Deep Learning Summer School

Michael Noukhovitch

Summer 2018

Contents

1 Neural Networks I (Hugo Larochelle)

1.1 Basics

1.1.1 Artificial Neuron

pre-activation $a(x) = w^T x + b$

activation function $h(x) = g(a(x))$

1.1.2 Capacity of Neural Networks

combining two simple linear neurons, can create a more complex non-linear shape

universal approximation a single hidden layer NN can approximate any continuous function arbitrarily well (given enough neurons)

1.1.3 Multilayer Neural Network

we can have L hidden layers, e.g. at layer k

$$a^{(k)}(x) = b^{(k)} + W^{(k)} h^{(k-1)}(x)$$
$$h^{(k)}(x) = g(a^{(k)}(x))$$

output layer $(k = L + 1)$

$$h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x)$$

1.1.4 Activation Function

sigmoid $\frac{1}{1+\exp(-a)}$ probability of a bernouilli

tanh $\frac{\exp(a)-\exp(-a)}{\exp(a)+\exp(-a)}$ squashes between -1 and 1

relu $\max(0, a)$ does not saturate

softmax multi-class conditional probabilities

1.2 Training Neural Networks

1.2.1 Optimization

learning as an optimization problem

loss function $l(f(x; \theta), y)$ between output and true label

regularizer $\Omega(\theta)$ penalize certain parameters θ

use **Stochastic Gradient Descent**

1. initialize θ

   for N epochs

for each example x, y

2. calculate gradient $\Delta = -\nabla l(\ldots)$
3. take a step $\theta \leftarrow \theta + \alpha\Delta$

1.2.2 Loss Function

maximize the probability of correct class (**maximum likelihood**)
equivalently **minimize** the negative log-probability aka **cross-entropy**

$$l(f(x), y) = -\sum_c 1_(y = c) \ldots$$

1.2.3 Backpropogation

use chain rule to compute gradients

1. compute gradient wrt pre-activation

$$\nabla_{a^{(L+1)}(x)} - \log f(x)_y \leftarrow -(e(y) - f(x))$$

for layer k from $L + 1$ to 1

2. compute gradient of hidden layer parameter
3. compute gradient of hidden layer below
4. compute gradient of pre-activation below

reversing the flow graph representation gives us backprop for free Section 2

1.2.4 Initialization

- biases $\leftarrow 0$
- weights $\leftarrow$ random sample
  - 0 doesn't work with tanh
  - same value doesn't work
  - break symmetry, close to 0

1.2.5 Model Selection

search for the best hyperparameters with

- **grid search** search all possible options
- **random search** sample from distribution over hyperparams
- bayesian optimization, pbt, ...

use a **validation set** of examples to choose the best model
stop training with **early stopping** when validation error is lowest

1.2.6 Tricks

- **normalize** your (real-valued) data
- **decay** your learning rate
- update your gradient on a **batch** of examples
- use exponential average of previous gradients, "gaining **momentum**"
- use adaptive learning rates: **Adagrad**, **RMSProp**, **Adam**

1.2.7 Gradient Checking

debug your implementation of fprop/bprop with a finite-difference approximation

$$\frac{df(x)}{dx} \approx \frac{f(x+\epsilon) - f(x-\epsilon)}{2\epsilon}$$

1.2.8 Debug on Small Dataset

overfit on a subsubset of your dataset, issues:

- units saturated before first update? initialization, regularization
- training error unstable? learning rate scheduling

1.3 Regularization

1.3.1 Unsupervised Pretraining

unsupervised pretraining initialize hidden layers by using unsupervised learning to represent the latent structure of data

fine-tuning training after initialization to adapt to data

auto-encoder feed-forward NN traine to reproduce it's input

1.3.2 Dropout

dropout remove hidden units stochastically (usually $p = 0.5$)

1.3.3 Batch Normalization

normalize inputs to speed up training

- normalize each unit's pre-activation
- subsubtract mean and std deviation, calculated per minibatch
- learn a linear transformation of the normalized pre-activation
- account for it during backprop
- use global mean and std deviation at test time

2 Automatic Differentiation (David Duvenaud)

2.1 Basic Autodiff

auto diff programmatically finding gradients for operations

forward mode building the jacobian on, in the order of initial operations (expensive)

reverse mode keeping track of jacobian at every step and adding then in reverse

2.1.1 Implementation

static read and generate source code (Tensorflow)

dynamic monitor function exection at runtime (Pytorch, autograd)

