Lecture 8

Deep learning

1

Overview

- Things we will discuss today
  - Review of Progress Exam 1
  - Visualizing what ConvNets Learn
  - Meta-Algorithms for optimizing the network - Batch Normalization
  - Introduction to RNNs

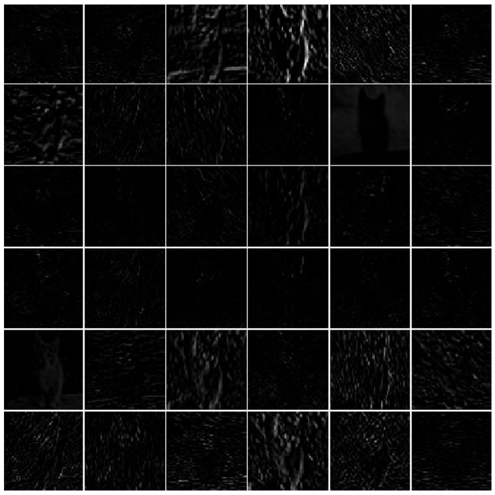
2

Visualizing what ConvNets Learn

Developed as a response to a common criticism that neural networks are not interpretable

Several methods:

1. Visualizing the activations Show the activations during the forward pass
   - Some activation map zero for many different inputs, which can indicate *dead* filters, and can be a symptom of high learning rates.



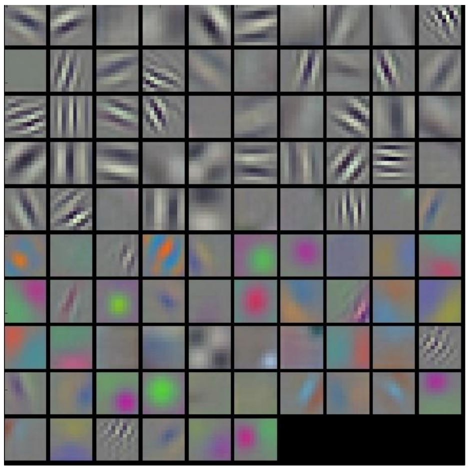
A trained AlexNet looking at a picture of a cat.

http://cs231n.github.io/understanding-cnn/

3

2- Visualize the filter weights.

- Well-trained networks show nice and smooth filters without noisy patterns.
- Noisy patterns can be an indicator of a network that hasn't been trained for long enough, or an overfitting scenario



Filters on the first CONV layer of a trained AlexNet

http://cs231n.github.io/understanding-cnn/

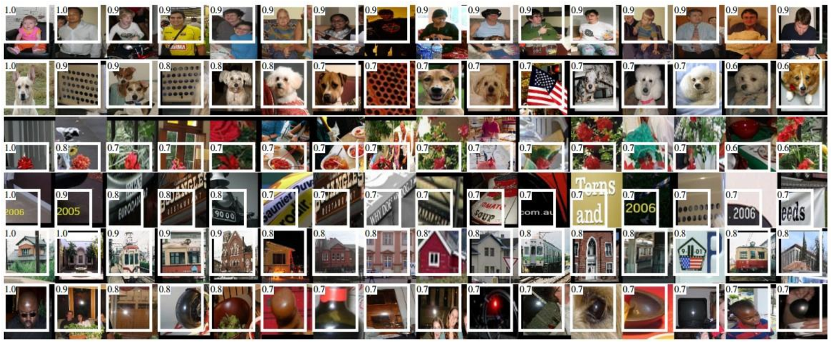
4

3- Keeping track of instances that maximally activate a neuron

- Try to understand what a neuron is looking for by looking into the images that activate it

- One problem with this approach is that ReLU neurons do not necessarily have any semantic meaning by themselves

- Rather, it is more appropriate to think of multiple ReLU neurons as the basis vectors of some space that represents in image patches.



Maximally activating images for some POOL5 (5th pool layer) neurons of an AlexNet.

http://cs231n.github.io/understanding-cnn/

5

t-SNE Embedding (t-distributed stochastic neighbor)

- A dimensionality reduction technique that is used for visualization.
- Unlike PCA, it can capture non-linear structures
- Used for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.
- Developed by van der Maaten and Geoffrey Hinton
- We can get a rough idea about the topology of this space by embedding images so that their low-dimensional representation has approximately equal distances than their high-dimensional representation.

