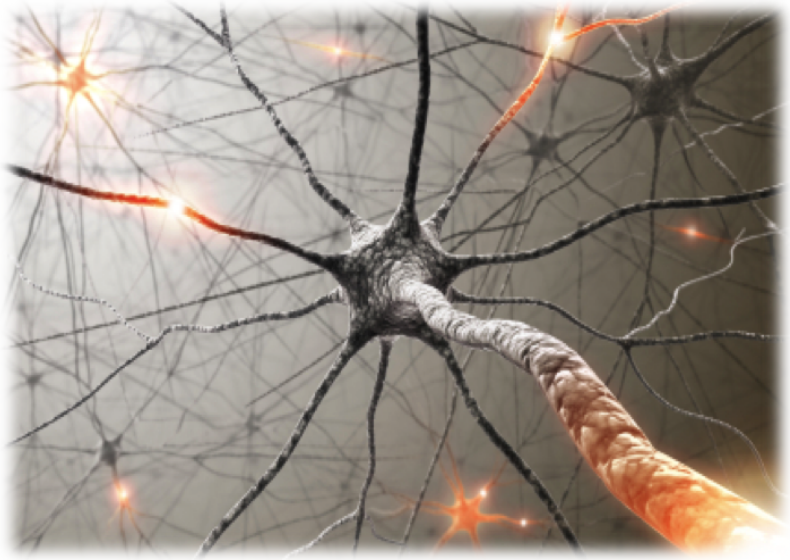
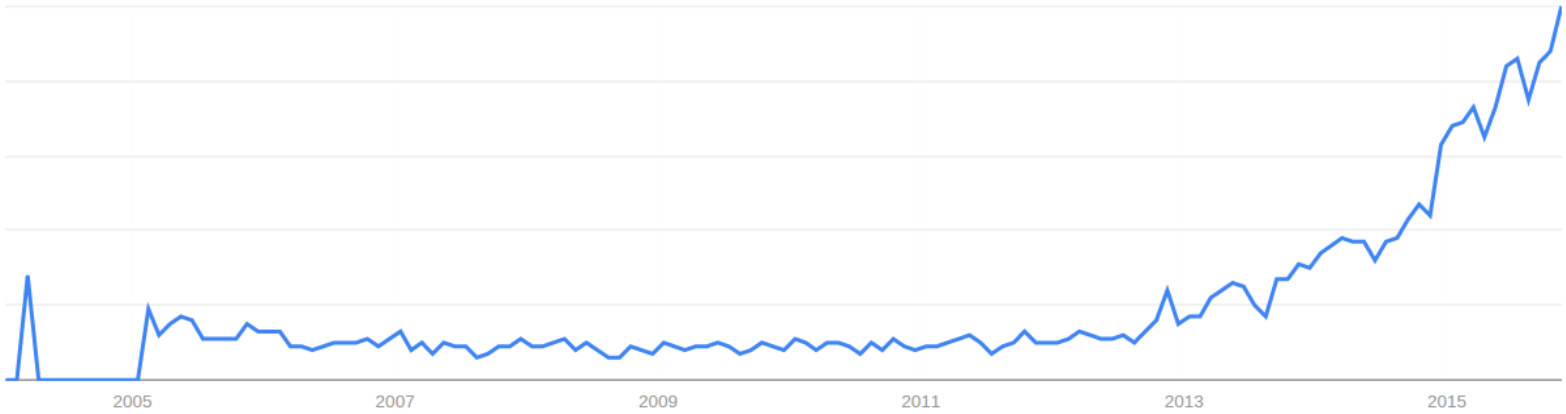
Lecture 19: Deep Learning



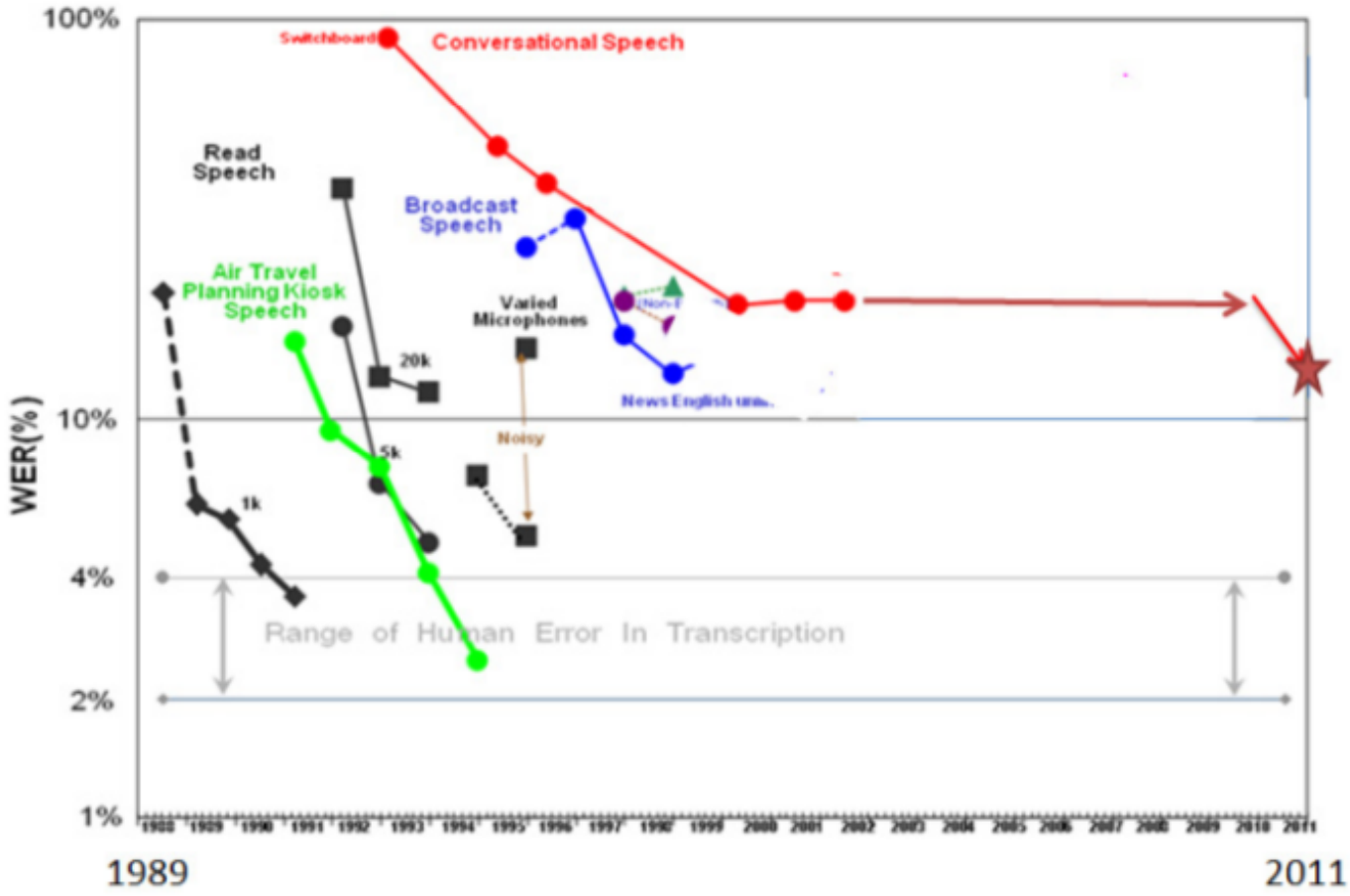
Google Trends

Query: deep learning



[figure from Li Deng]

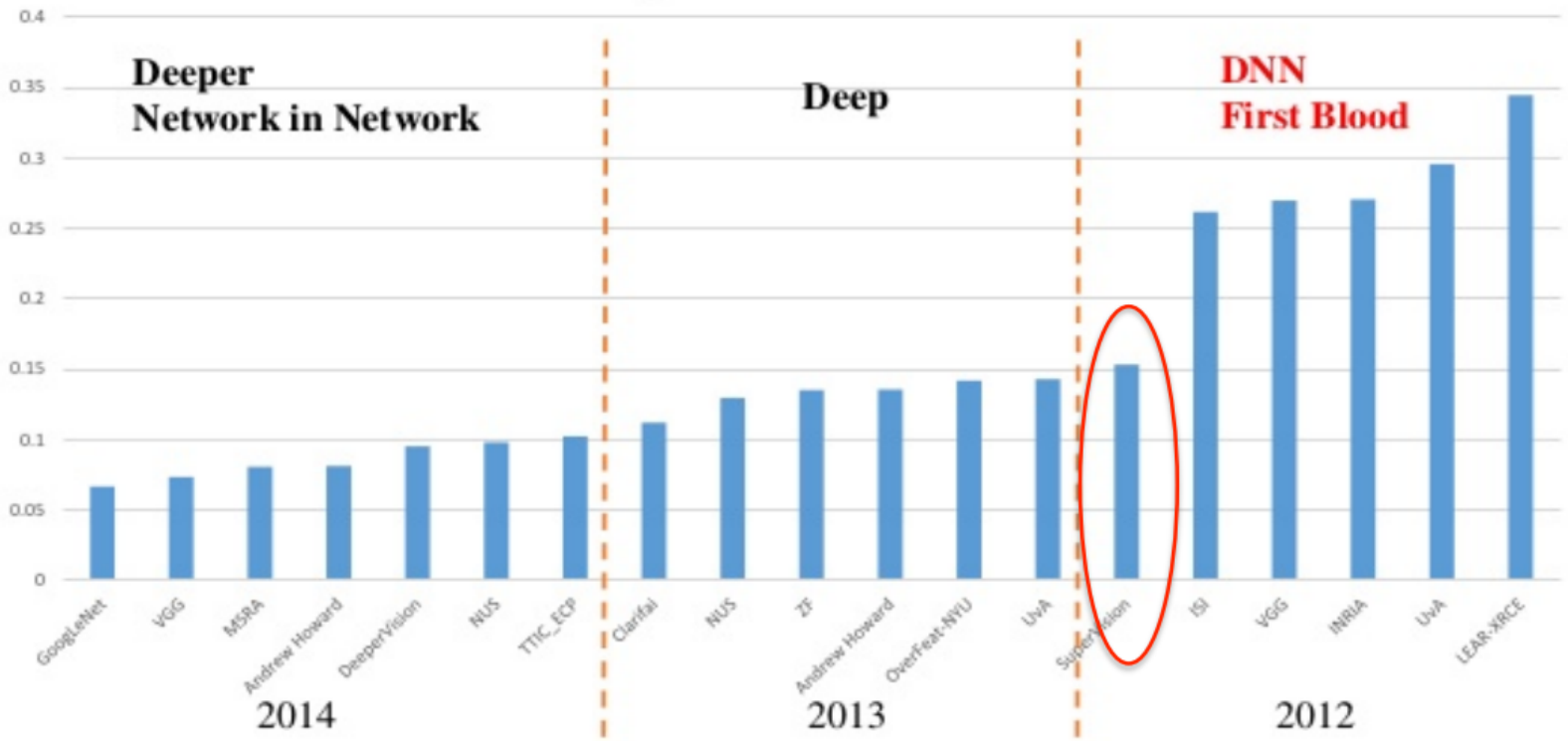
Speech recognition (2009-2011)



- Steep drop in WER due to deep learning
- IBM, Google, Microsoft all switched over from GMM-HMM

[Krizhevsky et al., 2012, a.k.a. AlexNet]

Object recognition (2012)

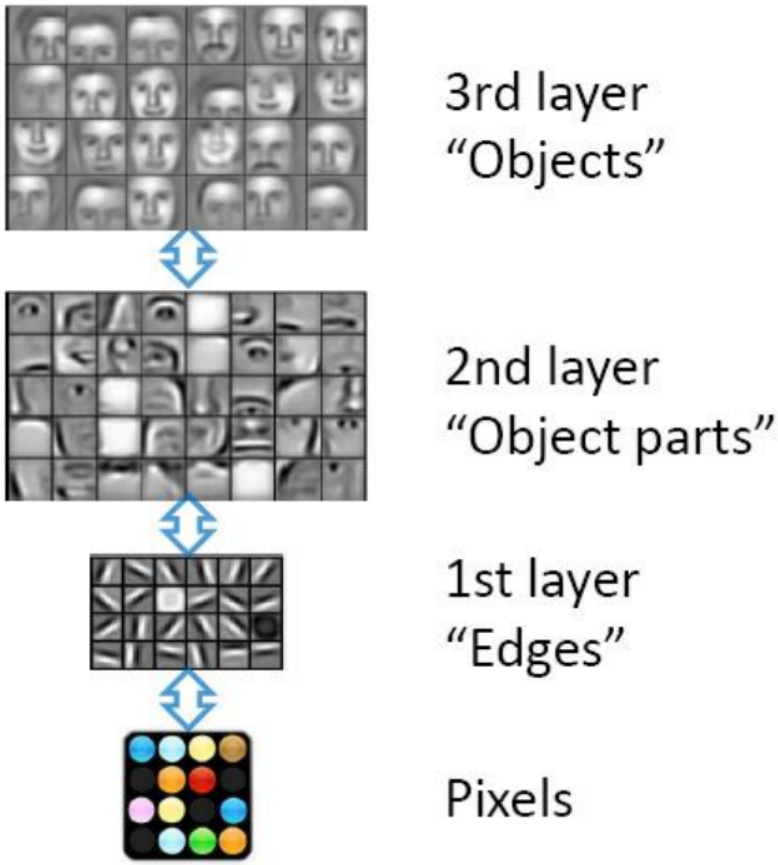


- Landslide win in ImageNet competition
- Computer vision community switched to CNNs

[figure from Honglak Lee]

What is deep learning?

