Deep Learning for Programming

Prof. Wray Buntine

http://Bayesian-Models.org

Dept. of Data Sc. & AI, Monash University

College of Eng. & Comp. Sc., Vin University

2021-08-17

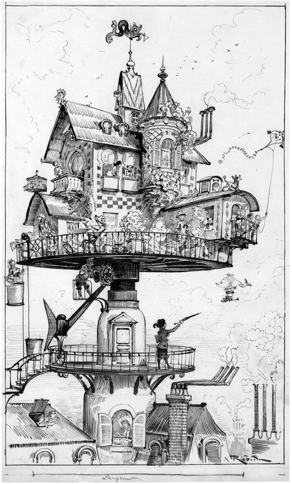
Or Thoughts ... From an Old Guy

Or Thoughts ... From an Old Guy

$\Longleftarrow$ ME

(before shaving and haircut)

Outline



Context

Old ML and Stats versus New ML

A Catalogue of Deep Learning Ideas

Conclusion

Context

from *MIT Technology Review*

Karen Hao, 03/11/2020

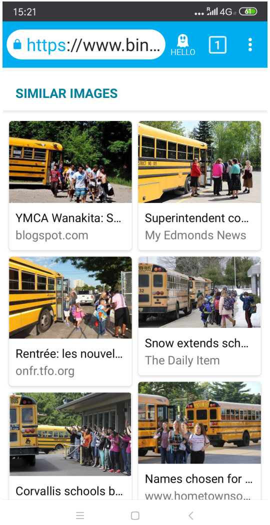
/ AP

Artificial intelligence / Machine learning

AI pioneer Geoff Hinton: “Deep learning is going to be able to do everything”

Information Retrieval: Images

image search gives



Information Retrieval Becomes ML

Information Retrieval (IR): from a single query (image, text), retrieve “best” matching documents. e.g. text/image search engine

Information Retrieval Becomes ML

Information Retrieval (IR): from a single query (image, text), retrieve "best" matching documents. e.g. text/image search engine

- this is like learning from just one example!
- called zero/one-shot learning in Deep Neural Networks, especially for images

Information Retrieval Becomes ML

Information Retrieval (IR): from a single query (image, text), retrieve "best" matching documents. e.g. text/image search engine

- this is like learning from just one example!
- called zero/one-shot learning in Deep Neural Networks, especially for images
- state-of-the-art in text IR is hybrid IR and transformer-based language models!
  - see SIGIR 2018 tutorial by Xu, He and Li
  - see QGenHyb by Ma, Korotkov, Yang, Hall, McDonald, EACL 2021
  - to my knowledge IR techniques and earlier one-shot researchers did not intersect!

Context, cont.

- Deep Learning successes:
  - machine translation, speech understanding
  - bio-informatics (e.g., 3-D protein folding)
  - object/person tracking in video

  } representation learning!

Context, cont.

- Deep Learning successes:
  - machine translation, speech understanding
  - bio-informatics (e.g., 3-D protein folding)
  - object/person tracking in video

  } representation learning!
- Machine Learning in the guise of Deep Learning is now happening at a world-wind pace:
  - a phase shift happening a few years ago,
  - huge teams making rapid advances,
  - results/methods already outdated and superceded at their point of publication,

Context, cont.

- Deep Learning successes:
  - machine translation, speech understanding
  - bio-informatics (e.g., 3-D protein folding)
  - object/person tracking in video

  } representation learning!
- Machine Learning in the guise of Deep Learning is now happening at a world-wind pace:
  - a phase shift happening a few years ago,
  - huge teams making rapid advances,
  - results/methods already outdated and superceded at their point of publication,
- How can we employ these techniques in traditional computer science?

Software Systems adding AI/ML

- source data needs quality management, versioning, etc. $\Longleftarrow$ the biggest problem
- AI systems components entangled in complex ways
- ML knowledge required of developers to harness the ML tools
- whole new testing and development frameworks needed
- explainability, fairness, transparency, fault tolerance, ...

Software Systems adding AI/ML

- source data needs quality management, versioning, etc. $\Longleftarrow$ the biggest problem
- AI systems components entangled in complex ways
- ML knowledge required of developers to harness the ML tools
- whole new testing and development frameworks needed
- explainability, fairness, transparency, fault tolerance, ...