1. trace execution as composition of primitives
2. define vector-Jacobian product (VJP) operator for each primitive
3. compose VJPs backwards

fun stuff:

- we get higher order autodiff for free
  since tape-based dynamic autodiff traces execution, we automatically trace the autodifferentiation itself, and just need to follow the execution trace of the previous autodiff
- forward mode is actually just a special case of reverse mode
- VJPs are as cheap as gradients

2.2 Advanced Autodiff

2.2.1 Higher Order Ops

higher order gradients are possible by changing vector-Jacobian for vector-Hessian etc ...

2.2.2 Meta-Optimization

can optimize through the whole network to learn the learning rate for the training

2.2.3 Implicit Function Theorem

3 Neural Networks II (Hugo Larochelle)

3.1 Types

supervised examples have labels

unsupervised examples don't have labels

semi-supervised some examples have labels

multi-task multiple labels per example

transfer multiple labels but test on a specific label

structured labels have arbitrary structure

domain adaptation training and testing distributions are different

zero-shot examples are completely novel

3.2 Intriguing Properties

- there are powerful small changes that create visual adversarial examples
- bad local optima are unlikely (for high dimensional loss surface like NNs)
- NNs are strangely non-convex
- flat minima are better than sharp minima (probably?)
- NNs can easily memorize
- knowledge can be distilled
- catastrophic forgetting is a real issue with SGD

4 Intro to Convolutional Neural Networks (Jon Shlens)

4.1 Convolutional Neural Networks

images are different from other data

- fully connected layers would use too many parameters to input an image
- convolutions provide translational invariance

convolutional neural networks

- learn filters that pass over the image
- use fewer parameters
- use more computations

4.2 Modern Developments

- normalization stabilizes activations and is important for training
  Group Normalization (Wu and He 2018)
- vanishing gradients motivate deeper, better architectures
- architectures transfer across tasks
- learned architecture searches find even better models
  NAS (Zoph 2016), DARTS (Liu et al 2018)

4.3 Understanding CNNs

how to approach what CNNs understand at each layer?

- find images/pixels that elicit largest activation
- reconstruct image from network activations
- distort pixels to amplify activations (deep dream)
- change lower level activations while maintaining high-level (neural style transfer)

5 Deep Computer Vision (Sanja Fidler)

5.1 Segmentation

convert classification network to segmentation network

1. pre-training
2. converting fully connected to fully convolutional layers

types of segmentation

semantic assign each pixel a category based on the object it belongs to

unsupervised group pixels

co-segmentation same objects from different images

instance differentiate instances (e.g. car 1, car 2)

panoptic instance + semantic

regularize for segmentation

CRF conditional random field (unary, pairwise, and optional global term)

dense CRF all pixels are connected pairwise

5.2 3D Semantic Segmentation

fusion late fusion of CNNs over depth and image

multi-view CNN over every viewpoint

VoxNet 3D convolution over point cloud

OctNet exploit sparsity in 3D representation using octree

PointNet represent point clouds directly

3D Graph NN point cloud graph

6 Generative Models II (Phil Isola)

6.1 Data Prediction

6.1.1 Conditional Generative Models

challenges

1. output is high-dimensional, structured
2. uncertainty in mapping, distribution of possibilities

two methods

1. model $p(x,)$ and use inference to get $p(x|y)$
2. directly model $p(x|y)$

GAN

- a generator G creates fake images
- a discriminator D tries to discern between the fake image and a real image

conditional GAN

- add class y as extra input to both G and D
-

6.1.2 Structured Prediction

want to model whole joint

- a GAN with sufficient capacity samples from full joint at equilibrium
- needs sufficient capacity and a lot of data

conditional VAE

- condition on observation x
- z learns to encode difference between x and y

6.2 Domain Mapping

domain mapping given two un-paired datasets x, y, lets us translate an input x to y

cycleGAN map from $x \to \hat{y} \to \hat{x}$ and minimize reconstruction

- why do we get correct mapping? **simplicity hypothesis**
- minimizing cycle loss in turn minimizes condition entropy

domain adaption given classifier in one domain, map it to another domain

6.3 Representation Learning

Generative models learn good representations

- generative models must learn to model the data distribution
- good models should learn to model the semantics of the data
- learned semantics should correspond to the same stuff we care about