Interactive demo
https://distill.pub/2016/misread-tsne/

Images that are nearby each other are also close in the CNN representation space, which implies that the CNN "sees" them as being very similar.

https://cs.stanford.edu/people/karpathy/cnnembed/

6

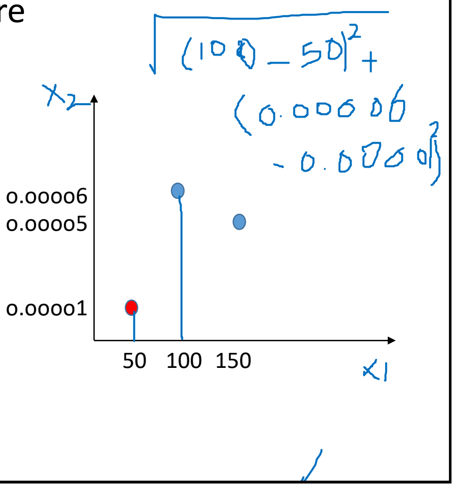
Normalization – review

- Suppose you have two features x_1, x_2. x_1 is in the range [0, 500] and x_2 in the range [0, 0.0001] and you are measuring the Euclidean distance. Which feature dominates the calculation of such distance.
- Normalization formula

$$\overline{x_1}^{(j)} = \frac{x_1{}^{(j)} - minx_1}{maxx_1 - minx_1}$$

- Standardization

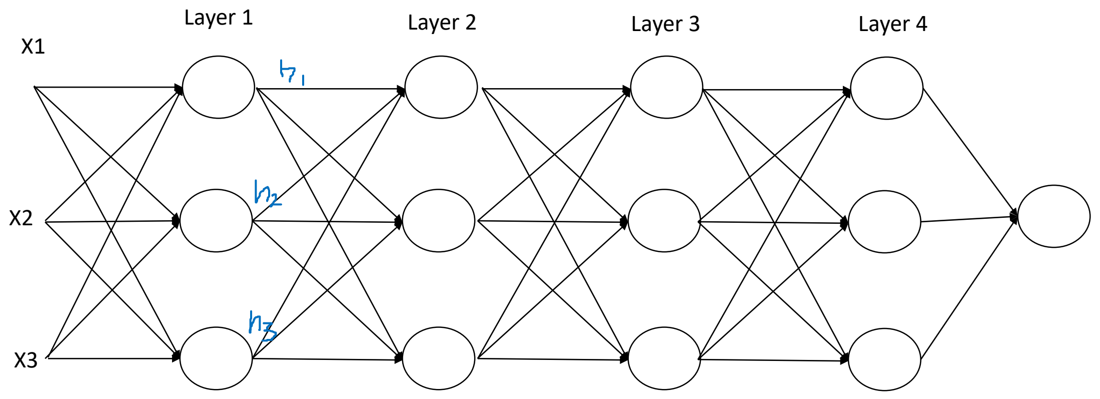
$$\overline{x_1}^{(j)} = \frac{x_1{}^{(j)} - \mu_{x_1}}{\sigma_{x_1}}$$



Goodfellow, Bengio, Courville 2016

7

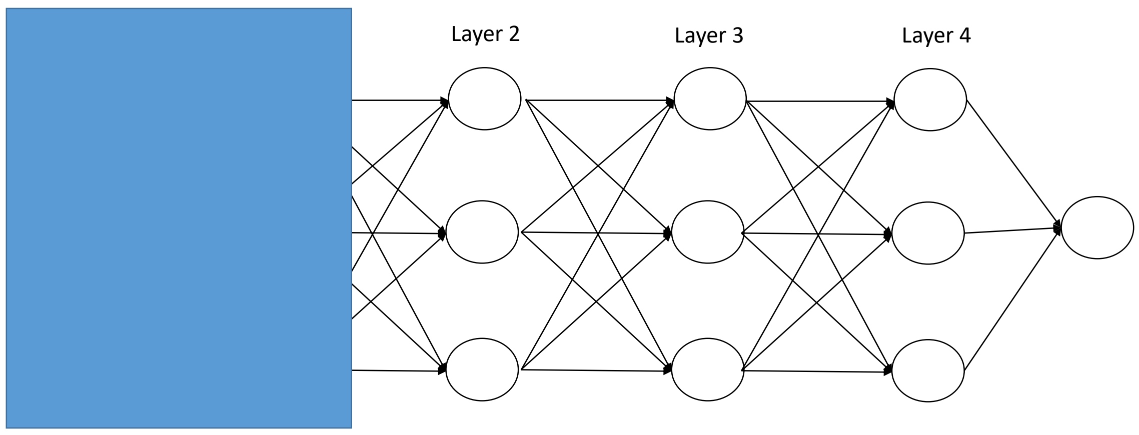
Batch Normalization (Can we use the same principle for hidden nodes?)