Philosophy: learn high-level abstractions automatically



A brief history

- 1950-60s: modeling brain using neural networks (Rosenblatt, Hebb, etc.)
- 1969: research stagnated after Minsky and Papert's paper
- 1986: popularization of backpropagation by Rumelhardt, Hinton, Williams
- 1990s: convolutional neural networks (LeCun)
- 1990s: recurrent neural networks (Schmidhuber)
- 2006: revival of deep networks (Hinton et al.)
- 2013-: massive industrial interest

Key problem: was difficult to get training multi-layer neural networks to work!

What's different today

Computation (time/memory)

Information (data)

Deep learning is fundamentally empirical

Roadmap

Supervised learning

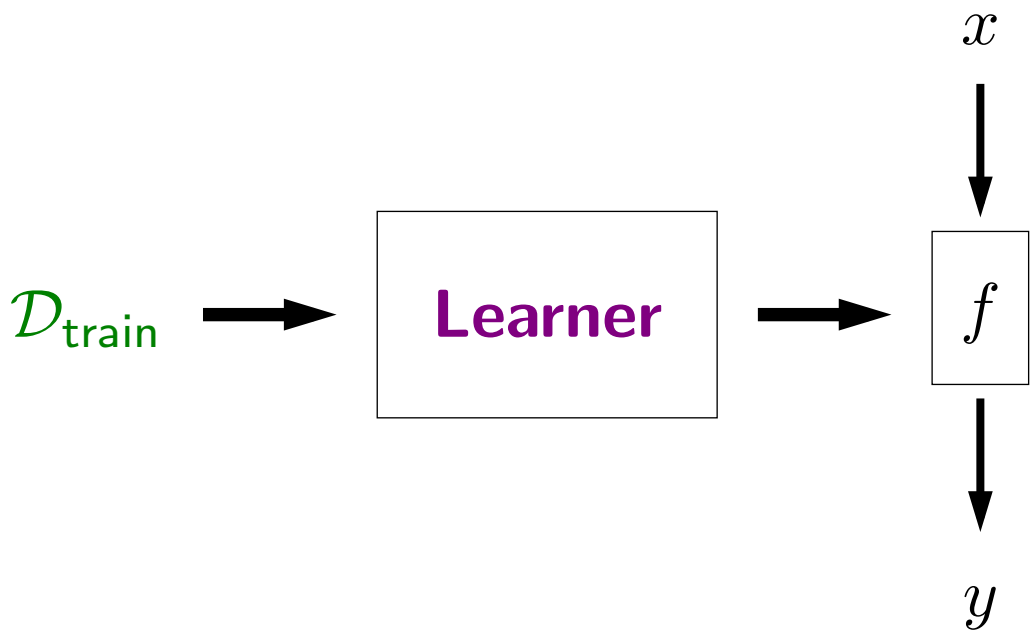
Unsupervised learning

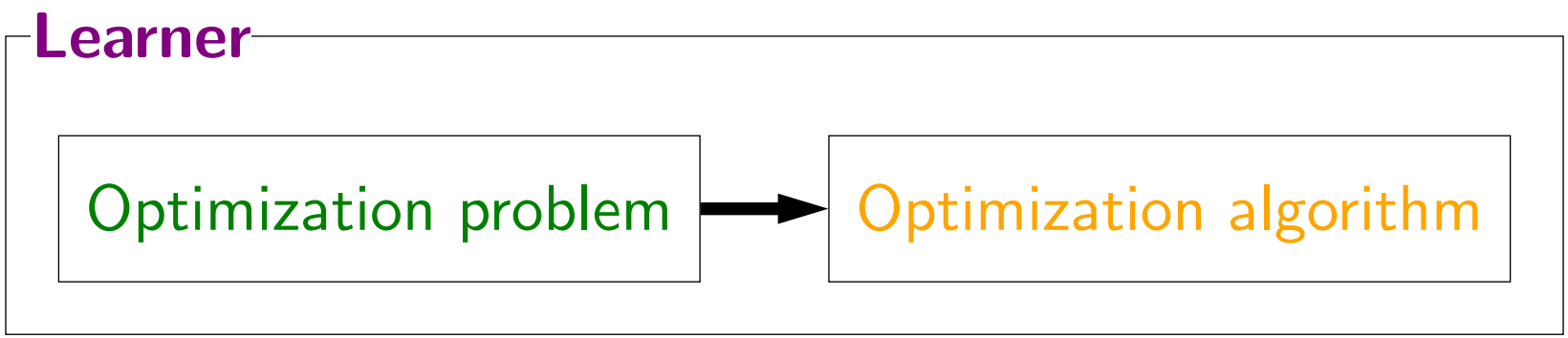
Convolutional neural networks

Recurrent neural networks

Final remarks

Framework


x
$\mathcal{D}_{\text{train}}$
Learner
f
y


Learner
Optimization problem
Optimization algorithm

Review: optimization

Regression:

$$\text{Loss}(x, y, \theta) = (f_\theta(x) - y)^2$$

Key idea: minimize training loss

$$\text{TrainLoss}(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \theta)$$

$$\min_{\theta \in \mathbb{R}^d} \text{TrainLoss}(\theta)$$

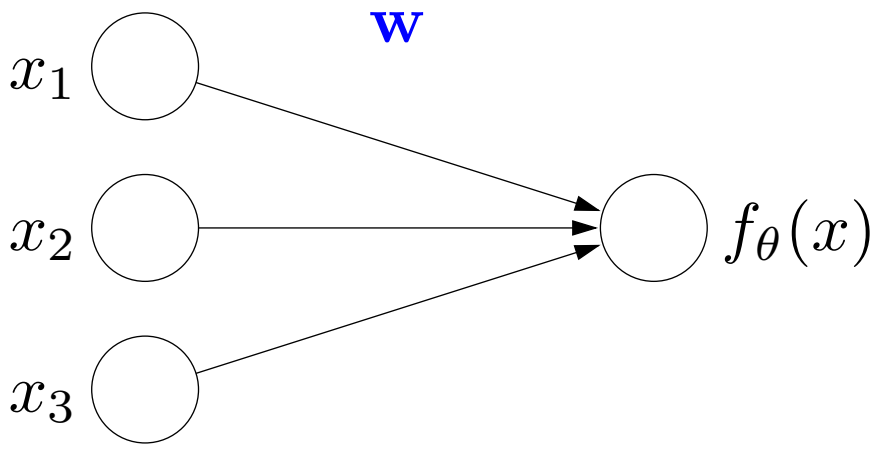
Algorithm: stochastic gradient descent

For $t = 1, \ldots, T$:

  For $(x, y) \in \mathcal{D}_{\text{train}}$:

    $\theta \leftarrow \theta - \eta_t \nabla_\theta \text{Loss}(x, y, \theta)$

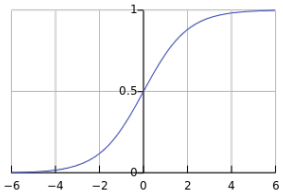
Review: linear predictors



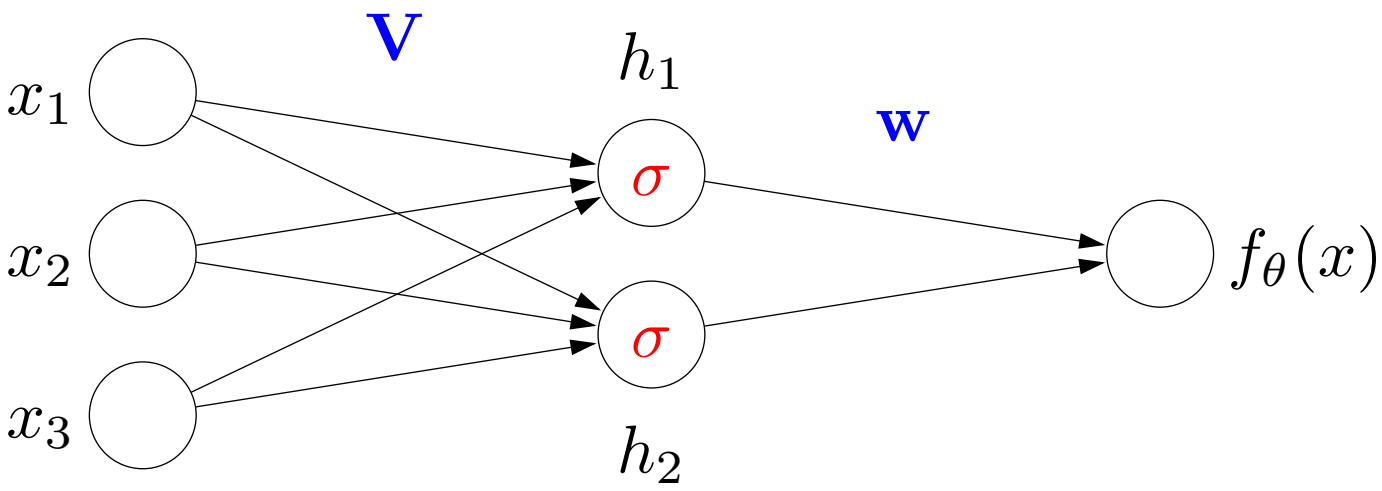
Output:

$$f_\theta(x) = \mathbf{w} \cdot x$$

Parameters: $\theta = \mathbf{w}$



Review: neural networks



Intermediate hidden units:

$$h_j(x) = \sigma(\mathbf{v}_j \cdot x) \quad \sigma(z) = (1 + e^{-z})^{-1}$$

Output:

$$f_\theta(x) = \mathbf{w} \cdot \mathbf{h}(x)$$

Parameters: $\theta = (\mathbf{V}, \mathbf{w})$

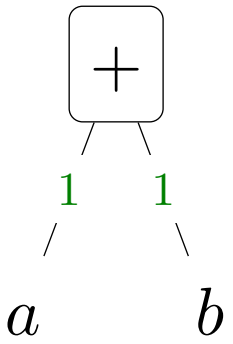
Summary so far

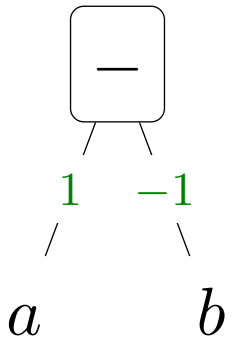
Neural network predictor: $f_\theta(x) = \mathbf{w} \cdot \sigma(\mathbf{V}x)$

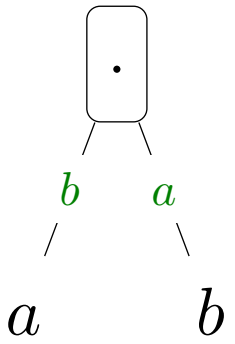
Squared loss: $\text{Loss}(x, y, \theta) = (f_\theta(x) - y)^2$

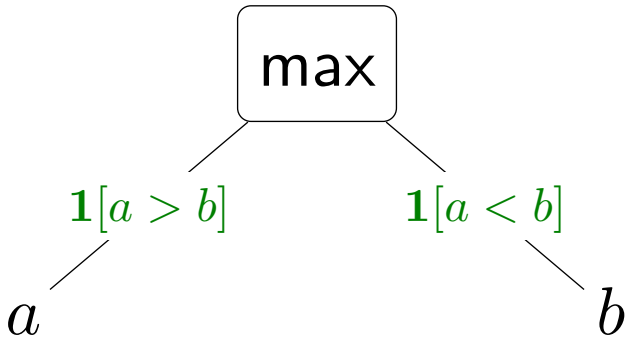
Next step: compute the gradient $\nabla_\theta \text{Loss}(x, y, \theta)$

