Ian goodfellow - DL - Representation Learn

15.1 Greedy layer-wise unsupervised Pretraining
15.2 transfer learning & Domain Application
15.3 Semi-supervised Disentrangle of causal factors
15.4 Distributed Representation
15.5 Exponential gains from Depth
15.6 Provide clue to Discover underlying causes

---

① First discuss what learned representation means

② how learned rep can be useful to design Deep architectures

Explore how Algos share statistical strength accross tasks

↳ e.g using unsupervised tasks to perform supervised tasks

(Assume this will mean using embeds as input to supervised)

Share reps are useful to:
① handle multiple domains
② transfer knowledge to tasks for which few/little example exist
↳ but a "task representation" exists

What makes a representation good?
Make a subsequent learning task easier

We can think of ~~fee~~ supervised feedforwards as a kind of representation learning

last layer = SOFTmax linear classifier

rest of the network learns to provide a representation to this layer/classifier

training leads to a representation @ each hidden layer

↳ takes on properties that make representation easier

in principle, the last layer does not need to be a linear classifier model

Could be a nearest neighbour

Penultimate layer will learn nice properties depending on the type of the last layer

unsupervised algos have a main training objective but learn representation as a side effect

Regardless of how representation was obtained, it can be used for another task

Representation learning provides a way to perform unsupervised & semi-supervised learning

Often we have very large amounts of unlabled data

↳ & little labled data

training on labled data often leads to overfitting

Semi-supervised offers way out by also learning from the unlabelled data

learn good respresenations for unlabelled

↳ use representations to solve the supervised task

Chapter hypothesis:

unlabled data can be used to learn good representation

15.1 Greedy layer-wise unsupervised pretraining

unsupervised learning gave ways to keep learning resurgence

Allowed for train of deep networks w/out architectural specializations
↳ such as convolution or recurrence

unsupervised pretraining = greedy layer wise

greedy layer wise relies on a single-layer representation algo
→ Single layer autoencoder

each layer is trained using unsupervised

output is used to train the next layer

layers are trained one-by-one

DL renaissance (2006) started w/ learning that greedy learning proc could be used to find a good initialization point
↳ instead of setting weights & bias randomly
↳ layers learn low level features
→ learned rep can be transferred & trained FROM

it is called "greedy" because it optimizes each peice of the solution independently

↳ instead of jointly optimizing, i.e. Backprop

"layer-wise" because each indep pieces is a layer of the network

trains k-th layer whilst keeping previous fixed

lower layers have no knowledge of later & are not retrained once introduced

"unsupervised" because each layer is trained w/ unsupervised representation learning

Pre-training → called this because it is only supposed to be the first step

↳ A joint training Algo is applied to fine-tune all layer together

for supervised learning, pre-training has 2 impacts:

(1) A regularized - Decrease test error (performance) w/out decreasing the training error

(2) form of parameter initialization

"Pretraining" commonly refers to the 2 step process as a whole

↳ Pretrain + Supervised learning

Supervised Part can be simple or complex

Simple → classifier on top of features learn in pre-train phase

Complex → Supervised fine-tune of entire network

15.1.1 when & why does pretraining work?

greedy layer-wise unsupervised pre-train often leads to substantial improvement in test error in classification

But it can sometime be harmful so important to understand when & why it works

Discussion is restricted to greedy-unsupervised
→ other semi-supvised paradigms exist

unsupervised pretraining combines two concepts

① initialization to improve regularization

② learning input distribution helps learn mapping from input to output

initalization

this point is the least understood

originally, people thought init point may allow a certain minima to be accessed

today, we know that nn's to not arrive @ a critical point in particular

it is possible that init allows the model to access areas that other-wise would be impossible

↳ i.e. surrounded by areas where the cost area varies so much the gradient are too messy to allow access

Sharing learned information

A algo picking up info learned in an unsupervised setting is better understood

Example, gen model for cars has some representation regarding wheels

not well understood @ a. many levels so hard to know which task will benefit from the/which unsupervised learning