Goodfellow, Bengio, Courville 2016

8

Batch Normalization (Can we use the same principle for hidden nodes?)



e.g. normalize the input to layer 2

Goodfellow, Bengio, Courville 2016

9

Batch Normalization

- Let H be a minibatch of activations of the layer to normalize, arranged as a design matrix, with the activations for each example appearing in a row of the matrix. To normalize H, we replace it with

$$H' = \frac{H - \mu}{\sigma}$$

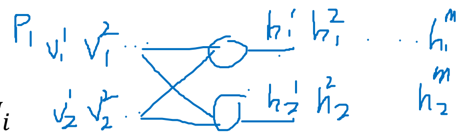
- where μ is a vector containing the mean of each unit and σ is a vector containing the standard deviation of each unit.
- How can we determine μ, σ (learnable parameters)

Goodfellow, Bengio, Courville 2016

10

Batch Normalization

- Sum over the examples in one minibatch



$$\mu = \frac{1}{m}\sum_i H_i$$

$$\sigma = \sqrt{\delta + \frac{1}{m}\sum_i H - \mu)_i^2}$$

- where δ is a small positive value such as 10^{-8} imposed to avoid encountering the undefined gradient of $\sqrt{z}$ at z = 0.
- Normalizing the mean and standard deviation of a unit can reduce the expressive power of the neural network containing that unit. In order to maintain the expressive power of the network, it is common to replace the batch of hidden unit activations H with $\gamma H' + \beta$
- γ and β are learned parameters that allow the new variable to have any mean and standard deviation.

Goodfellow, Bengio, Courville 2016

11

Batch Normalization –test time

- At test time, you cannot calculate μ and σ using a minibatch. How do we perform batch normalization?
- We cannot use a μ and σ calculated from one example
- At test time, μ and ***σ*** may be replaced by running averages (typically exponentially weighted average) that were collected during training time.
- This allows the model to be evaluated on a single example, without needing to use definitions of **μ** and **σ** that depend on an entire minibatch.

Goodfellow, Bengio, Courville 2016

12

Batch Normalization (section 8.7.1)

- Paper: "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" Sergey Ioff and Christian Szegedy
- One of the methods used in optimizing deep neural networks and it is actually not an optimization algorithm at all.
- A method of adaptive re-parametrization, motivated by the difficulty of training very deep models
- Batch normalization can be applied to any input or hidden layer in a network. It significantly reduces the problem of coordinating updates across many layers by reducing internal covariate shift.

Goodfellow, Bengio, Courville 2016

13

Batch Normalization (how it works)

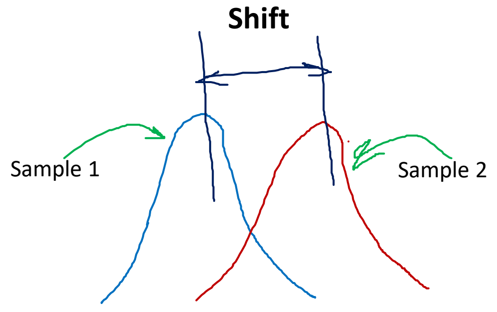
- Very deep models involve the composition of several functions or layers.
$$f^{(1)}(f^{(2)}(f^{(3)}(x))$$
- The gradient tells how to update each parameter, under the assumption that the other layers do not change. In practice, we update all of the layers simultaneously.
- When we make the update, unexpected results can happen because many functions composed together are changed simultaneously, using updates that were computed under the assumption that the other functions remain constant.