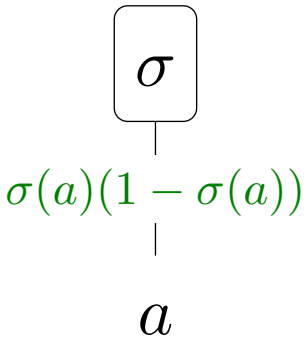
Basic building blocks



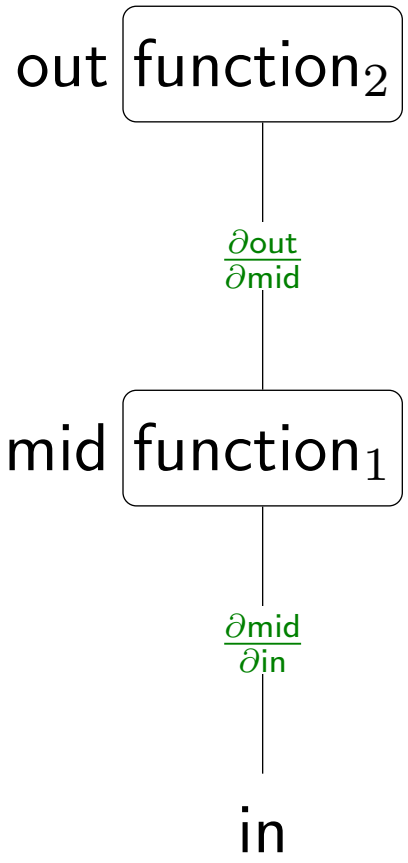








Composing functions



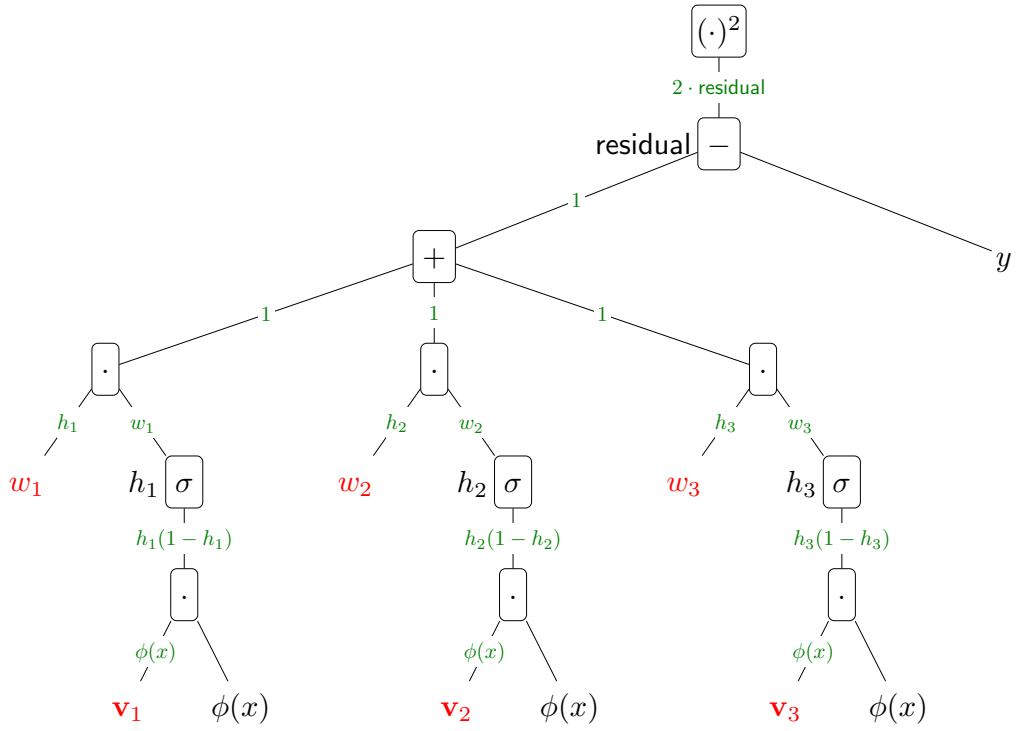
Chain rule:

$$\frac{\partial \text{out}}{\partial \text{in}} = \frac{\partial \text{out}}{\partial \text{mid}} \frac{\partial \text{mid}}{\partial \text{in}}$$

Computing the gradient

$$\text{Loss}(x, y, \mathbf{w}) = \left(\sum_{j=1}^{k} w_j \sigma(\mathbf{v}_j \cdot \phi(x)) - y \right)^2$$

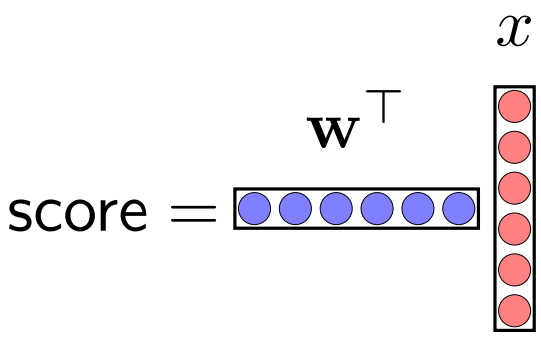
Assume labels $\{1, 2, 3\}$ and correct label is $y = 1$



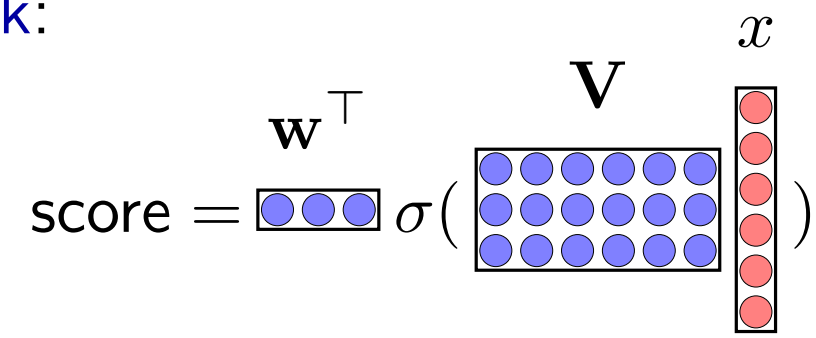
[Andrej Karpathy's demo]

Deep neural networks

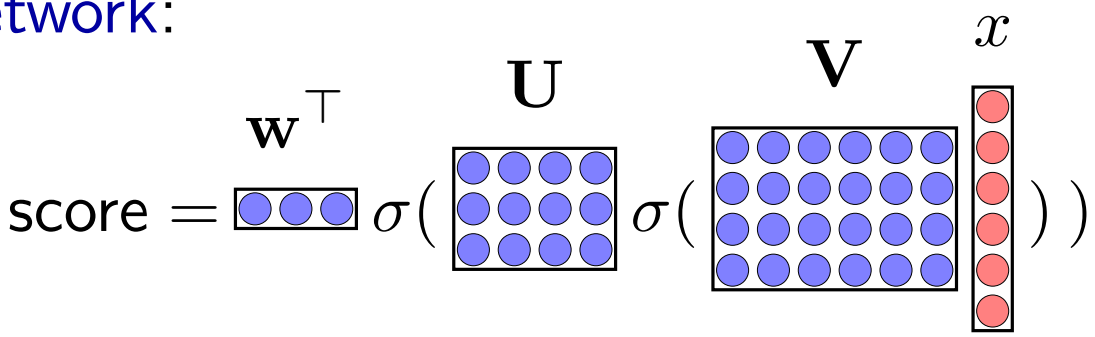
1-layer neural network:



2-layer neural network:

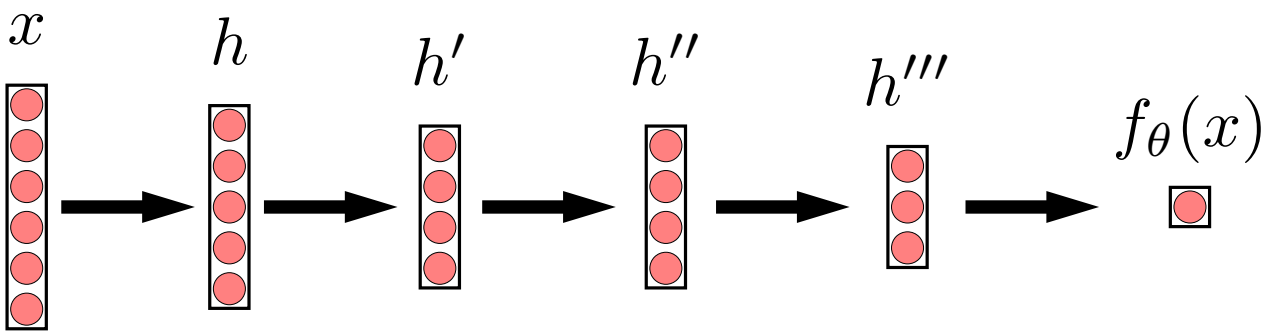


3-layer neural network:



...

Depth



Intuitions:

- Hierarchical feature representations
- Can simulate a bounded computation logic circuit (original motivation from McCulloch/Pitts, 1943)
- Learn this computation (and potentially more because networks are real-valued)
- Depth $k+1$ logic circuits can represent more than depth k (counting argument)
- Formal theory/understanding is still incomplete

Supervised learning

- Construct deep neural networks by composing non-linearities (σ) and linear transformations (matrix multiplication)

- Train via SGD, use backpragation to compute gradients

- Non-convex optimization, but works empirically given enough compute and data

Roadmap

Supervised learning

Unsupervised learning

Convolutional neural networks

Recurrent neural networks

Final remarks

Motivation

- Deep neural networks requires lots of data

- Sometimes not very much labeled data, but plenty of unlabeled data (text, images, videos)

- Humans rarely get direct supervision; can learn from raw sensory information?

Autoencoders

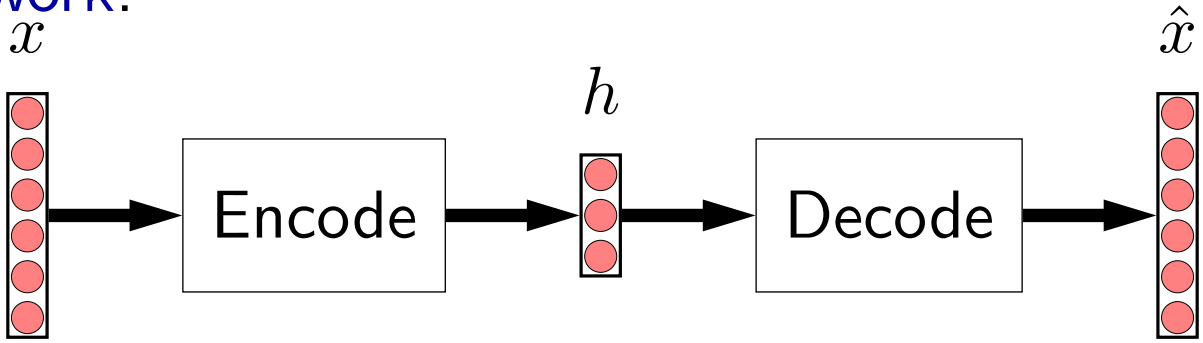
Analogy:

A A A A B B B B B → 4 A's, 5 B's → A A A A B B B B B

Key idea: autoencoders

If can compress a data point and still reconstruct it, then we have learned something generally useful.

