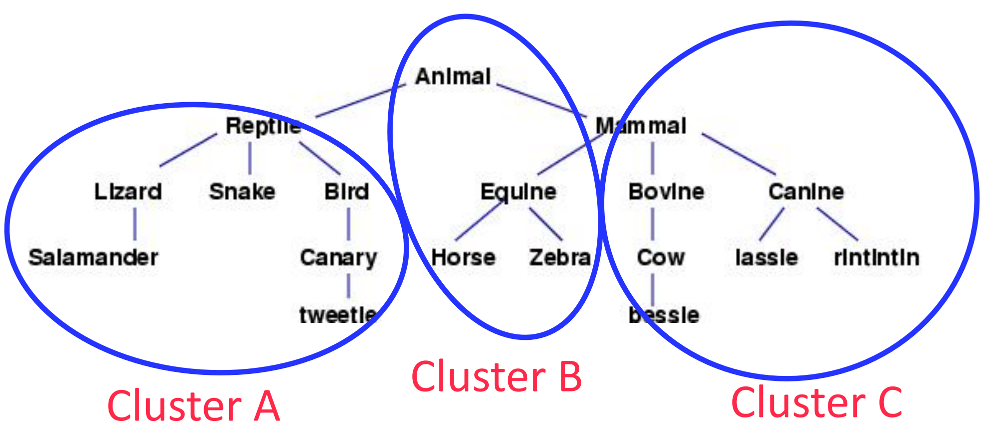
Feature extraction - structured

Structured data



Trees define a distance between any two nodes (length of path connecting them)

Given distances, you can assign each node to a cluster

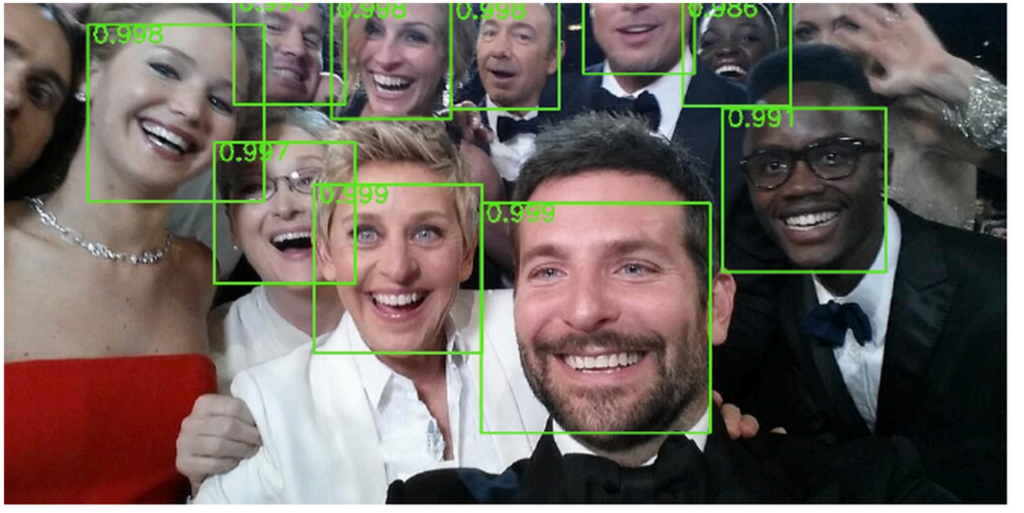


Then one-hot encode:

$$\text{cluster} \in \{A, B, C\}$$

Feature extraction given Image data

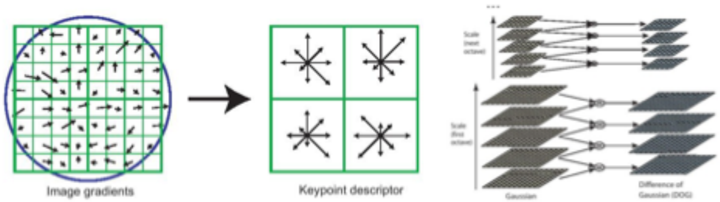
Computer Vision



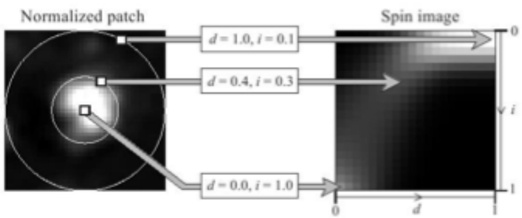
Find a feature vector for the image:

- Recognition
- Identification
- Detection
- Image classification
- etc…

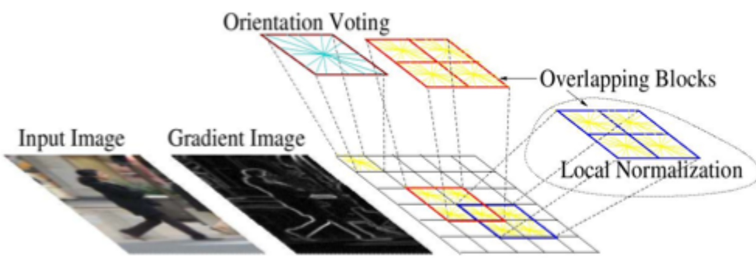
Some hand-created image features



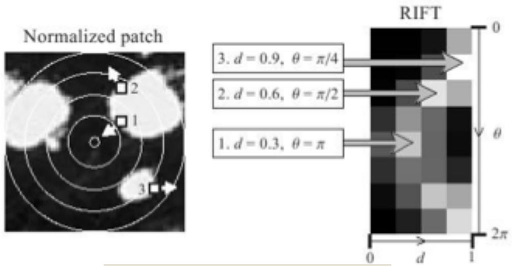
SIFT



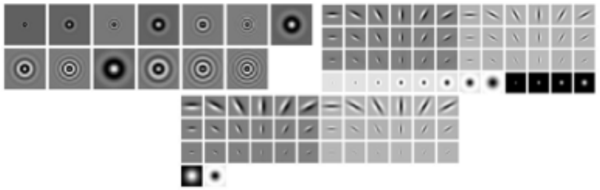
Spin Image