30 years ago a similar problem in AI was faced when integrating/developing **expert systems** with traditional software:

- machine learning was proposed as the solution!
- even then the problem of data quality management was recognised

AI/ML Dev. Tasks

TABLE II

THE TOP-RANKED CHALLENGES AND PERSONAL EXPERIENCE WITH AI. RESPONDENTS WERE GROUPED INTO THREE BUCKETS (LOW, MEDIUM, HIGH) BASED ON THE 33RD AND 67TH PERCENTILE OF THE NUMBER OF YEARS OF AI EXPERIENCE THEY PERSONALLY HAD (N=308). THE COLUMN *Frequency* SHOWS THE INCREASE/DECREASE OF THE FREQUENCY IN THE MEDIUM AND HIGH BUCKETS COMPARED TO THE LOW BUCKETS. THE COLUMN *Rank* SHOWS THE RANKING OF THE CHALLENGES WITHIN EACH EXPERIENCE BUCKET, WITH 1 BEING THE MOST FREQUENT CHALLENGE.

	Frequency			Rank		
					Experience	
Challenge	Medium vs. Low	High vs. Low	Trend	Low	Medium	High
Data Availability, Collection, Cleaning, and Management	-2%	60%		1	1	1
Education and Training	-69%	-78%		1	5	9
Hardware Resources	-32%	13%		3	8	6
End-to-end pipeline support	65%	41%		4	2	4
Collaboration and working culture	19%	69%		5	6	6
Specification	2%	50%		5	8	8
Integrating AI into larger systems	-49%	-62%		5	16	13
Education: Guidance and Mentoring	-83%	-81%		5	21	18
AI Tools	144%	193%		9	3	2
Scale	154%	210%		10	4	3
Model Evolution, Evaluation, and Deployment	137%	276%		15	6	4

from "Software Engineering for Machine Learning: A Case Study," Amershi et al., ICSE 2019

AI/ML Dev. Pipeline

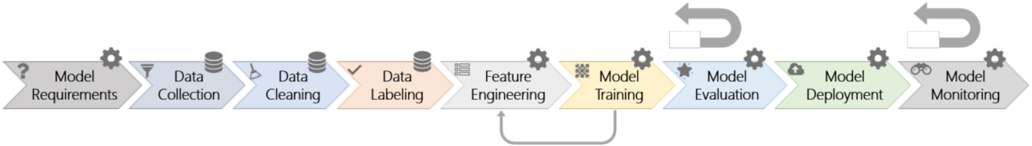


Fig. 1. The nine stages of the machine learning workflow. Some stages are data-oriented (e.g., collection, cleaning, and labeling) and others are model-oriented (e.g., model requirements, feature engineering, training, evaluation, deployment, and monitoring). There are many feedback loops in the workflow. The larger feedback arrows denote that model evaluation and monitoring may loop back to any of the previous stages. The smaller feedback arrow illustrates that model training may loop back to feature engineering (e.g., in representation learning).

from "Software Engineering for Machine Learning: A Case Study," Amershi et al., ICSE 2019

different to traditional agile or other!

AI/ML Dev. Pipeline

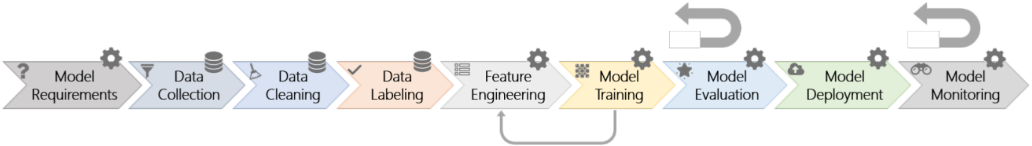


Fig. 1. The nine stages of the machine learning workflow. Some stages are data-oriented (e.g., collection, cleaning, and labeling) and others are model-oriented (e.g., model requirements, feature engineering, training, evaluation, deployment, and monitoring). There are many feedback loops in the workflow. The larger feedback arrows denote that model evaluation and monitoring may loop back to any of the previous stages. The smaller feedback arrow illustrates that model training may loop back to feature engineering (e.g., in representation learning).