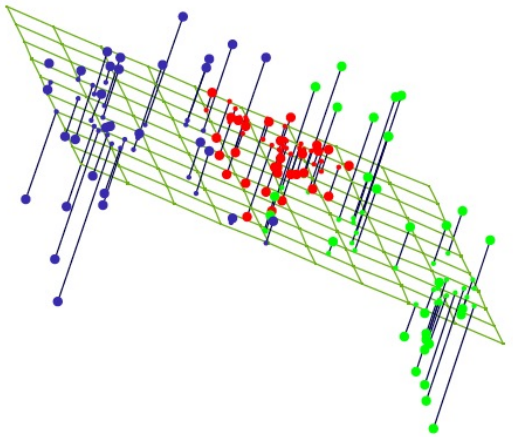
General framework:



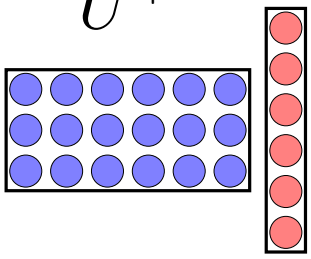
minimize $\|x - \hat{x}\|^2$

Principal component analysis

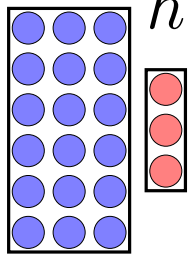
Input: points $x_1, \ldots, x_n$



$$\text{Encode}(x) = U^\top x$$



$$\text{Decode}(h) = Uh$$



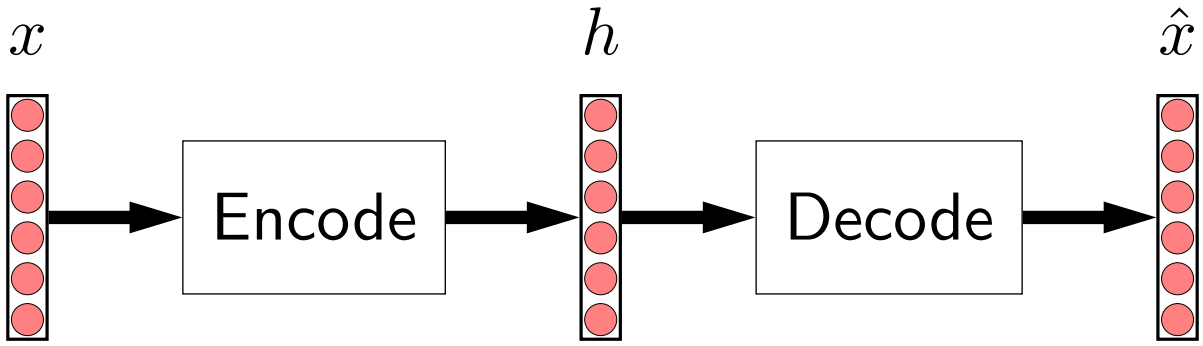
(assume x_i's are mean zero and U is orthogonal)

PCA objective:

$$\text{minimize} \sum_{i=1}^{n} \|x_i - \text{Decode}(\text{Encode}(x_i))\|^2$$

Autoencoders

Increase dimensionality of hidden dimension:

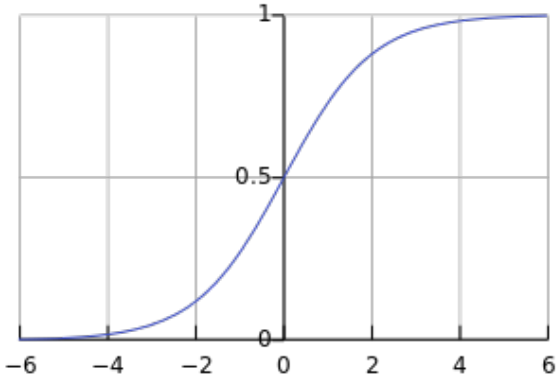


- Problem: learning nothing — just set Encode, Decode to identity function!

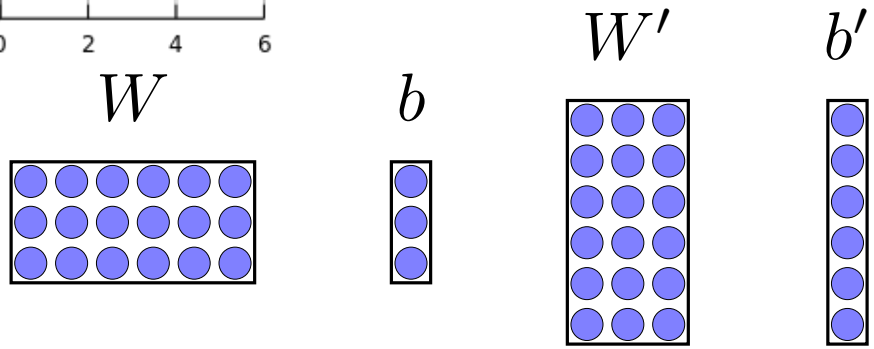
- Need to control complexity of Encode and Decode somehow...

Non-linear autoencoders

Non-linear transformation (e.g., logistic function):



$$\text{Encode}(x) = \sigma(Wx + b)$$
$$\text{Decode}(h) = \sigma(W'h + b')$$

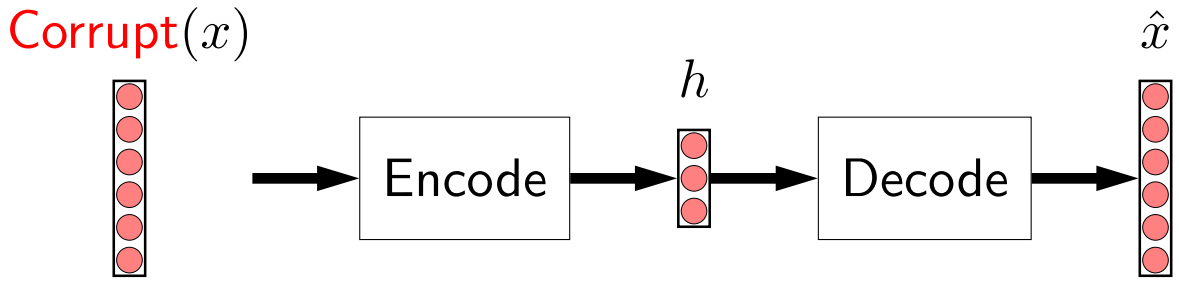


Loss function:

$$\text{minimize } \|x - \text{Decode}(\text{Encode}(x))\|^2$$

Key: non-linearity makes life harder, prevents degeneracy

Denoising autoencoders



Types of noise:

- Blankout: Corrupt$([1,2,3,4]) = [0,2,3,0]$
- Gaussian: Corrupt$([1,2,3,4]) = [1.1, 1.9, 3.3, 4.2]$

Objective:

$$\text{minimize } \|x - \text{Decode}(\text{Encode}(\text{Corrupt}(x)))\|^2$$

Algorithm: pick example, add fresh noise, SGD update

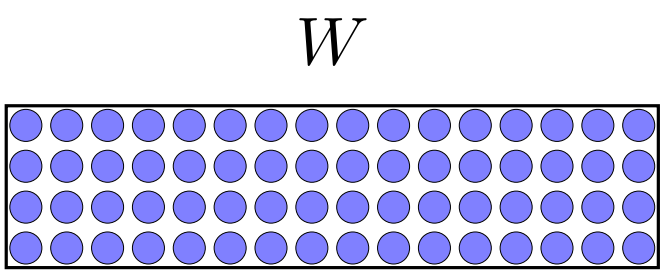
Key: noise makes life harder, prevents degeneracy

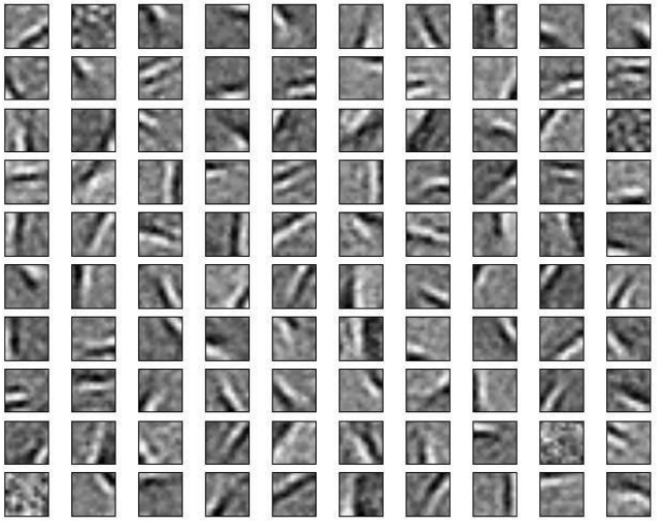
[Figure 7 of Vincent et al. (2010)]

Denoising autoencoders

MNIST: 60,000 images of digits (784 dimensions)

200 learned filters (rows of W):

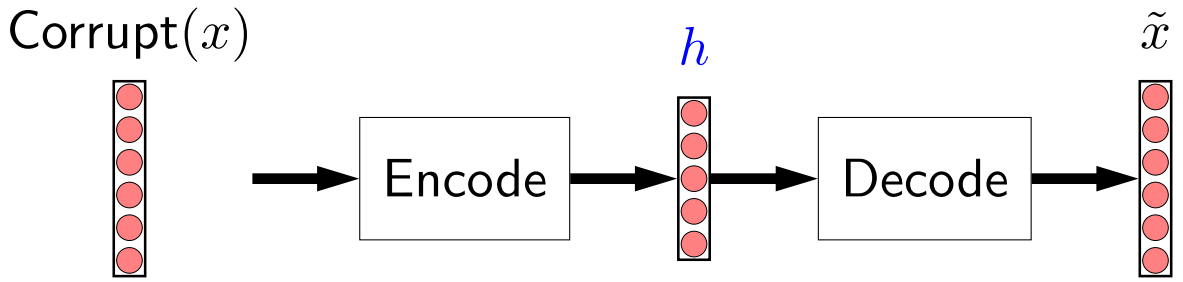




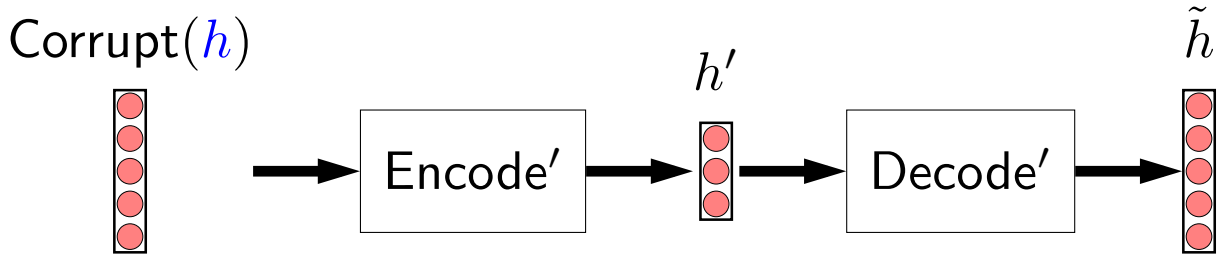
Stacked denoising autoencoders

Goal: learn hierarchical features

Train first layer:



Train second layer:



...

Test time: Encode′(Encode(x))

Probabilistic models

So far:

$$\text{Decode}(\text{Encode}(x))$$

Probabilistic model: distribution over inputs and hidden states

$$p(x, h)$$

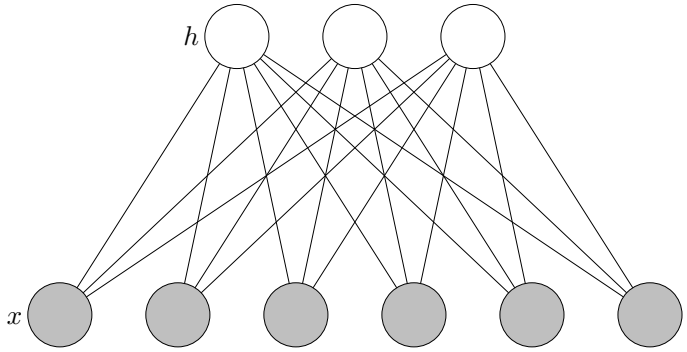
Two types:

- Restricted Boltzmann machines (Markov network)
- Deep belief network (Bayesian network)

For simplicity, assume x and h are binary vectors

Restricted Boltzmann machines

Markov network (factor graph):



Sampling: $h \mid x$ is easy, x is hard

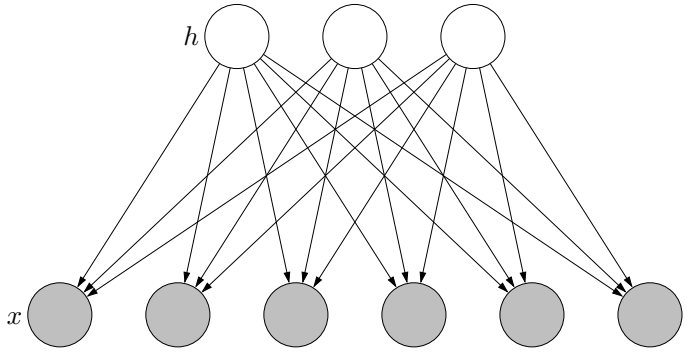
$$p(x,h) \propto \exp(\quad + b^\top h + c^\top x)$$

Learning: SGD; gradient requires summing over all (x, h)

Contrastive divergence: initialize x, 1 step of Gibbs sampling

Deep belief networks

Bayesian network:



Sampling: $h \mid x$ is hard, x is easy

$$p(x|h) \propto \exp(\; h^\top \; W \; x \; + b^\top h + c^\top x)$$

Learning: maximum likelihood is intractable, so use same algorithm as RBM; repeat to get deep (like for stacked denoising autoencoders)

Distributional semantics: warmup

The new design has _____ lines.

Let's try to keep the kitchen _____.

I forgot to _____ out the cabinet.

What does _____ mean?

Distributional semantics

The new design has _____ lines.