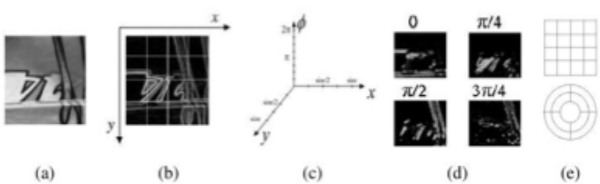
HoG



RIFT



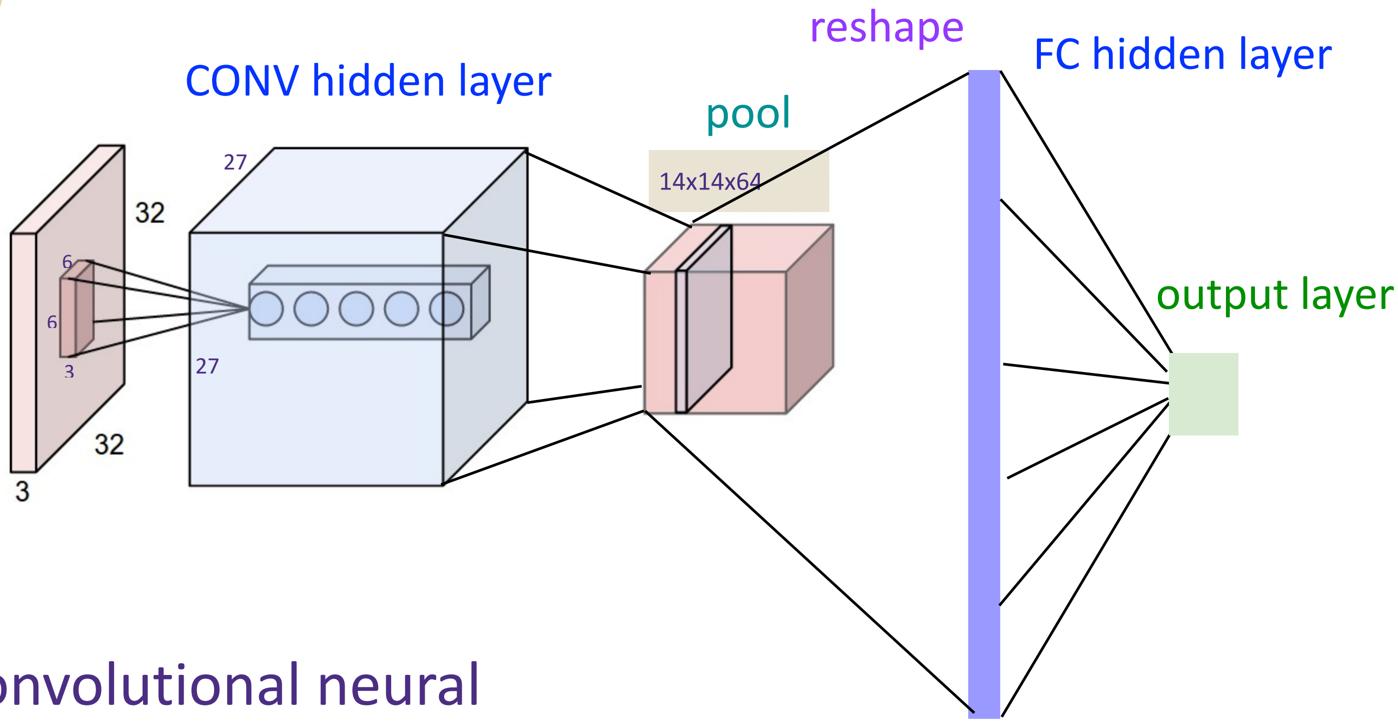
Texton



GLOH

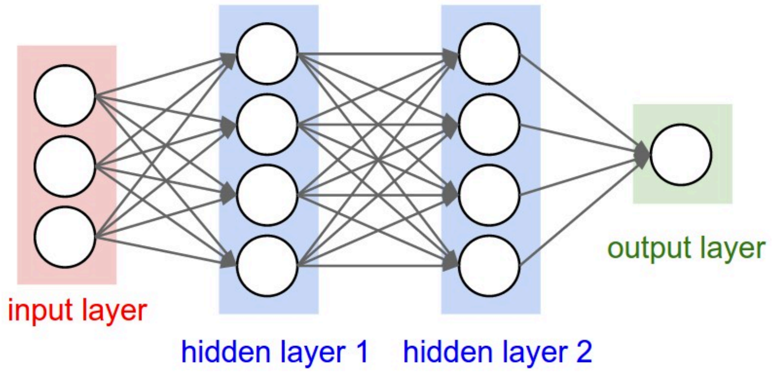
Slide from Honglak Lee

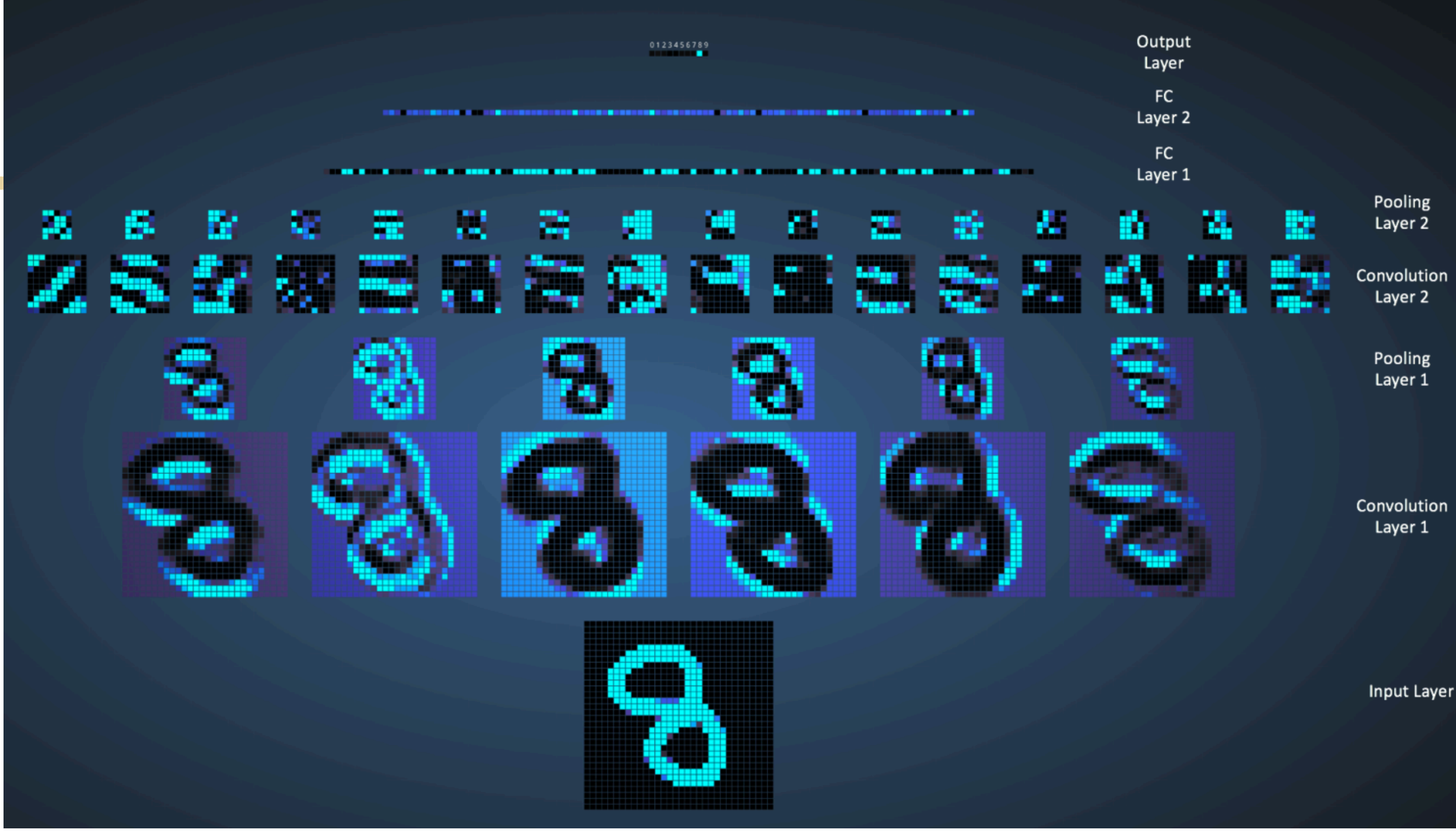
Learning Features with Convolutional Networks