From point of view, Pretrain as representation

Pretrain is more effective when initial representation is poor

ex. word embeds

one-hot encode is not good because any word is the same distance from each

learned word embeds encode similarity by their distance fm each

unsupervised pretrain is useful for words

it is less useful for words

↳ maybe because images already lie in rich vector space

↳ Distances already provide a local qual sim metric

words - hot-encode cat, car = similar

image - Pixels = number = tell similar

Frompoint of vew, regularzer/initalizaton

Most helpful when labled examplе are very small

↳ Also may extect unlabled to be very large

Other factors

unsupervised pretrain most likely to be useful when the funchon to be learned is extremely complicated

regularzes such as weight decay bias towards a simple funchon

rather unsuper leads towards feature func for the unsupervised task
→ which is complicated

if the true underlying funchons are complicated & shaped by input dist
↳ unsupervisel learn may be a more appropriate regularizer

unsup PT is most known to improve classifys & Reduce test set error

Erhan et al (2010) Experiments suggest pretrain takes paras into a region that would otherwise be inaccessible

NNs are non-deterministic & converge to a different function everytime it is run

training may halt where the gradient becomes to small
→ to prevent overfit

networkes that receive unsuper pretrain consistently halt in the same region

region where pretrained networks arrive is smaller
↳ suggests reduces the variance of the estimation process
↳ in turn, reduces risk of overfit

Pretraining inits nn paras into a region that they dont escape
↳ results are more consistent & less likely to be very bad

today, pre-train has largely been abandoned except in the field of NLP

Pretrain often done of extremely large unlabeled dataset

↳ learn good representation

↳ use rep. to fine-tune for a supervised task

Pioneered collobert & weston (2008b)
turian (2010)
Collobert (2011)

15.2 transfer learning & Domain Application

learned in one setting P_1
↳ Applied to another P_2

this generalizes the notion from the previous section
↳ unsupervised into supervised

Transfer learning the learner must perform two or more different task

Assumption Many of the factors capture from P_1 also Explain variations in P_2

example - Model for cats & dogs to use as start point for diff visual category
↳ Ants vs wasps
↳ cats & dogs has much more data

Many categories share low level characteristics
- notion of edges
- geometric changes
- lighting

transfer learning is best achieved via representation when:

- there exists features that are useful for different settings or tasks
  - ↳ corresponding to underlying factors that appear in more than one setting

task Specific Preprocessing (P. 535)

Diff layers in a network learning DIFF aspects

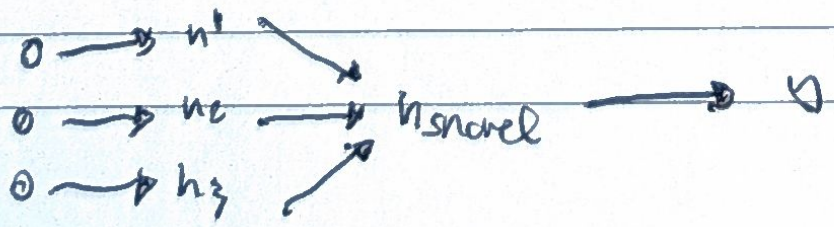
early layers may be very specific to the train data & task

later layers around the output layer are more general to the task @ large & may be more transferable

this why it is often better to do task-specific preprocessing that ~~that~~ lead into shareable layers

often used when the output task e.g. has a general semantic task meaning

but the inputs can be multiple and vary in meaning or even DIMENSION



Domain adaption

the task remains the same but input distribution is slightly different

Sentiment analysis - Pos or Neg

Sentiment predictor may be trained on a large corpus of text: reviews, books, videos, etc.

there will exist an underlying general function denoting sentiment

but vocab will likely vary greatly across domains

Steps:

for sentiment analysis, unsupervised pretrain (ie denoising auto-encoders) has been found to be very good regard Domain adaption

15.3 Semi-Supervised Disentangling of Causal Factors

What makes representation good?

one hypothesis of ideal representation =

- features learned in the representation
- correspond to the underlying causes of the observed data

↳ w/ distinct features learned corresponds to Diff causes

↳ i.e. the rep disentangles the causes

hypo motivates the approach
→ first seek good rep of $P(x)$

then rep many be good rep for comp $P(y|x)$

if y is promient cause of x

in other approaches, concered w/ reps that are easy to model.
→ e.g entries are sparse or independent

Sparse = most categories = 0 → zeros are comp Eff
& reduces noise & overfits

indep = simpler to model statistically not related
- info of one does not influence other
- models can be simpler & more EFF → think Bayes vs Naive Bayes
- Decomposition → Analyse each component seperately