Representation Learning with Contrastive Predictive Coding (van der Oord et al 2018)

- encode sequence with RNN
- predict future latent states
- modelling new states is now easier since you have predictions about what they should be before you see them

6.4 Model-based Intelligence

Yann Lecun's cake

- cake: understanding and modelling the world around us well
- cherry on top: planning and reasoning using that representation

deep visual foresight (Finn and Levine 2017)

1. model to predict future frame given action
2. specify target future frame
3. select best action for that

world models (Ha and Schmidhuber 2018)

1. simulate video game with RNN
2. train a policy on the simulation (latent space)
3. policy works on the real game

incentivizing exploration in RL with deep predictive models (Stadie et al 2015)

1. model to predict next state given action and previous state
2. reward agent for visiting "surprising" states

7 Interpretability (Been Kim)

7.1 Why Interpretability

there is **no one-size-fits-all** method for interpretability

- decision trees can't explain the most important factor
- rule lists can be too long to be understandable

the goal is use machine learning more **responsibly**

-

the issue is **underspecification** of what we truly want

- neither more data nor better algorithms can solve it

7.2 Interpretability Methods

before training: **exploratory data analysis**

- visualize data by features, splits
- find examples that are outliers in their class

during training: building a model

- rule-based methods as a model (e.g. rule lists)
- fit a simpler function for each feature
- choose representative examples to cluster
- train on sparse features
- learn a monotonic function (always increasing)
- distill your model

after training: investigate the model

- **ablation** train without certain features to see their effect
- test **input-feature importance** by sensitivity analysis (e.g. saliency maps)
- concept activation vector **CAV** separates activations of true class and rest

7.3 Evaluating Methods

- testing with humans
- ground truth experiments

8 Theoretical Understanding (Sanjeev Arora)

8.1 Optimization

8.2 Generalization

overparametrization may help optimization
overfitting is possible
but we still have excess capacity
previous ideas

- flat minima is tough to quantify
- bounds on true capacity and parameters is elusive

noise stability networks' higher layers are stable to noise introduced in lower layers

-

8.3 Depth

does depth help?

+ better expressiveness
- more difficult optimization

but overparametrization can actually accelerate training because of SGD

8.4 Unsupervised Learning

manifold assumption

- learn joint probability $p(X, Z)$
- code Z should be a good representation of X on a manifold

GAN

- generator "wins" if objective ≈ 0, equilibrium
-

mode collapse

- theorem: mode collapse happens if discriminator is too small
- for discriminator of size N, generator with $N \log N$ images wins

8.5 Simpler Methods

we should first understand linear models before approaching deep models

- linear methods for sentence embeddings can be just as effective

9 Optimization I (Jimmy Ba)

9.1 Random Search vs Gradient Descent

learning in neural networks is difficult

- non-convex, many local optima
- not smooth, unknown lipschitz constant
- millions of trainable weights

learning with **random search**

1. perturb weights by random vector $\Delta W \sim N(0, \mu^2 I)$
2. evaluate perturbed averaged loss
3. add the perturbation weighted by the perturbed loss

!! intuitively d samples gives directional gradient,

more efficient if we query the gradient directly, **gradient descent**

9.2 Better Search Directions

single gradient can be bad on a bad curvature (zig-zagging)

- **momentum** keeping a running average of gradient updates
- momentum with a lookahead trick gives **nesterov momentum**
- stochastically sample a subset, **batch**, and **decay learning rate**

gradient descent needs a constraint for the loss

- **euclidean** constraints are used normally
- **probability** can be constrained with KL (natural gradient)

9.3 "White-Box" Optimization

first order methods can still fail

- second order methods, find good preconditioning matrix that focuses on the weight space that hasn't been explored much
- speed up learning for parameters with low variance, **Adam** and **Adagrad**
- **distributed learning** with larger batch size

KFAC, kronecker factored natural gradient

- address scalability in the natural gradient, factorize fischer information
- memory efficient and tractable inverse
- still computationally expensive, so distribute

10 Optimization II (Jorge Nocedal)

10.1 Deterministic and Stochastic

there is a continuum between two worlds, with similarities

- large-scale non-linear **deterministic** (optimal trajectory)
- **stochastic** optimization (SGD)

momentum

- 2D intuition can be deceiving
- not convergent on convex functions
- similar to CG but that breaks down with noise