from "Software Engineering for Machine Learning: A Case Study," Amershi et al., ICSE 2019

different to traditional agile or other!

near identical to the data science dev. pipeline from 2013

and they've had huge trouble integrating into the career/HR environment

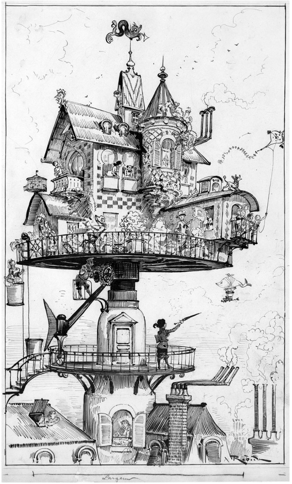
Programming Languages for AI/ML

- support for building & testing ML models
- libraries for:
  - streaming data
  - hyper-parameter optimisation
  - probability components
  - matrices and tensors
  - auto-differentiation
  - model components (convolutions, transformers, etc.)
- probabilistic programming

AI/ML for Programming Languages

- debugging, testing and code reviews
- comments checking and generation
- code search and matching
- auto-generating software
- smart compilation
- auto-generating distributed and/or low-level code

Outline



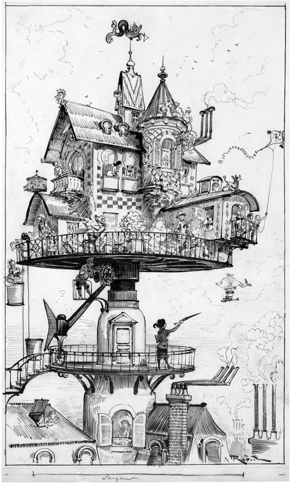
Context

Old ML and Stats versus New ML

A Catalogue of Deep Learning Ideas

Conclusion

Outline



What did We Learn from Old ML?

- how to do exponential family, linear and partitioning models well

  e.g., XGBoost, online LDA, Gaussian mixtures, logistic regression

  - plus how to augment them with fancy mathematical tricks

    e.g., kernels, non-parametric Bayesian

    i.e., infinite vectors

  i.e., cases where learning admits computational simplifications

What did We Learn from Old ML?

- how to do exponential family, linear and partitioning models well

  e.g., XGBoost, online LDA, Gaussian mixtures, logistic regression

  - plus how to augment them with fancy mathematical tricks

    e.g., kernels, non-parametric Bayesian

    i.e., infinite vectors

  i.e., cases where learning admits computational simplifications

- models as first-class objects

  e.g., Bayesian & Markov networks, neural networks

What did We Learn from Old ML?

- how to do exponential family, linear and partitioning models well

  e.g., XGBoost, online LDA, Gaussian mixtures, logistic regression

  - plus how to augment them with fancy mathematical tricks

    e.g., kernels, non-parametric Bayesian

    i.e., infinite vectors

  i.e., cases where learning admits computational simplifications

- models as first-class objects

  e.g., Bayesian & Markov networks, neural networks

- statistical ML theory

  e.g., bias-variance tradeoff, variational methods, ensembles

  e.g., Bayesian MCMC as "gold standard" learning

What did We Learn, cont.?

- cost functions (using statistical ML theory)

  e.g., regularisation, metrics and divergences,

  i.e., the theory of objective functions for deep NNs

What did We Learn, cont.?

- cost functions (using statistical ML theory)

  e.g., regularisation, metrics and divergences,

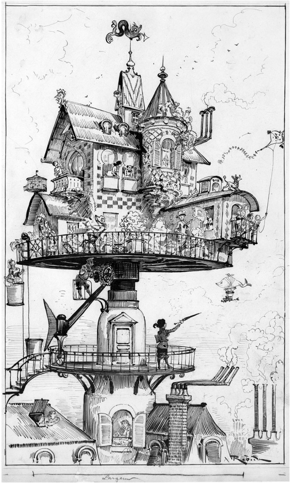
  i.e., the theory of objective functions for deep NNs