Sparse / indep are easy to model —
- simple algos
- relationships are simpler
- mappings easy to model

Cleaning Seperating Causal Factors

these are true latent properties of the data

e.g: face = head pose, light, nose, eyes, age

Clean sep = each dimension refers directly to a unique specific causal factor

Issue: true causal factors are often intertwined — think nose & eyes shape

to distinctly unravel might require a very complex representation

Goal Rep

- goal of "sparse / indepenant" is to make learned rep simpler downstream
- "cleanly seperating causal factors" make representation interpretable

A Futher hypo says these two are linked

↳ once we have underlying representations for what we observe

we can begin to isolate attributes from other

e.g.

if h represents many cause of x

& y = most common output

easy to predict y from h

When does Semi-Super fail?

- learn $p(x)$ is not help to $P(y|x)$
- $P(x)$ uniformly dist
- want to learn $f(x) = E[y|x]$
- obs train x alone give no $P(y|x)$ info

When does Semi-Super Succeed?

consider x arises from A mixture

w/ mixtures corresponding to y values

if mixture components are well seperated ~~then~~

the model P(x) reveals where each component is

then, A single labelled example will be enough to perfectly learn ~~P(x,y)~~ P(y|x)

what could tie P(y|x) & P(x)?

if y is closely assoc w/ causal factors of x then P(x) & P(y|x) strongly tied

An unsupervised representation learning that tries to disentangle the underlying factors of variation is likely to be useful

as a semi-super learn strat.

15.4 Distributed Representations

15.5. Exponential Gains from Depth

Sec 6.4.1 MLP's are universal approximators

+ Some funcs can be repped by exponentially smaller deep networks

Decrease in model size leads to improved stat eff

this section describes how similar results apply to other models/architectures w/ distributed hidden representations

Example 15.4 = Radford(2015) gen model learned about explanatory factors underlying images of faces
→ based on deep net
→ Deep to learn abstract explan features

In AI tasks, factors that can be chosen almost indep. yet still correspond to meaningful inputs are more likely to be very high level & related in highly non-linear ways to the input

thus, this demands deep distributed reps where high level features are obtained through the composition of many non-linearities

Notes from gemini....

"Distributed Representation"

each concept or piece of information is represented by a pattern of activity across many individual units/DIMs

many units contribute to rep multiple concepts

or each unit contributes to many DIFF concepts

↳ A single neuron encodes info about various features

this is in contrast to local representation

Distributed Representation allows for semantic sim & generalization

↳
- think word-to-vec
- words w/ sim meaning are embedded closer together
- unlike local-hot-one encode, cat, car = close
- this allows models to generalize automatic
- if learnt ~~about~~ something about cats can partially ~~apply~~ tranfer knowledge to dogs
- Reduces amount of training data needs

Capacity

- local = not-one-enode = n = represent N distinct items
- w/ distributed representation can potentially do 2^N if each unit can take on continuous values

Summary of distributed Representation

uses compact, dense & continuous vector where info about a concept is spread accross multiple dimensions

this allows for:
- Automatic capture of semantic rels
- Superair generalization
- EFF use of parameters

Back to goodfellow...

Proved many times that comp organization through the composition of many non-linearities & hierachy of re reused features.

↳ can give expo boost to stat EFF

↳ on top of boost from Distributed rep

A Model that is a universal approx can Approx all cont functions upto any non-zero tolerance given enough hidden units

however hidden units may be very large

15.6 Providing clues to discover underlying causes

come back round to:
"what makes a representation good?"

prev introduced: ideal representation is one that disentagles the underlying causal factors of variation that generates the data

MOST strats for Rep learning are based on introducing clues that help find these underlying factors of variation

clues help learners seperate observed factors from others

Supervised learning - $\underset{\text{label}}{y}$ & x is the strongest clue

unlabeled data is more common/vast & ~~many~~ ~~Ma~~ rep learning makes use of more subtle clues about the underlying factors

these hints take the form of implicit prior beleifs that we the algo Designer impose to guide the learner

how is regularization a prior & cue?

L1 reg implicit expresses ~~that~~ a prior belief that many features are irrelevant

L2 = prior that func is smooth & large weights could create sensitivity

Dropout = belief that model should not rely on a single feature or neuron too heavily (robust)

architecture = some models has implicit beliefs. CNNs = translation invariance (feature vector regardless of location) & locality (pixels closer together)