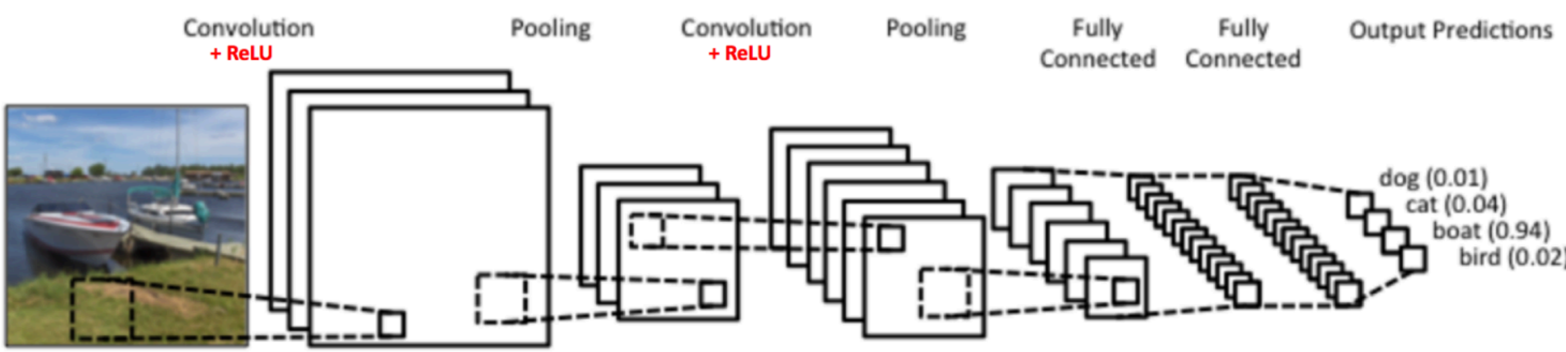
Recall: Convolutional neural networks (CNN) are just regular fully connected (FC) neural networks with some connections removed.

Train with SGD!



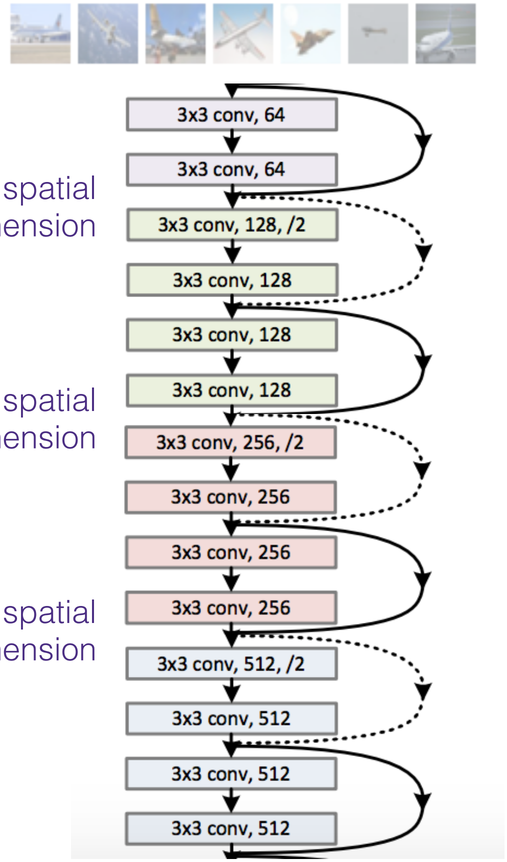


Real example network: LeNet



Real networks

Residual Network of [HeZhangRenSun'15]