- paradigms

  e.g., online learning, active learning, transfer learning

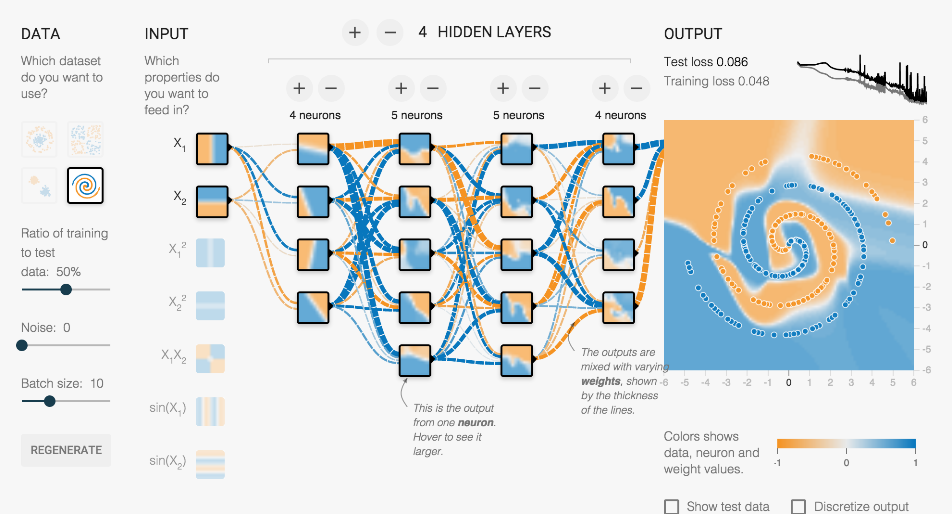
- learning on networks
  - deterministic networks
  - probabilistic networks

Outline

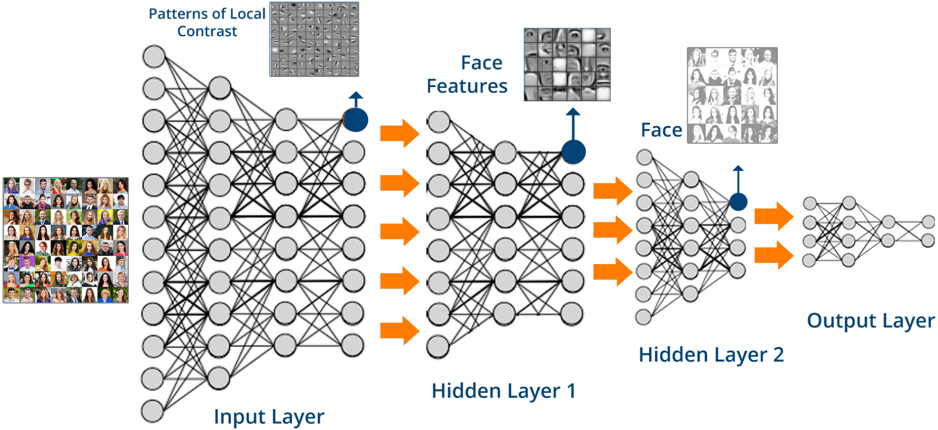


A Non-trivial Network

see playground.tensorflow.org



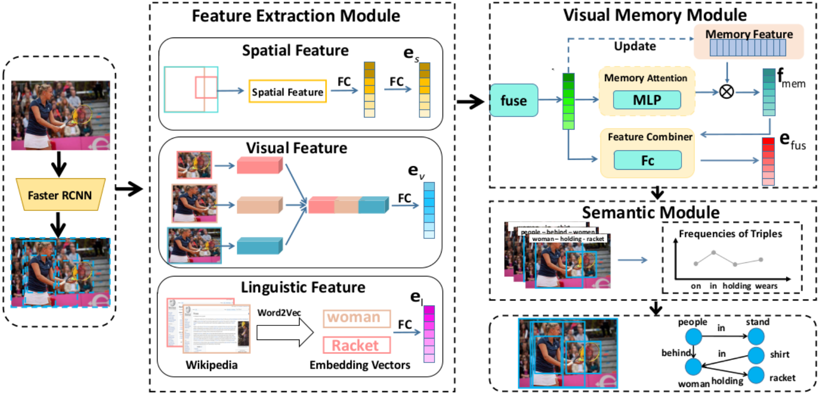
Deep Representations



from https://thedatascientist.com/what-deep-learning-is-and-isnt/

Observation: different layers of the network “learn” alternative representations.