Goodfellow, Bengio, Courville 2016

14

Covariate Shift

- When there is a change in the probability distribution of input X while P(Y|X) is the same.
- By reducing internal covariate shift among layers, batch normalization helps to accelerate the training of deep networks allowing faster optimization of the network



Goodfellow, Bengio, Courville 2016

Parameter Initialization Strategies (8.4)

- Most algorithms when training deep learning models are strongly affected by the choice of initialization.
- The initial point can determine whether the algorithm converges at all.
- Designing improved initialization strategies is a difficult task because neural network optimization is not yet well understood.
- Most initialization strategies are based on achieving some nice properties when the network is initialized.
- Perhaps the only property known with complete certainty is that the initial parameters need to "break symmetry" between different units.
- If two hidden units with the same activation function are connected to the same inputs, then these units must have different initial parameters. (why?) [illegible]

Goodfellow, Bengio, Courville 2016

Parameter Initialization Strategies (cont.)

- If they have the same initial parameters, then a deterministic learning algorithm applied to a deterministic cost and model will constantly update both of these units in the same way.
- Even if the model or training algorithm is capable of using stochasticity to compute different updates for different units (for example, if one trains with dropout), it is usually best to initialize each unit to compute a different function from all of the other units.
- The goal of having each unit compute a different function motivates random initialization of the parameters.
- We could explicitly search for a large set of basis functions that are all mutually different from each other, but this often incurs a noticeable computational cost

Goodfellow, Bengio, Courville 2016

17

Parameter Initialization Strategies (cont.)

- Typically, we set the biases for each unit to heuristically chosen constants, and initialize only the weights randomly.
- We almost always initialize all the weights in the model to values drawn randomly from a Gaussian or uniform distribution.
- The choice of Gaussian or uniform distribution does not seem to matter very much, but has not been exhaustively studied.
- The scale of the initial distribution, however, does have a large effect on both the outcome of the optimization procedure and on the ability of the network to generalize.

Goodfellow, Bengio, Courville 2016

18

Parameter Initialization Strategies (cont.)

- Optimization suggests that the weights should be large enough to propagate information successfully
- Larger initial weights will yield a stronger symmetry breaking effect, helping to avoid redundant units.
- They also help to avoid losing signal during forward or back-propagation through the linear component of each layer—larger values in the matrix result in larger outputs of matrix multiplication.
- Initial weights that are too large may, however, result in exploding values during forward propagation or back-propagation.
- In recurrent networks, large weights can also result in increased sensitivity to small perturbations of the input.

Goodfellow, Bengio, Courville 2016

19

Example: Xavier Initialization

- Some heuristics are available for choosing the initial scale of the weights.
- An example, **Xavier Initialization** (Glorot and Bengio (2010)) suggests that to initialize the weights of a fully connected layer with m inputs and n outputs by sampling each weight from