Reduce spatial dimension

Reduce spatial dimension

Reduce spatial dimension

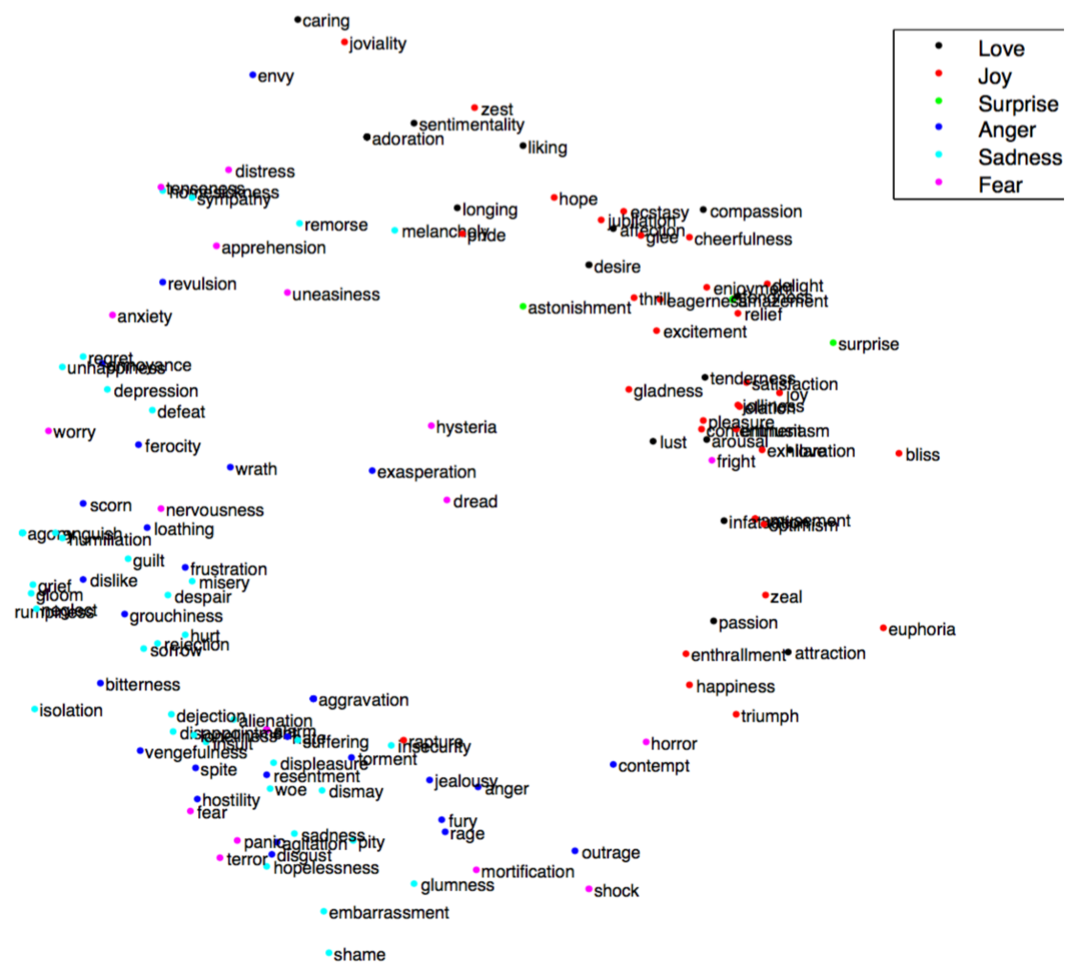
Feature extraction given Text data

Word embeddings, word2vec

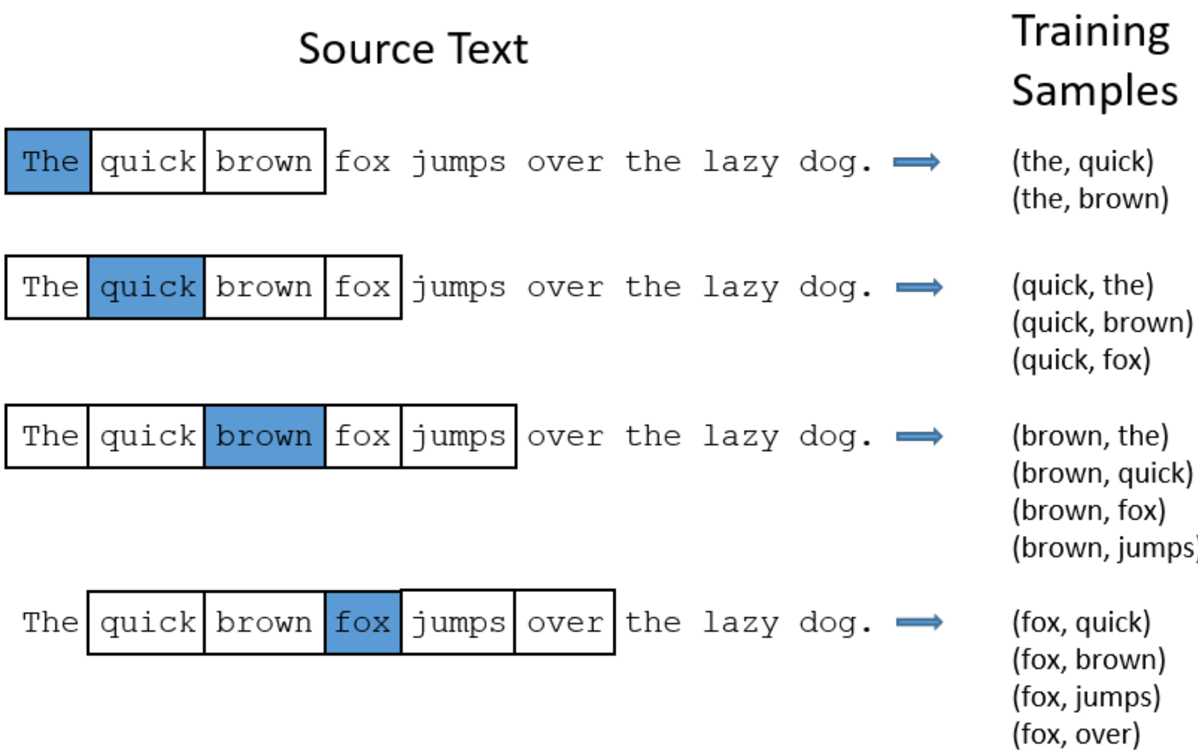
Can we **embed words** into a latent space?