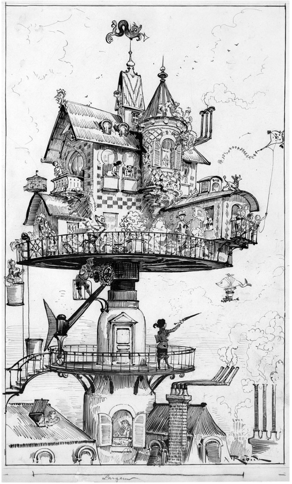
Deep Architechures



from "Memory-Based Network for Scene Graph with Unbalanced Relations", MM'20, Wang et al.

Complex systems are built from neural modules.

Outline



Context

Old ML and Stats versus New ML

- Old ML
- Deep ML Reminders
- New (Deep) ML

A Catalogue of Deep Learning Ideas

Conclusion

How Does Deep Learning Innovate?

- Model/Spec driven black-box algorithms ease the workload of developers.
  - machine learning without statistics!

How Does Deep Learning Innovate?

- Model/Spec driven black-box algorithms ease the workload of developers.
  - machine learning without statistics!
- Porting down to GPUs or multi-core allows real speed.

How Does Deep Learning Innovate?

- Model/Spec driven black-box algorithms ease the workload of developers.
  - machine learning without statistics!
- Porting down to GPUs or multi-core allows real speed.
- Learning representations and discovering higher order concepts.
  - convolutions, structures, sequences, ...
  - high capacity makes them very flexible in fitting and does implicit parallel search

How Does Deep Learning Innovate?

- Model/Spec driven black-box algorithms ease the workload of developers.
  - machine learning without statistics!
- Porting down to GPUs or multi-core allows real speed.
- Learning representations and discovering higher order concepts.
  - convolutions, structures, sequences, ...
  - high capacity makes them very flexible in fitting and does implicit parallel search
- self-supervision

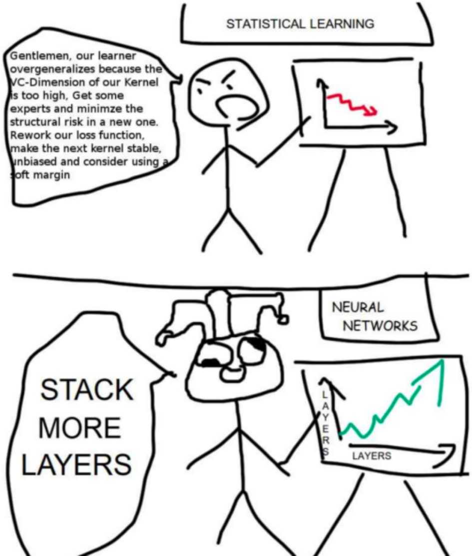
  i.e., pre-training networks with fabricated but useful tasks

How Does Deep Learning Innovate?

- Model/Spec driven black-box algorithms ease the workload of developers.
  - machine learning without statistics!
- Porting down to GPUs or multi-core allows real speed.
- Learning representations and discovering higher order concepts.
  - convolutions, structures, sequences, ...
  - high capacity makes them very flexible in fitting and does implicit parallel search
- self-supervision

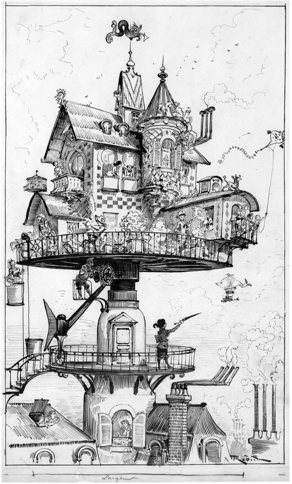
  i.e., pre-training networks with fabricated but useful tasks
- Allows "modelling in the large":
  - multi-task learning, imitation learning, meta-learning
  - components like convolutions, structures, sequences, ...