SGD

- deterministic GD has simple proof of convergence in convexity
- SGD is more complex, proven to converge with $\alpha \to 0$

10.2 First and Second Order

first order methods

- lack of state
- not solvable through universal formula
- suffer from ill-conditioning

second order information

- inexact newton method with hessian sub-sampling
- natural gradient methods
- quasi-newton with progressive sampling

10.3 SGD

tricks to converge

- changing batch size instead of step size
- robust optimization (similar to adversarial learning)
- progressive sampling batch size, and learning to sample

instead of scaling step size, focus on step direction $\to$ **newton's method**

- **quasi-newton** subsamples in the case when matrix is too big to invert
- newton moves in direction of smallest eigenvalue, so approximate **inexact newton** approximates the smallest
- non-convex case can be solved by following **newton-CG** until negative curvature, then following it

11 Recurrent Neural Networks

11.0.1 Reccurent Neural Networks

11.0.2 Generative RNNs

11.0.3 Conditional RNNs

- sequence to vector
- vector to sequence
- sequence to sequence

teacher forcing during training, previous y is true y taken from training data

- at test time, prev y is generated
- mismatch

scheduled sampling stochastically choose to either generate or take y **professor forcing** using a GAN to sample trajectories?

11.1 Deeper RNNs

stacking stacking hidden layers

mixing hidden, input, output hid to out, hid to hid, in to hid

skip connections mixing, with skips

11.1.1 Architechtures

dimensional

recursive tree-structured recursive computation

multi-directional 2D graph where input comes from two separate dimensions

bidirectional concatenation of forward and backward RNNs hidden states

modifications

mutliplicative replace addition by dot product in hidden

multi-scale different time scale RNNs

11.2 Long Term Dependencies

11.2.1 The Problem

a simple neuron can store 1 bit as a dynamical system

- basins of attraction
- gradient must be high at boundary of basins

spectral radius determines stability

- if > 1 then noise can kick neuron out of state
- if < 1 it is stable

gradient calculation requires multiplying each neuron's matrix

- if spectral norms are $< 1 \rightarrow$ total product goes to 0

gradient that is propogated will therefore either increase (if above 1) or decrease (if below 1) causing it to either **explode** or **vanish**

11.2.2 Solutions

delays and hierarchies

- learned time scales / hierarchies (Chung et al 2016)
- hierarchical multiscale RNNs (Chung et al 2017)

gating

- self-loop for gradients, learned gating
- eigenvalue of Jacobian only slightly less than 1
- LSTM (Hochreiter and Schmidhuber 1997)
- GRU (Cho et al 2014)

attention

- learn to weigh your whole input vector based on current state
- soft content-based attention (Bahdanau et al 2014)

11.2.3 Attention

attention networks

- graph attention networks (Velickovic 2018)
- attention for memory, NTM (Bahadanau et al 2014)
- pointing unknown words (Gulcehre 2016)

self-attention (Google 2017)

- encode location, transform location based on attention from others
- parallalelized
- self-attentive backtracking (Ke et al 2018)

11.2.4 Consciousness Prior

- we reason in latent space
- our representations have few, low-dimensional variables
- constrain by selecting relevant abstract concepts

two levels of representation: **consciousness prior**

- high-dependsional abstract representation
- low-dimensional conscious thought
- attention mechanism over both to create conscious thought
- objective function in abstract space, mutual information between past and future

12 Language Understanding (Graham Neubig)

12.1 Overview

- language modelling
- text classification
- sequence transduction
- structurual analysis
- semantic analysis

12.2 Sentence Classification

bag of words

- sum of word's value + softmax
- can't handle negations, double-negatives

deep continuous bag of words

- cbow + tanh activations
- effective but still can't handle sequential dependencies

bag of ngrams, CNNs

- ngrams: sliding window over sentence
- soft bag of ngrams + pooling: time delay NN (Waibel et al 1989) $\approx$ 1D conv
- great for short-distance feature extractions
- fail on long-distance dependencies

RNN

- recurrence allows for ”remembering”
- long term and sequential dependencies
- credit assignment is difficult

count-based models

- counting frequency and dividing

12.3 Language Models

word embedddings

- create a space for words that semantically represents them
- sharing strength and reducing parameter space