This embedding came from directly querying for relationships.

word2vec is a popular unsupervised learning approach that just uses a text corpus (e.g. nytimes.com)

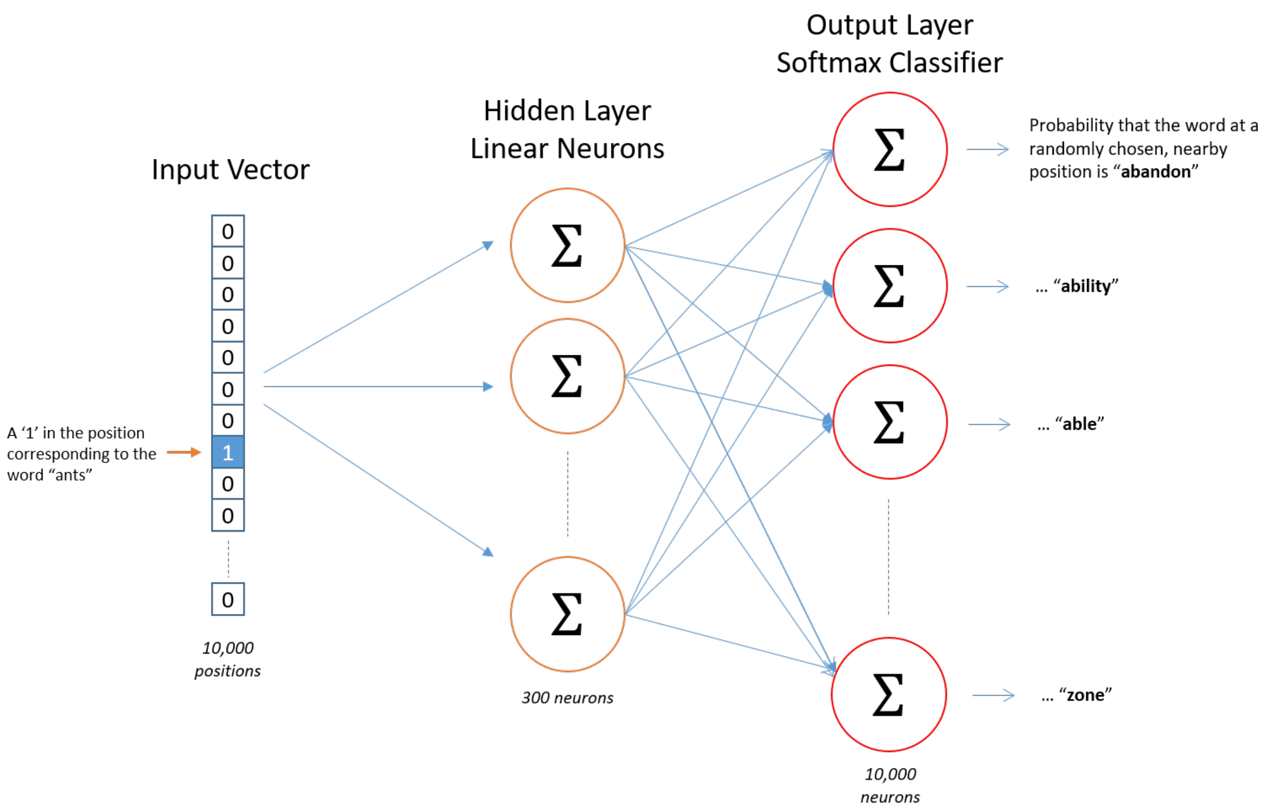


Word embeddings, word2vec



slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

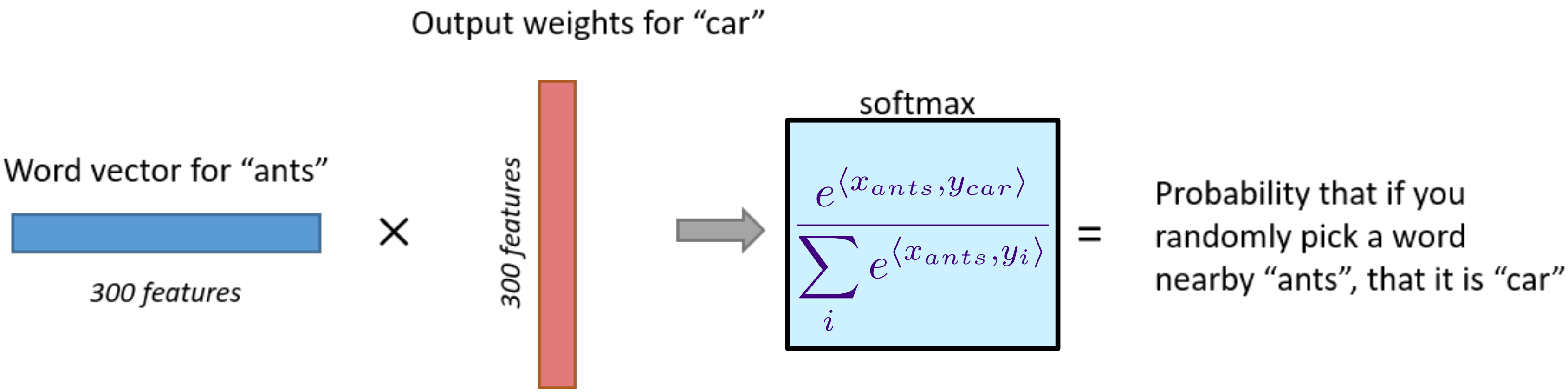
Word embeddings, word2vec



Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

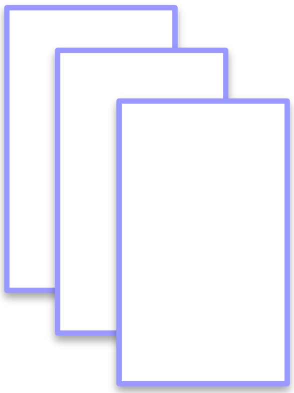
Word embeddings, word2vec



Training neural network to predict co-occuring words. Use first layer weights as embedding, throw out output layer