Why is Deep Neural Nets Successful?

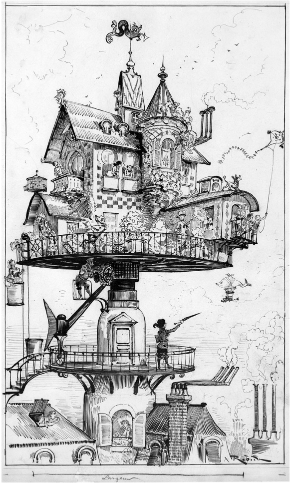


from Stanford Uni. NLP course, CS224N/Ling284 2020

Outline



Outline



Context

Old ML and Stats versus New ML

A Catalogue of Deep Learning Ideas

- Ensembling
- Data Augmentation
- Self-Supervision
- Probabilistic Programming

Ensembling

- Have set of (x, y) data Data and network parameters $\theta \in \Theta$.
- The Bayesian posterior, predicting y given new test point x:

$$\int_{\theta \in \Theta} \underbrace{p(y \,|\, x, \theta)}_{\text{prediction using parameters } \theta} \cdot \underbrace{p(\theta | \text{Data})}_{\text{posterior on parameters } \theta} \, \mathrm{d}\theta$$

- The gold standard algorithm is complex MCMC
- The simple ensemble says to train a small set of "good but different" models $\hat{\Theta}$ and pool them together

$$\frac{1}{|\hat{\Theta}|} \sum_{\theta \in \hat{\Theta}} p(y \,|\, x, \theta)$$

Ensembling

- Have set of (x, y) data Data and network parameters $\theta \in \Theta$.
- The Bayesian posterior, predicting y given new test point x:

$$\int_{\theta \in \Theta} \underbrace{p(y \,|\, x, \theta)}_{\text{prediction using parameters } \theta} \cdot \underbrace{p(\theta | \text{Data})}_{\text{posterior on parameters } \theta} \, d\theta$$

- The gold standard algorithm is complex MCMC
- The simple ensemble says to train a small set of "good but different" models $\hat{\Theta}$ and pool them together

$$\frac{1}{|\hat{\Theta}|} \sum_{\theta \in \hat{\Theta}} p(y \,|\, x, \theta)$$

- First did as a student's masters project at University of Sydney in 1988.

The (Frequentist) Laws of Learning

- The First Law of Learning for model family H (Geman & Geman, 1992)

$$\text{Mean-Squared-Error}(H) = \text{Bias(H)}^2 + \text{Variance(H)} + \text{IrreducibleError}$$

The (Frequentist) Laws of Learning

- The First Law of Learning for model family H (Geman & Geman, 1992)

$$\text{Mean-Squared-Error}(H) = \text{Bias(H)}^2 + \text{Variance(H)} + \text{IrreducibleError}$$

- The Second Law of Learning for ensemble $\mathcal{H}$ of models (Uedo & Nakano, 1996):

$$\begin{aligned}\text{Mean-Squared-Error}(\mathcal{H}) &= \overline{\text{Bias}}(\mathcal{H})^2 + \frac{1}{|\mathcal{H}|}\overline{\text{Variance}}(\mathcal{H}) + \text{IrreducibleError} \\ &\quad + \left(1 - \frac{1}{|\mathcal{H}|}\right)\text{Covariance}(\mathcal{H})\end{aligned}$$

The (Frequentist) Laws of Learning

- The First Law of Learning for model family H (Geman & Geman, 1992)

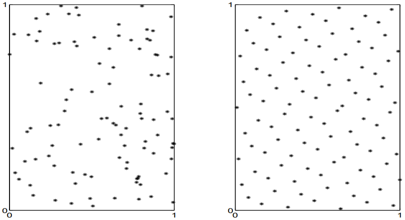
$$\text{Mean-Squared-Error}(H) = \text{Bias(H)}^2 + \text{Variance(H)} + \text{IrreducibleError}$$