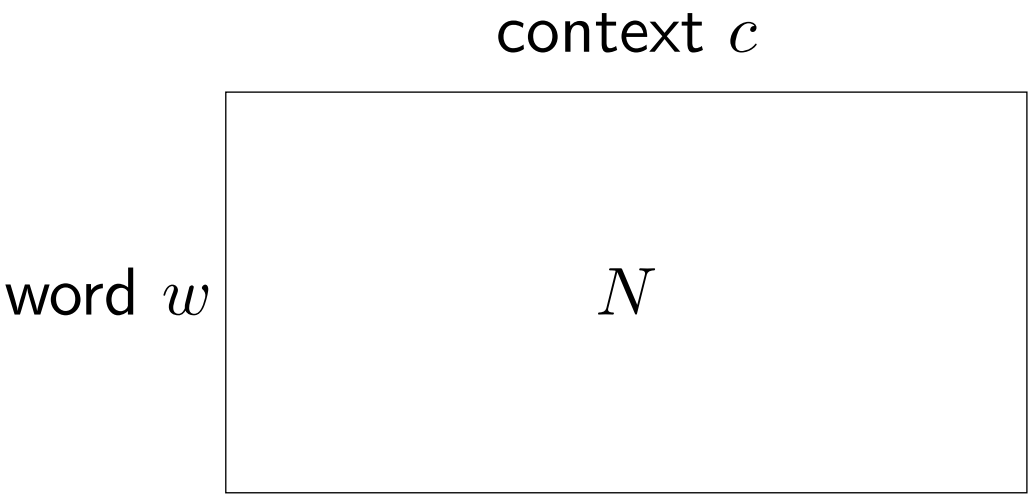
Observation: **context** can tell us a lot about word meaning

Autoencoding: predict x from x

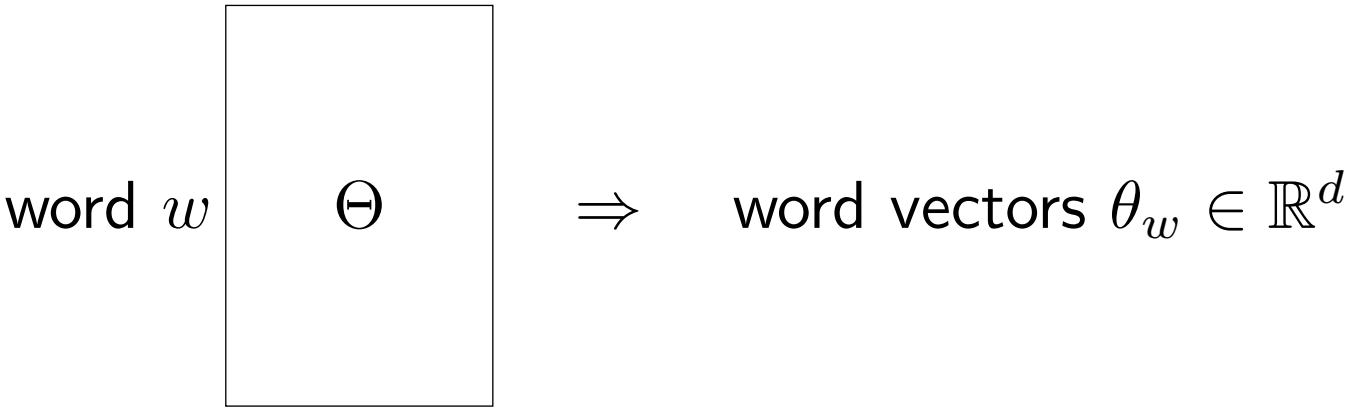
Distributional methods: predict x from context

General recipe

1. Form a **word-context matrix** of counts (data)



2. Perform **dimensionality reduction** (generalize)



[Deerwater/Dumais/Furnas/Landauer/Harshman, 1990]

Latent semantic analysis

Data:

Doc1: *Cats have tails.*

Doc2: *Dogs have tails.*

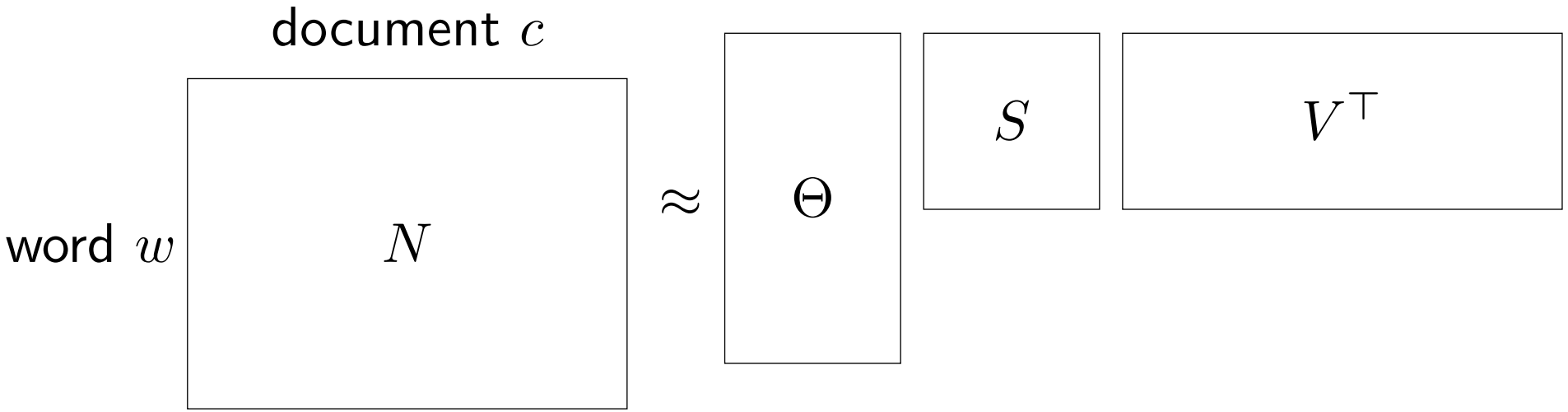
Matrix: context = **documents** that word appear in

	Doc1	Doc2
cats	1	0
dogs	0	1
have	1	1
tails	1	1

[Deerwater/Dumais/Furnas/Landauer/Harshman, 1990]

Latent semantic analysis

Dimensionality reduction: **SVD**



- Used for information retrieval
- Match query to documents in latent space rather than on keywords

[Mikolov/Sutskever/Chen/Corrado/Dean, 2013 (word2vec)]

Skip-gram model with negative sampling

Data:

Cats and dogs have tails.

Matrix: context = words in a window

	cats	and	dogs	have	tails
cats	0	1	1	0	0
and	1	0	1	1	0
dogs	1	1	0	1	1
have	0	1	1	0	1
tails	0	0	1	1	0

[Mikolov/Sutskever/Chen/Corrado/Dean, 2013 (word2vec)]

Skip-gram model with negative sampling

Cats are smarter than the best AI.

Dimensionality reduction: **logistic regression with SGD**

Model: predict good (w, c) using logistic regression

$$p_\theta(g = 1 \mid w, c) = (1 + \exp(\theta_w \cdot \beta_c))^{-1}$$

Positives: (w, c) from data

Negatives: (w, c') for irrelevant c' (k times more)

$+$(cats, AI) $-$(cats, linguistics) $-$(cats, statistics)

[Mikolov/Sutskever/Chen/Corrado/Dean, 2013 (word2vec)]

Skip-gram model with negative sampling

Data distribution:

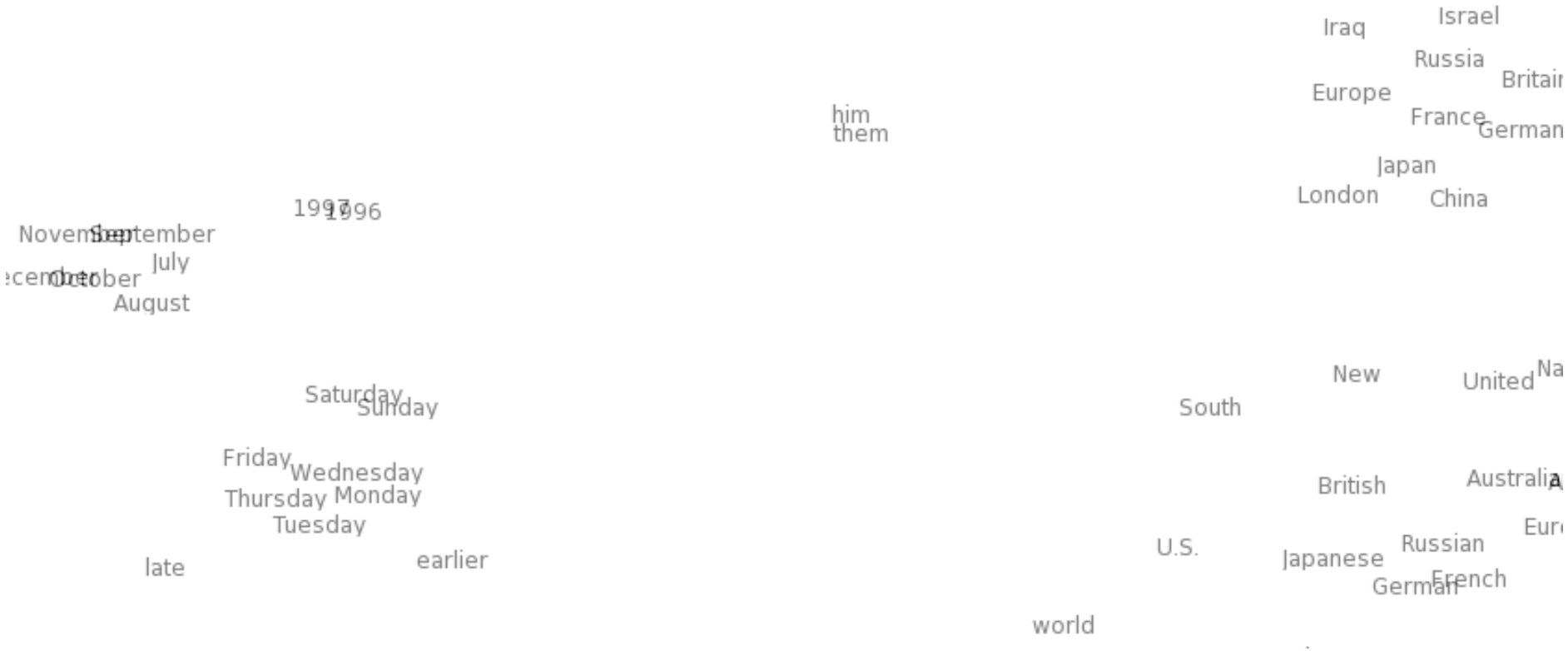
$$\hat{p}(w, c) \propto N(w, c)$$

Objective:

$$\max_{\theta,\beta} \sum_{w,c} \hat{p}(w, c) \log p(g = 1 \mid w, c) +$$

$$k \sum_{w,c'} \hat{p}(w)\hat{p}(c') \log p(g = 0 \mid w, c')$$

2D visualization of word vectors



[Mikolov/Yih/Zweig, 2013; Levy/Goldberg, 2014]

Analogies

Differences in context vectors capture relations:

$$\theta_{\text{king}} - \theta_{\text{man}} \approx \theta_{\text{queen}} - \theta_{\text{woman}} \text{ (gender)}$$

$$\theta_{\text{france}} - \theta_{\text{french}} \approx \theta_{\text{mexico}} - \theta_{\text{spanish}} \text{ (language)}$$

$$\theta_{\text{car}} - \theta_{\text{cars}} \approx \theta_{\text{apple}} - \theta_{\text{apples}} \text{ (plural)}$$

Intuition:

$$\underbrace{\theta_{\text{king}}}_{\text{[crown,he]}} - \underbrace{\theta_{\text{man}}}_{\text{[he]}} \approx \underbrace{\theta_{\text{queen}}}_{\text{[crown,she]}} - \underbrace{\theta_{\text{woman}}}_{\text{[she]}}$$

Unsupervised learning

- Principle: make up prediction tasks (e.g., x given x or context)
- Hard task $\rightarrow$ pressure to learn something
- Loss minimzation using SGD
- Discriminatively fine tune: initialize feedforward neural network and backpropagate to optimize task accuracy
- Helps less given large amounts of labeled data, but doesn't mean unsupervised learning is solved — quite the opposite!