slide: http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

Bag of Words



n documents/articles with lots of text

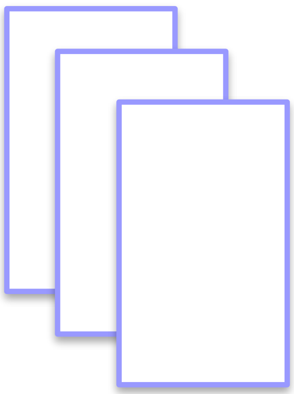
Questions:
- How to get a feature representation of each article?
- How to cluster documents into topics?

Bag of words model:

ith document: $x_i \in \mathbb{R}^D$

$$x_{i,j} = \text{ proportion of times } j\text{th word occurred in } i\text{th document}$$

Bag of Words



n documents/articles with lots of text

- **Can we embed each document into a feature space?**

Bag of words model:

i*th* document: $x_i \in \mathbb{R}^D$

$x_{i,j} =$ proportion of times jth word occurred in ith document

Given vectors, run k-means or Gaussian mixture model to find k clusters/topics

Nonnegative matrix factorization (NMF)

$A \in \mathbb{R}^{m \times n}$ $\quad A_{i,j} = \text{frequency of } j\text{th word in document } i$

Nonnegative Matrix factorization:

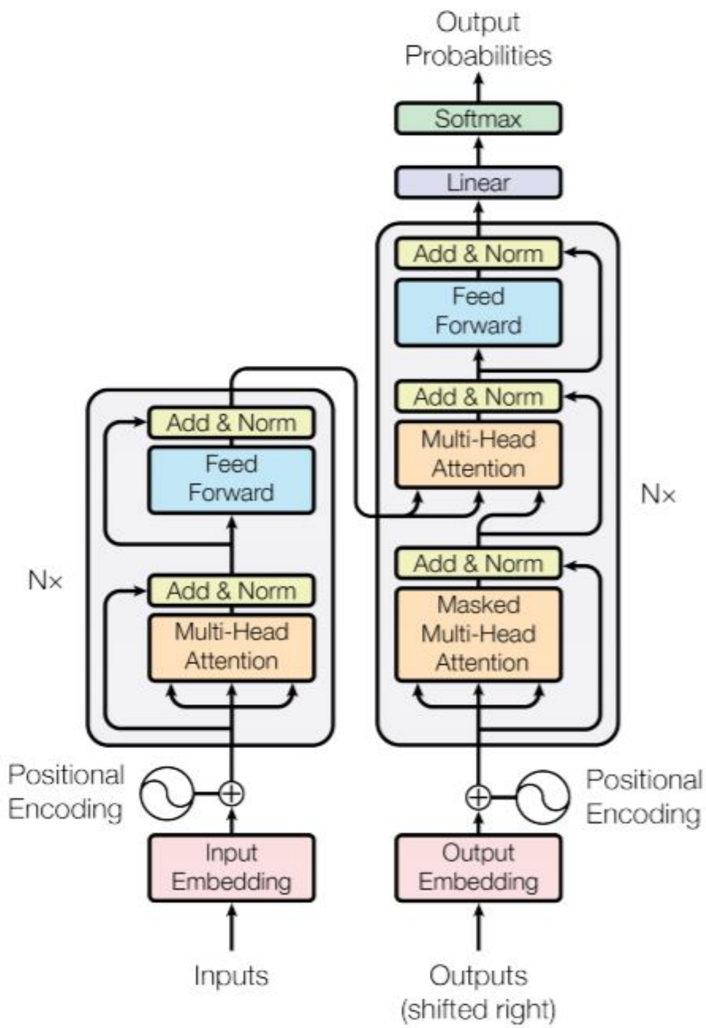
$$\min_{W \in \mathbb{R}_+^{m \times d}, H \in \mathbb{R}_+^{n \times d}} \|A - WH^T\|_F^2$$

d is number of topics

Each column of H represents a cluster of a topic,
Each row W is some weights a combination of topics

Also see latent Dirichlet factorization (LDA)

BERT



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Feature extraction given sequential data

Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

Prediction model: $p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots)$

Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

$h_t \in \mathbb{R}^d$: hidden latent state of AAPL

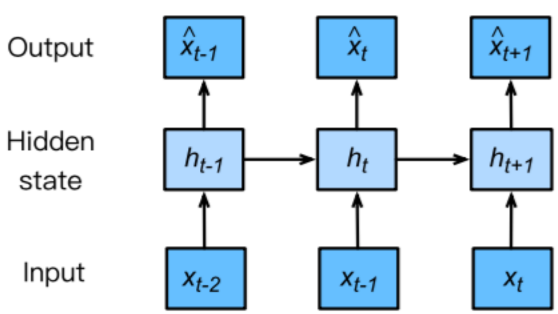
$$\text{Prediction model: } p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \ldots) \approx p(x_{t+1} | x_t, h_{t+1})$$

Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

$h_t \in \mathbb{R}^d$: hidden latent state of AAPL

$$\text{Prediction model: } p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots) \approx p(x_{t+1} | x_t, h_{t+1})$$



$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Zhang et al. "Dive into Deep Learning"

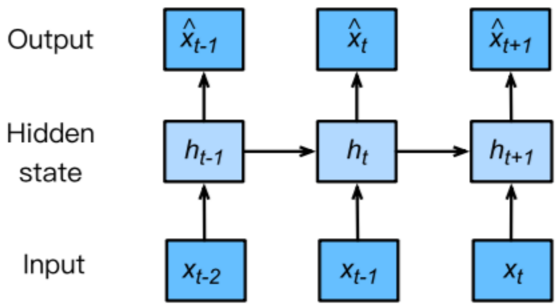
Time-dependent data

$x_t \in \mathbb{R}$: AAPL stock price at time t

$h_t \in \mathbb{R}^d$: hidden latent state of AAPL

Prediction model: $p(x_{t+1} | x_t, x_{t-1}, x_{t-2}, \dots)$

$$\approx p(x_{t+1} | x_t, h_{t+1})$$



$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Explicit:

$$h_{t+1} = \sigma(Ah_t + Bx_t)$$
$$\widehat{x}_{t+1} = Ch_{t+1} + Dx_t$$

$$\sum_t (x_t - \widehat{x}_t)^2$$

Zhang et al. "Dive into Deep Learning"

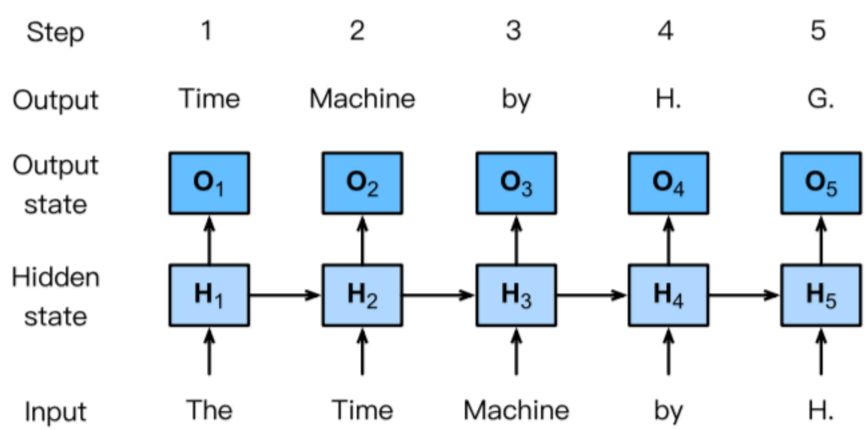
Time-dependent data

$$\text{Prediction model: } p(x_{t+1}|x_t, x_{t-1}, x_{t-2}, \dots)$$
$$\approx p(x_{t+1}|x_t, h_{t+1})$$

$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Model also works with text!



Zhang et al. "Dive into Deep Learning"

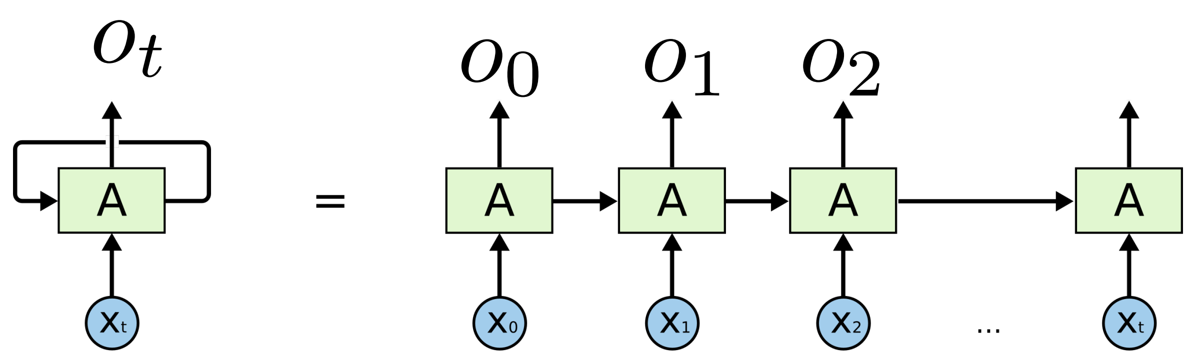
Time-dependent data

$$\text{Prediction model: } p(x_{t+1}|x_t, x_{t-1}, x_{t-2}, \dots) \approx p(x_{t+1}|x_t, h_{t+1})$$

$$h_{t+1} = g(h_t, x_t)$$

Hidden state and g never observed, but learned!

Recurrent Neural Network



Slide: http://colah.github.io/posts/2015-08-Understanding-LSTMs/