RNN LMs

- RNNs using word embeddings
- long terms dependencies and also smaller parameters space
- effective and standard method

evaluation

- log-likelihood
- per-word log-likelihood
- per-word cross entropy ($\log 2$)
- perplexity (e^{LL})

12.3.1 Conditioned Language Models

generate text according to specification

- condition RNN through hidden state

12.4 Generation Problem

sampling generate a random sentence

ancestral sampling generate words one-by-one

- **greedy search** pick word with highest prob
- **beam search** keep multiple hypotheses

argmax generate sentence with highest prob, e.g. sentence translation

- **attention** keep vector of context
- **self-attention**

12.5 Parsing and Tagging

linearized tree + seq2seq

- make your tree a string and "translate"
- simple but doesn't exploit structure

bi-LSTM CRF

- CRF layer ensure consistency between tags
- training and test with dynamic programming

stack LSTMs (Dyer et al 2015)

- 3 LSTMS: over words and over compositional representation
- gives better structural and semantic understanding, exploit structure

12.6 Small Data

pre-training on another task

- word embeddings (Fasttext)
- sentence representation (Dai et al 2015)
- contextual word embeddings (ELMo)

multi-task learning

- jointly training on your task and another

cross-domain

- learn over many languages simultaneously to help translating other
- fine-tuning on your specific language
- adding language tags

design models that scale, similar to our own linguistic understanding

12.7 Semantics

12.7.1 Semantic Parsing

executable representations

- parsing to SQL queries
- parsing to commands, compositional trees
- parsing to source code

semantic parsing with syntax

- express as abstract syntax tree (AST)
- works better than just seq2seq

12.7.2 Machine Reading

multiple-choice question tasks

- MCTest 500 passages, 2k questions
- RACE 28k passages, 100k questions

span selection

- SQuAD 500 passages, 100k questions
- triviaQA

necessary

- extract only salient parts
- do abstract reasoning on language

bidirectional language flow (Seo et al 2017)

- calculate doc2ctxt, ctxt2doc
- concatenate representations and extract

memory networks (Sukhbaatar et al 2015)

- multiple layers of attention to save/read info
- does ”reasoning”

12.8 Challenges

- scaling down to scarce data
- robust solutions in big data scenarios
- deeper reasoning
- interpretability

13 Multi-modal Learning (Jamie Kiros)

- encoding
- decoding
- ...

13.1 Building Blocks

matching learning a joint embedding space for retrieval

fusion combining representations of modalities

- simple
- gating
- bilinear pooling

FiLM condition one modality on another

- VQA (Perez et al 2017)
- RL with instructions (Chaplot et al 2017)

translation from one modality to another

13.2 Research

grounding learning one modality given correspondance in another

contextualization harness large corpus to learn contextualized word embeddings (Peters et al 2018)

- winograd schemas

multi-apt representations flexible modulation across contexts/tasks

- InferLite (Kiros and Chan 2018) multi-apt sentence embeddings

relevance realization zoning in on what's relevant

- dynamic representation, changing with relevance
- aligning books and movies (Kiros and Zhu 2015)

specificity certain images have more variability in their possible descriptions

- increase specificity through grounding, GuessWhat?! (De Vries et al 2017)
- Hierarchical Neural Story Generation (Fan et al 2018)

14 Generative Models for Music **(Sageev Oore)**

14.1 Music

14.1.1 Music Background

pitch perceptual concept

interval distance between pitches, not easily relative

consonance "niceness" not always good, not invariant

dynamics you can "weight" notes, changes overall sound

it is difficult to evaluate a melody, a good composer might make any melody work

14.1.2 Representations

MIDI sequential description of music data

- good for pitch
- not good for timbre

score what notes to play, interpreted by performer

spectrogram frequencies and their intensity

we can consider MIDI / score a sort of *language*

14.2 Modelling

14.2.1 Score

sequence tutor (jacques et al 2017)

- RNN that has to follow rules made by a Q-network

musical dialogue (Bretan et al 2017)

- call and response music
- music embeddings with an auto-encoder
- clustering + unit selection to create response

"flat" note VAE

- hierarchical decoder with biLSTM encoder

14.2.2 Performance

performance RNN (Simon and Oore 2017)

- train on single-instrument human performance
- generate directly in the performance space

relative self-attention for music (Huang et al 2018)

- using the transformer for music generation

14.2.3 Audio

musical timbre transfer (Huang et al 2018)

- cycleGAN to transform waveform
- transfer flute to violin