Roadmap

Supervised learning

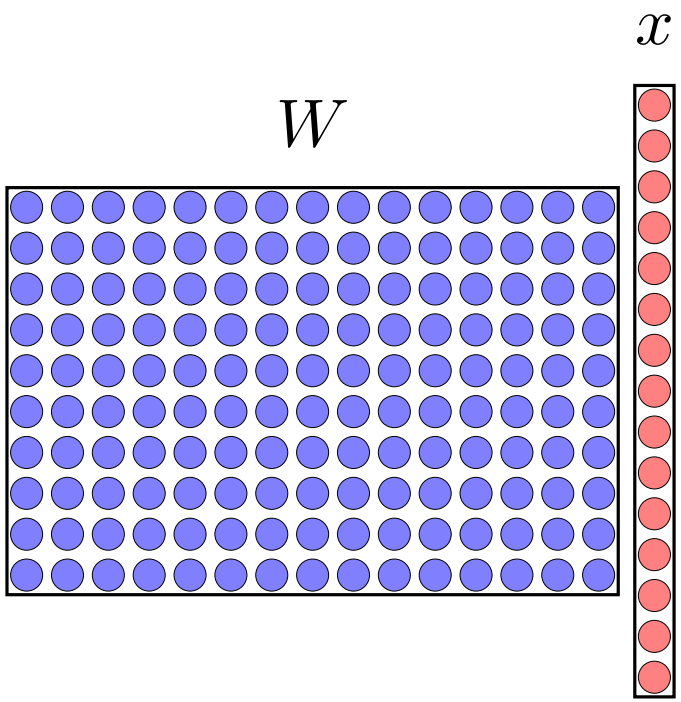
Unsupervised learning

Convolutional neural networks

Recurrent neural networks

Final remarks

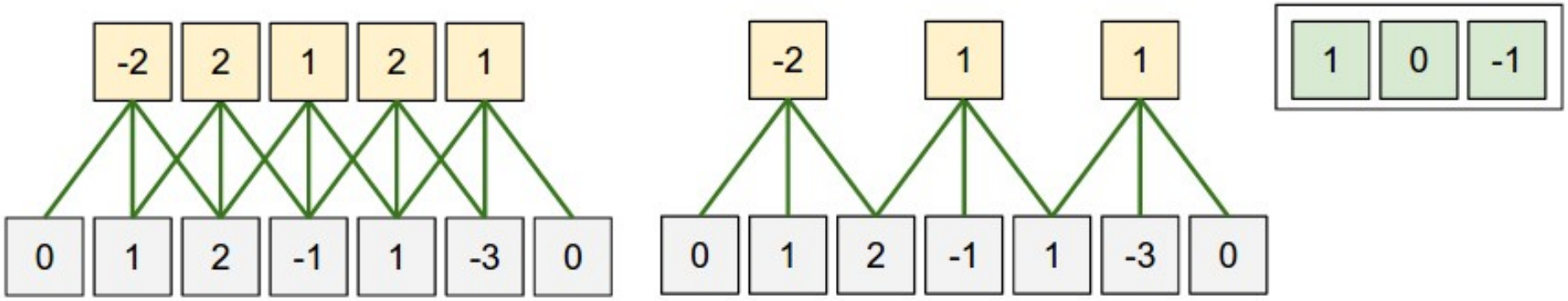
Motivation



- Observation: images are not arbitrary vectors
- Goal: leverage spatial structure of images (translation invariance)

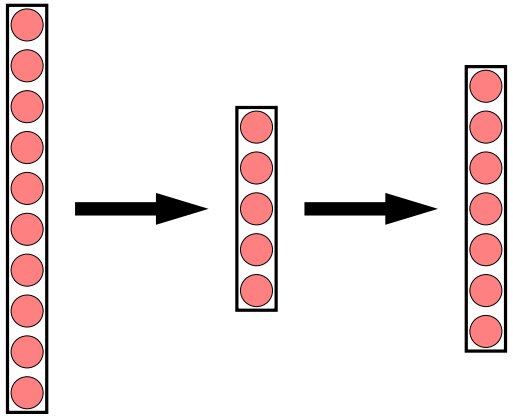
[figure from Andrej Karpathy]

Prior knowledge

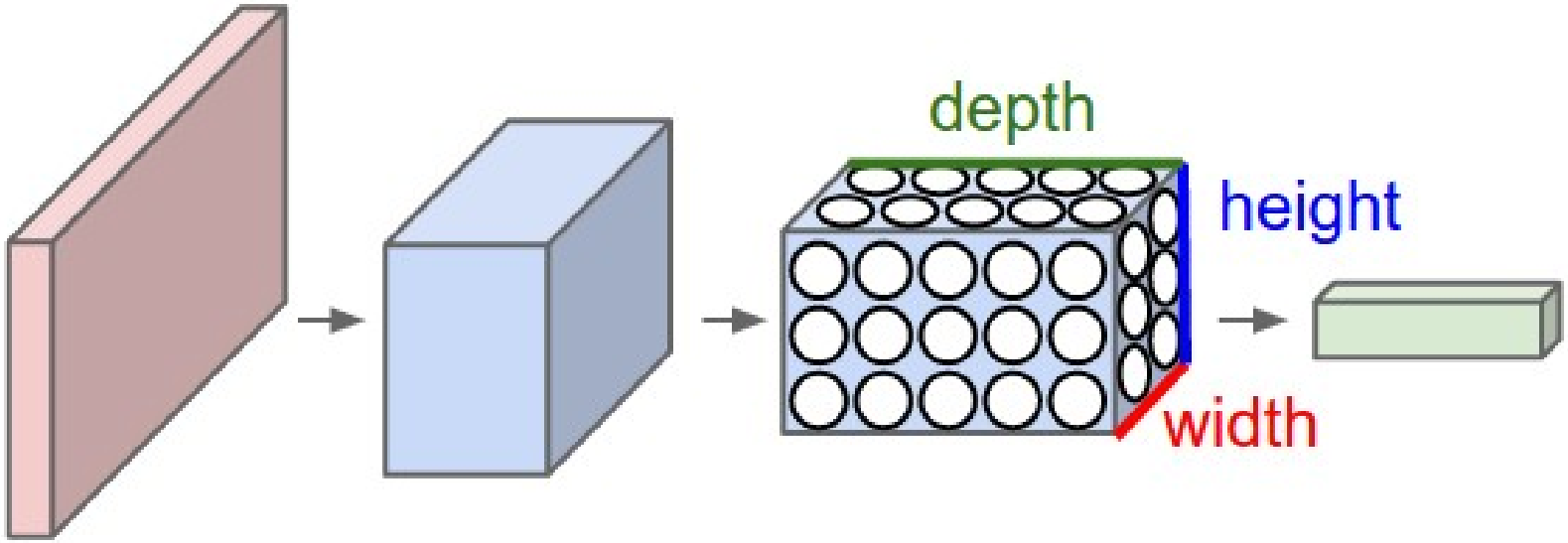


- Local connectivity: each hidden unit operates on a local image patch (3 instead of 7 connections per hidden unit)
- Parameter sharing: processing of each image patch is same (3 parameters instead of $3 \cdot 5$)
- Intuition: try to match a pattern in image

Fully-connected:



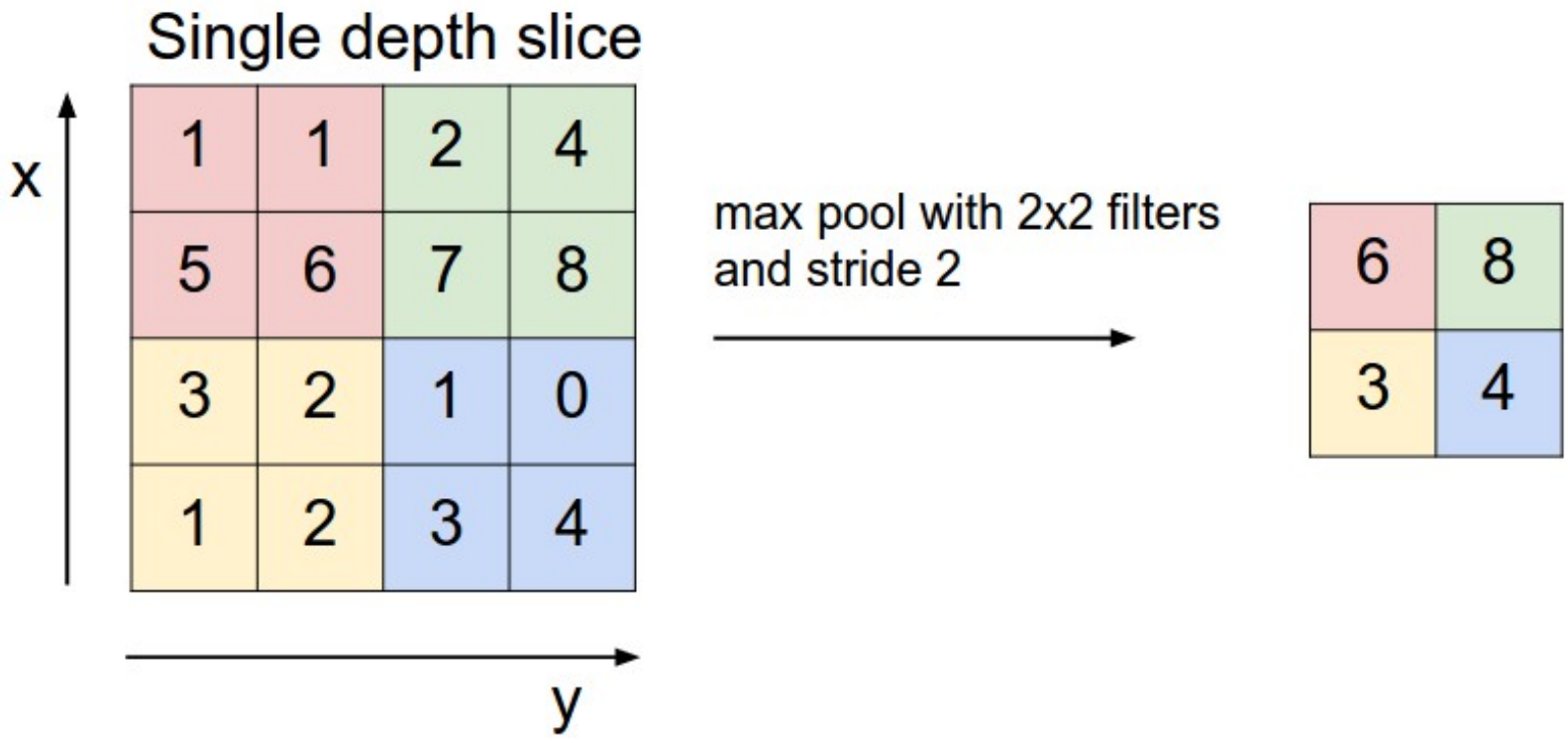
Convolutional: each depth column produced from localized region (in height/width)



[Andrej Karpathy's demo]

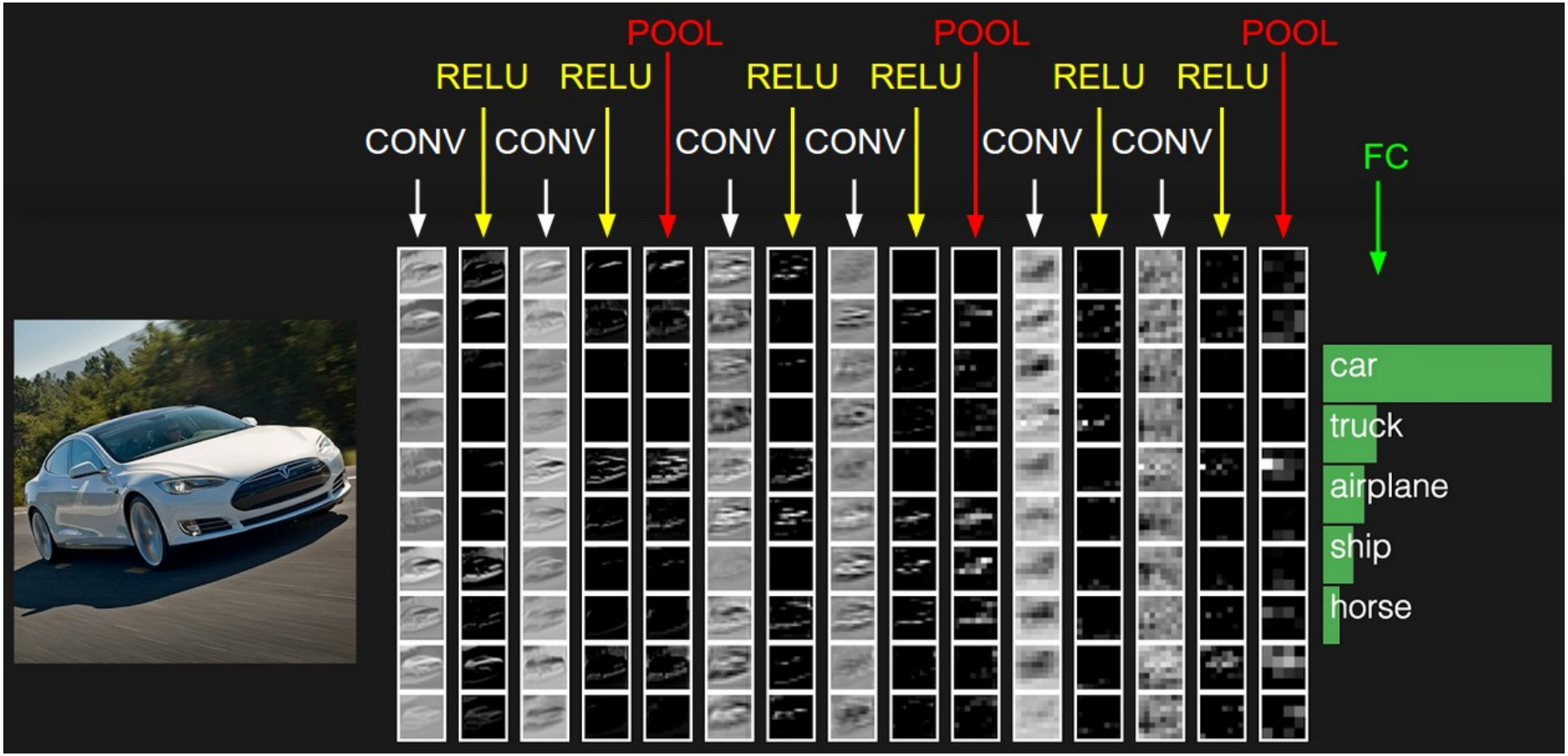
[figure from Andrej Karpathy]