$$W_{i,j} \sim U(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}})$$

Goodfellow, Bengio, Courville 2016

20

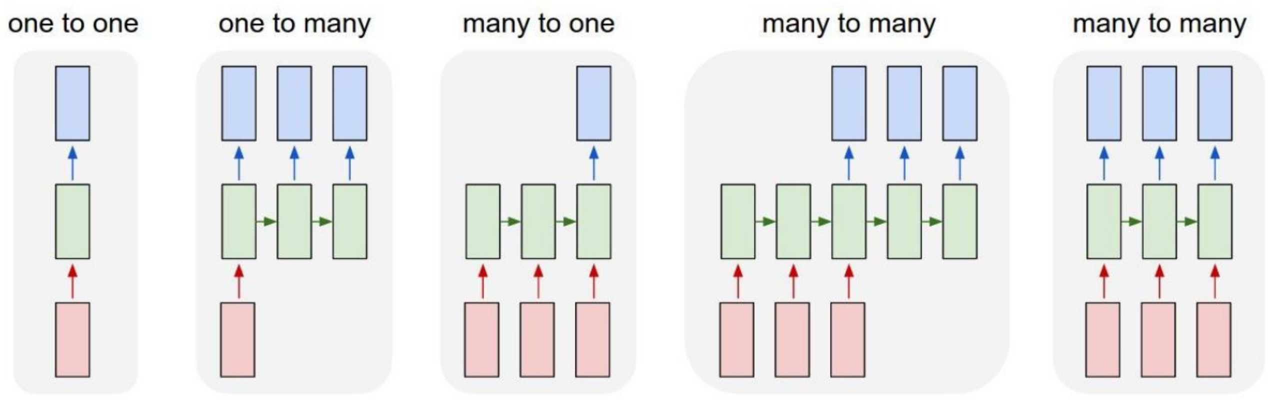
Sequence Modelling

- Typical MLPs accept an input of fixed dimensionality and map it to an output of fixed dimensionality
- It is required to work on sequences that are of arbitrary lengths (a sequence of words in a machine translation application) and produce output of an arbitrary length.
- MLPs treat every training example independently. Not a good approach when you deal with a sequence of values. Why?
- In sequence modelling, we typically need to reuse the knowledge about previous events to help classifying the current.

Goodfellow, Bengio, Courville 2016

21

Sequence Modeling



e.g. image Captioning

e.g. Sentiment Classification

e.g. Machine translation

e.g. Video Classification

http://cs231n.github.io/understanding-cnn/

22

Example

- Consider the two sentences “I went to Nepal in 2009” and “In 2009, I went to Nepal.” If we ask a machine learning model to read each sentence and extract the year in which the narrator went to Nepal, we would like it to recognize the year 2009 as the relevant piece of information, whether it appears in the sixth word or the second word of the sentence.
- Suppose that we trained a feedforward network that processes sentences of fixed length. A traditional fully connected feedforward network would have separate parameters for each input feature, so it would need to learn all of the rules of the language separately at each position in the sentence.

Goodfellow, Bengio, Courville 2016

23

Recurrent Neural Network

- A family of neural networks for processing sequential data
- operating on a sequence that contains vectors $x^{(t)}$ with the time step index t ranging from 1 to τ . $(x^{(1)}, \dots ,\ x^{(\tau)})$
- Uses a form of parameter sharing that makes it possible to extend and apply the model to examples of different forms (different lengths) and generalize across them.
- Recurrent networks share parameters in a way that is different from CNN (a filter defined by same parameters). Each member of the output is a function of the previous members of the output.

Goodfellow, Bengio, Courville 2016

24

Recurrent Neural Network

- Consider the classical form of a dynamical system:

$$s^{(t)} = f(s^{t-1}; \theta)$$

$s^{(t)}$: state of the system

- The equation is recurrent because the definition of s at time t refers back to the same definition at t-1

$$s^{(3)} = f\left(s^{(2)}; \theta\right) = f(f(s^{(1)}; \theta); \theta)$$

- let us consider a dynamical system driven by an external input signal x(t)

$$s^{(t)} = f\left(s^{(t-1)}, x^{(t)}; \theta\right)$$

Goodfellow, Bengio, Courville 2016

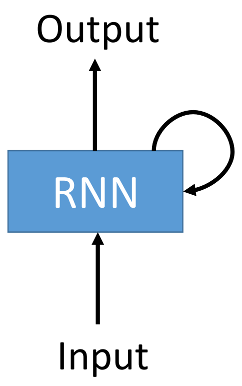
25

Recurrent Neural Network

- **The recurrence formula**

$$h^{(t)} = f\left(h^{(t-1)}, x^{(t)}; \theta\right)$$

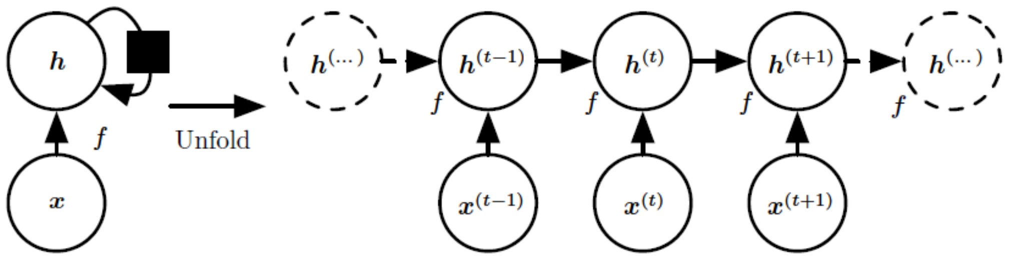
- x(t) an input sequence at time t
- Same function and same set of parameters are used at each time step



Goodfellow, Bengio, Courville 2016

26

Unfolding



Goodfellow, Bengio, Courville 2016

27