Max-pooling



- Intuition: test if there exists a pattern in neighborhood
- Reduce computation, prevent overfitting

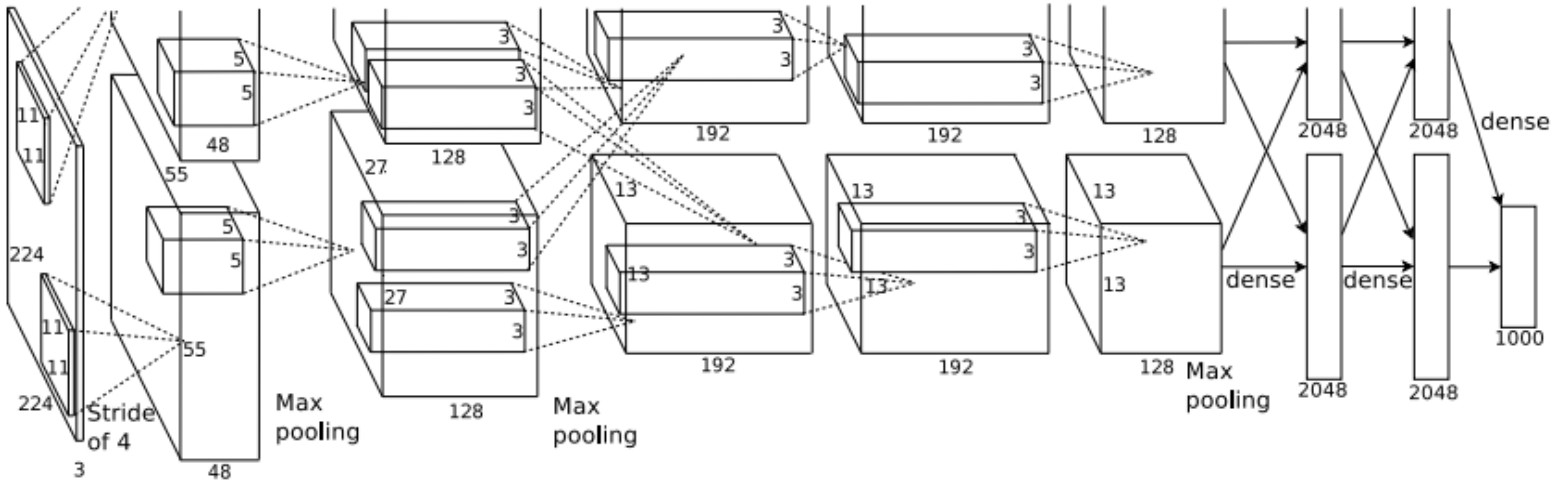
Example of function evaluation



[Andrej Karpathy's demo]

[Krizhevsky et al., 2012, a.k.a. AlexNet]

AlexNet



- Non-linearity: use ReIU $(\max(z, 0))$ instead of logistic
- Data augmentation: translate, horizontal reflection, vary intensity, dropout (guard against overfitting)
- Computation: parallelize across two GPUs (6 days)
- Impressive results: 15% error; next best was 25%!

Summary

- Intuition: spatial regularity across the input
- Key idea: locality and parameter sharing
- Dominant in computer vision
- Applications to text classification and speech recognition

Roadmap

Supervised learning

Unsupervised learning

Convolutional neural networks

Recurrent neural networks

Final remarks

Motivation

Model sequences (sentences):

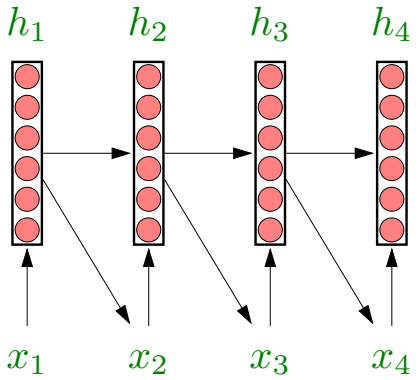
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
Paris	*Talks*	*Set*	*Stage*	*for*	*Action*	*as*	*Risks*	*to*	*the*	*Climate*	*Rise*

Goal: rich probabilistic model

$$p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2)p(x_4 \mid x_1, x_2, x_3) \cdots$$

No conditionally independence!

Recurrent neural networks



$h_1 = \text{Encode}(x_1)$

$x_2 \sim \text{Decode}(h_1)$

$h_2 = \text{Encode}(h_1, x_2)$

$x_3 \sim \text{Decode}(h_2)$

$h_3 = \text{Encode}(h_2, x_3)$

$x_4 \sim \text{Decode}(h_3)$

$h_4 = \text{Encode}(h_3, x_4)$

Update context vector:

$$h_t = \text{Encode}(h_{t-1}, x_t)$$

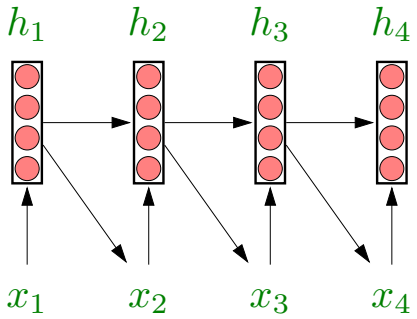
Predict next character:

$$x_{t+1} = \text{Decode}(h_t)$$

context h_t compresses $x_1, \ldots x_t$

[Elman, 1990]

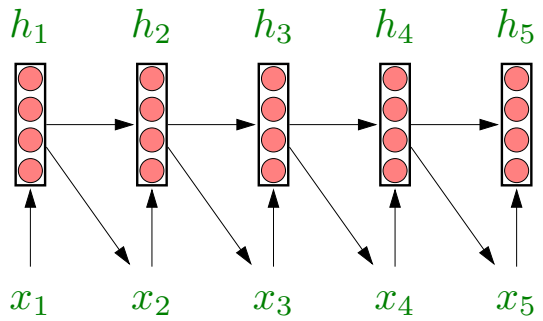
Simple recurrent network



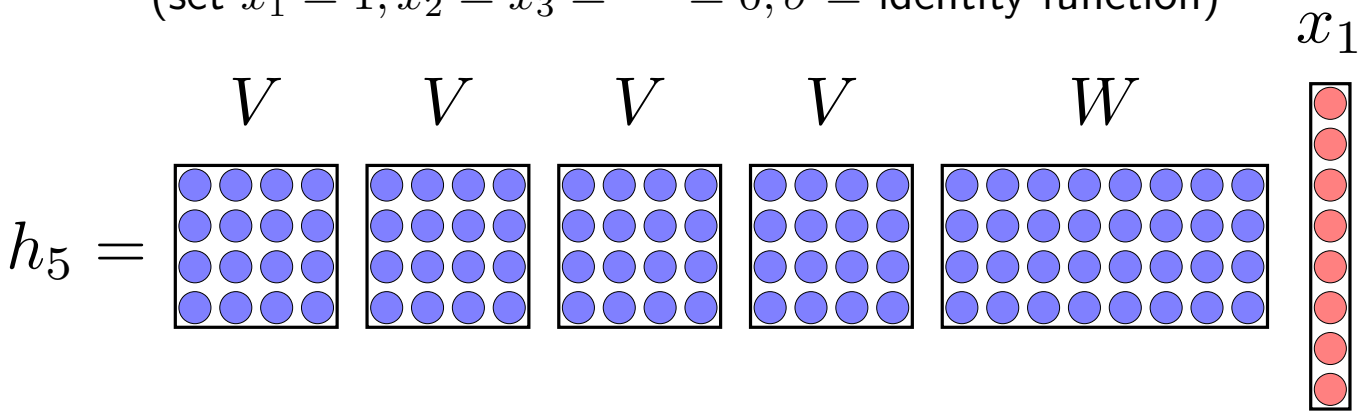
$$\text{Encode}(h_{t-1}, x_t) = \sigma(V \; h_{t-1} + W \; x_t) = h_t$$

$$\text{Decode}(h_t) \sim \text{softmax}(W' \; h_t) = p(x_{t+1})$$

Vanishing gradient problem



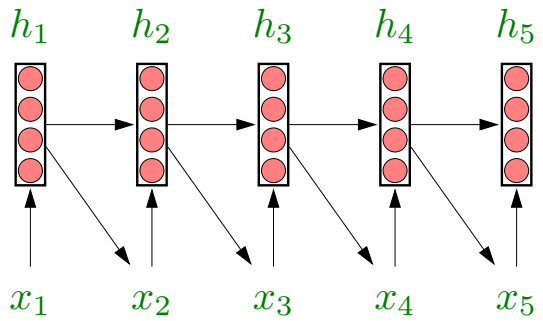
(set $x_1 = 1, x_2 = x_3 = \cdots = 0, \sigma =$ identity function)



If $V = 0.1$, then

- Value: $h_t = 0.1^{t-1} W$
- Gradient: $\frac{\partial h_t}{\partial W} = 0.1^{t-1}$ (vanishes as length increases)

Additive combinations



What if:

$$h_t = h_{t-1} + Wx_t$$

Then:

(set $x_1 = 1, x_2 = x_3 = \cdots = 0, \sigma =$ identity function)

- Value: $h_t = W$
- Gradient: $\frac{\partial h_t}{\partial W} = 1$ for any t

[Schmidhuber & Hochreiter, 1997]

Long Short Term Memory (LSTM)

API:

$$(h_t, c_t) = \text{LSTM}(h_{t-1}, c_{t-1}, x_t)$$

Input gate:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i)$$

Forget gate (initialize with b_f large, so close to 1):

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f)$$

Cell: additive combination of RNN update with previous cell

$$c_t = i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) + f_t \odot c_{t-1}$$

Output gate:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o)$$

Hidden state:

$$h_t = o_t \odot \tanh(c_t)$$

[from Andrej Karpathy's blog]

Character-level language modeling

Sampled output:

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25—21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict.

[from Andrej Karpathy's blog]

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                       struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
      df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
  int i;
  if (classes[class]) {
    for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
      if (mask[i] & classes[class][i])
        return 0;
  }
  return 1;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

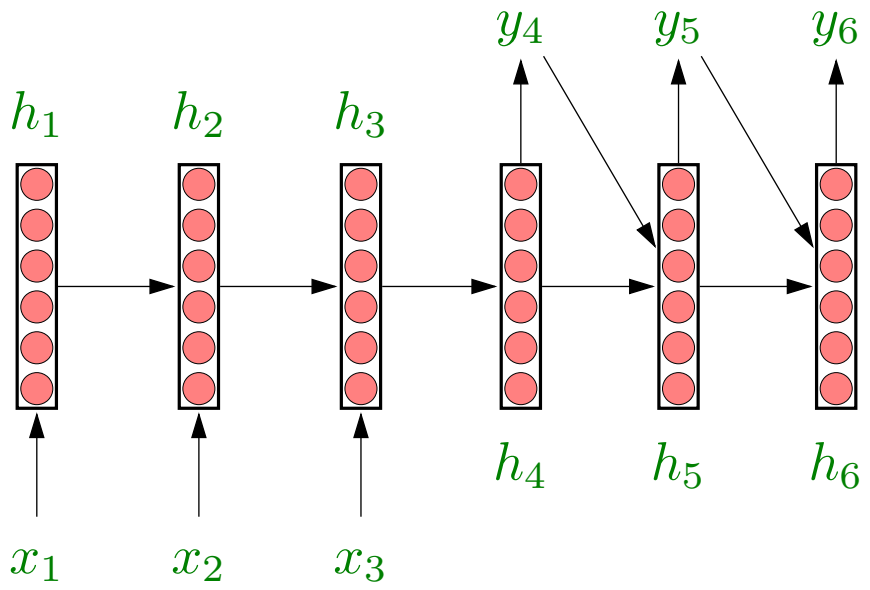
[Sutskever et al., 2014]

Sequence-to-sequence model

Motivation: machine translation

x: *Je crains l'homme de un seul livre.*

y: *Fear the man of one book.*



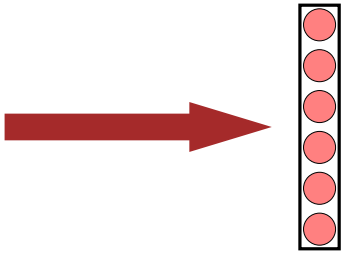
Read in a sentence first, output according to RNN:

$$h_t = \text{Encode}(h_{t-1}, x_t \text{ or } y_{t-1}), \quad y_t = \text{Decode}(h_t)$$

Attention-based models

Motivation: long sentences — compress to finite dimensional vector?

Eine Folge von Ereignissen bewirkte, dass aus Beethovens Studienreise nach Wien ein dauerhafter und endgültiger Aufenthalt wurde. Kurz nach Beethovens Ankunft, am 18. Dezember 1792, starb sein Vater. 1794 besetzten französische Truppen das Rheinland, und der kurfürstliche Hof musste fliehen.

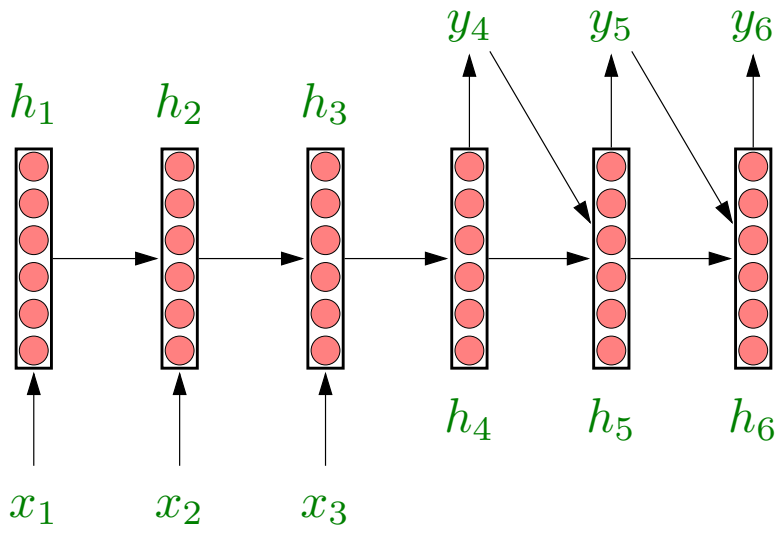


Key idea: attention

Learn to look back at your notes.

[Bahdanau et al., 2015]

Attention-based models



Distribution over input positions:

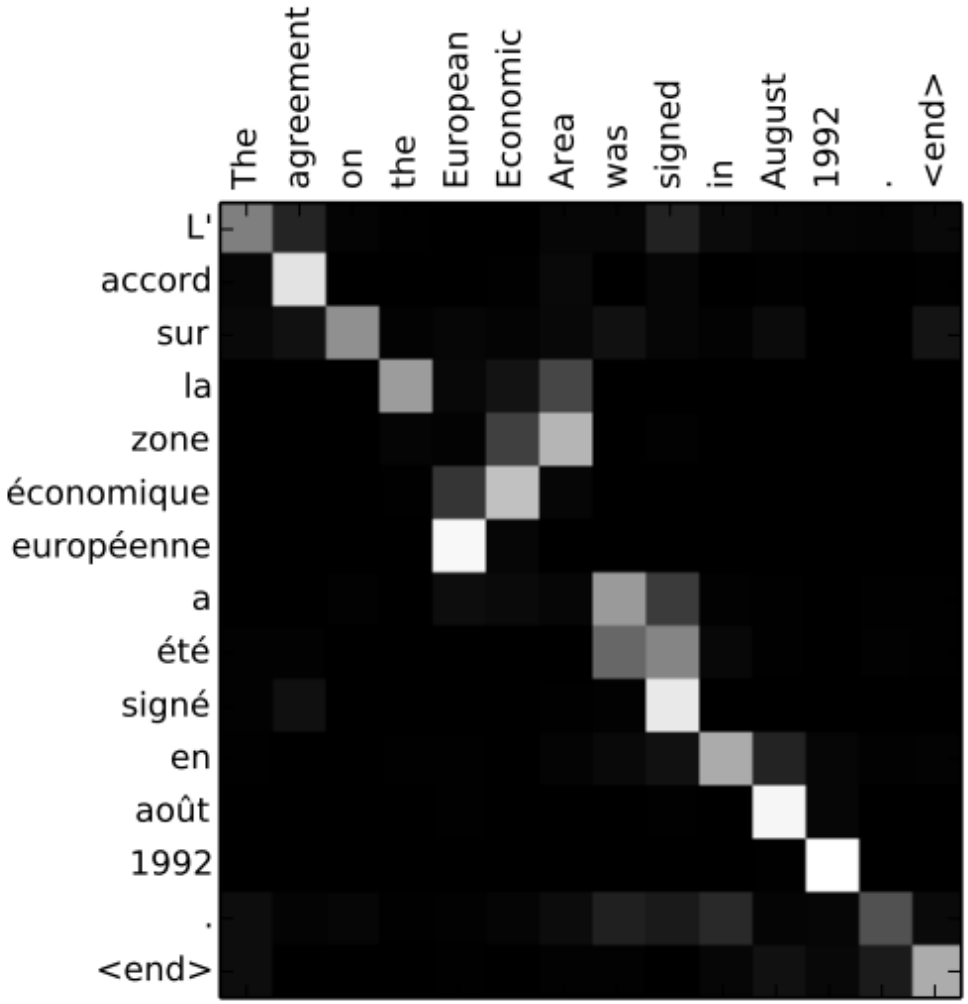
$$\alpha_t = \text{softmax}([\text{Attend}(h_1, h_{t-1}), \ldots, \text{Attend}(h_L, h_{t-1})])$$

Generate with attended input:

$$h_t = \text{Encode}(h_{t-1}, y_{t-1}, \sum_{j=1}^{L} \alpha_t h_j)$$

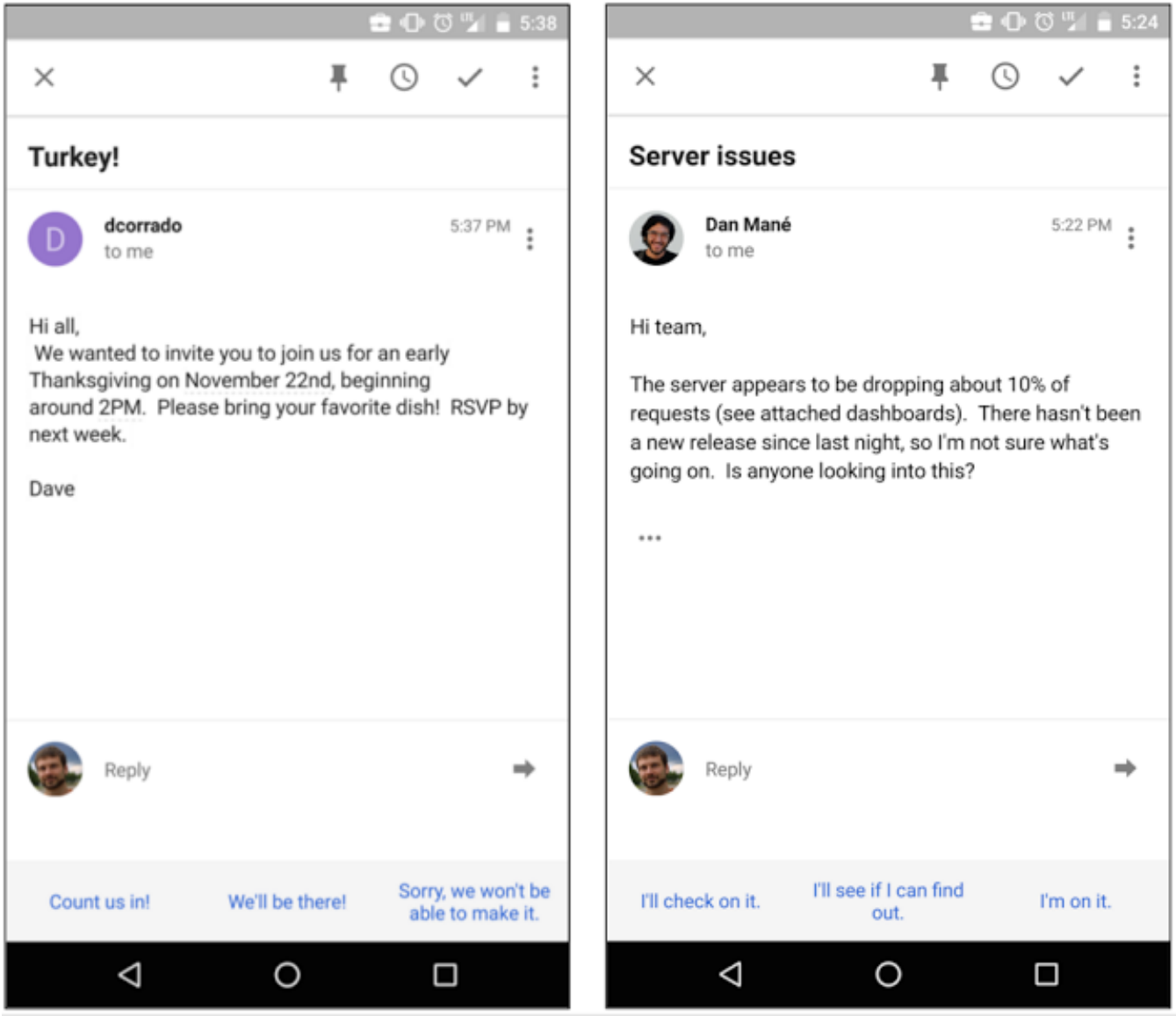
[Bahdanau et al., 2015]

Machine translation



[Google, 2015]

Email responder



[Xu et al., 2015]

Image captioning

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

Summary

- Recurrent neural networks: model sequences (non-linear version of Kalman filter or HMM)
- Logic intuition: learning a program with a for loop (reduce)
- LSTMs mitigate the vanishing gradient problem
- Attention-based models: when only part of input is relevant at a time
- Newer models with "external memory": memory networks, neural Turing machines

Roadmap

Supervised learning

Unsupervised learning

Convolutional neural networks

Recurrent neural networks

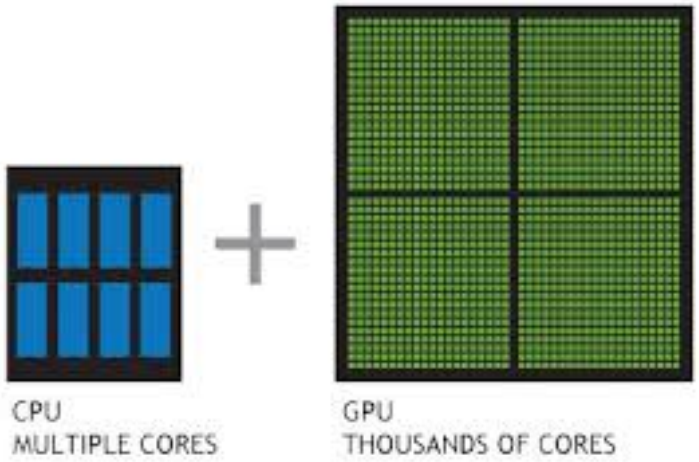
Final remarks

Computation

...wait for a long time...

Better optimization algorithms: SGD, SGD+momentum, AdaGrad, AdaDelta, momentum, Nesterov, Adam

Buy GPUs:



...wait for a long time...

Theory: why does it work?

Two questions:

- Approximation: why are neural networks good hypothesis classes?
- Optimization: why can SGD optimize a high-dimensional non-convex problem?

Partial answers:

- 1-layer neural networks can approximate any continuous function on compact set [Cybenko, 1989; Barron, 1993]
- Generate random features works too [Rahimi/Recht, 2009; Andoni et. al, 2014]
- Use statistical physics to analyze loss surfaces [Choromanska et al., 2014]

Summary

Phenomena	Ideas
Fixed vectors	Feedforward NNs
Spatial structure	convolutional NNs
Sequence	recurrent NNs LSTMs
Sequence-to-sequence	encoder-decoder attention-based models
Unsupervised	belief networks RBMs autoencoders

References

Tutorials:

- http://deeplearning.net/tutorial/
- http://deeplearning.stanford.edu/tutorial/
- http://cs.stanford.edu/people/karpathy/convnetjs/

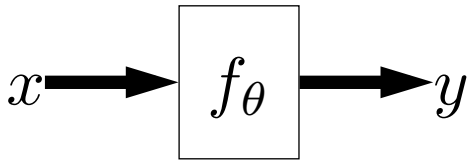
Software:

- Caffe (Berkeley): centered around computer vision
- Theano (Montreal); also see Keras: Python
- Torch (Facebook): fast, but write in Lua
- TensorFlow (Google): new!

Outlook

Extensibility: able to compose modules

Learning programs: think about analogy with a computer



Data:

reinforcement learning? unsupervised learning?