- The Second Law of Learning for ensemble $\mathcal{H}$ of models (Uedo & Nakano, 1996):

$$\begin{aligned}\text{Mean-Squared-Error}(\mathcal{H}) &= \overline{\text{Bias}}(\mathcal{H})^2 + \frac{1}{|\mathcal{H}|}\overline{\text{Variance}}(\mathcal{H}) + \text{IrreducibleError} \\ &+ \left(1 - \frac{1}{|\mathcal{H}|}\right)\text{Covariance}(\mathcal{H})\end{aligned}$$

- Therefore:
  - elements of the ensemble should have lower bias and variance
  - elements of the ensemble should be de-correlated

Understanding the Second Law



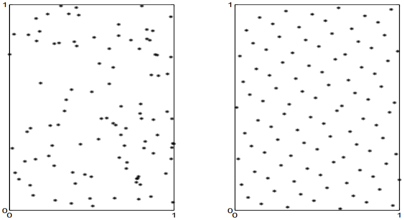
random versus quasi-random

- estimating $\int_{\theta\in\Theta} p(y \mid x, \theta) \cdot p(\theta|\text{Data})\, d\theta$
- the random points clump "statistically"
- the quasi-random points are de-correlated with little clumping

  e.g., determinantal point processes (like some adversarial training)

  e.g., Stein variational gradient descent

Understanding the Second Law



random versus quasi-random

- estimating $\int_{\theta \in \Theta} p(y \mid x, \theta) \cdot p(\theta | \text{Data})\, d\theta$
- the random points clump "statistically"
- the quasi-random points are de-correlated with little clumping

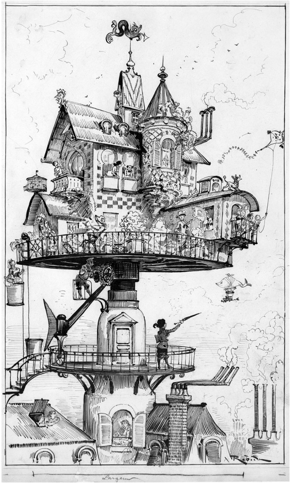
  e.g., determinantal point processes

  (like some adversarial training)

  e.g., Stein variational gradient descent

- ensemble $\sum_{\theta \in \hat{\Theta}} p(y \mid x, \theta)$ works better using quasi-random points

Outline



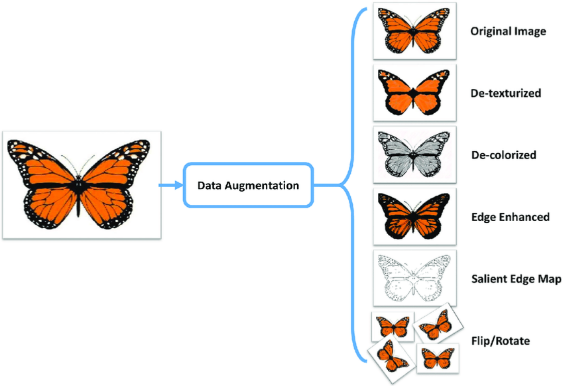
Context

Old ML and Stats versus New ML

A Catalogue of Deep Learning Ideas

- Ensembling
- Data Augmentation
- Self-Supervision
- Probabilistic Programming

Data Augmentation



from Ahmad, Muhammad and Baik, PLoS ONE 2017

IDEA: classified data is hard to get, so lets generate new data!

Data Augmentation: MNIST

- technique first used in Zipcode recognition for the US Post
- images can be rotated, shifted, thinned, etc.

Data Augmentation: Theory

- We need to identify an invariant in our data we want to hold.

e.g. For text, we could replace synonyms, back-translate, etc.

- We need an algorithm to apply changes to the data reflecting the invariant.
- Probabilistic model, for the augmentation distribution Aug

$$p(y_i \mid x_i, \theta) \qquad \text{standard data likelihood}$$

$$p(y_i \mid x_i, \theta, \mathrm{Aug}) = \int_{x'} p(y_i \mid x_i', \theta) p(x' | x_i, \mathrm{Aug}) \, \mathrm{d}x' \qquad \text{augmented data likelihood}$$

(the augmentation acts like a convolution)

Data Augmentation: Theory

- We need to identify an invariant in our data we want to hold.

e.g. For text, we could replace synonyms, back-translate, etc.

- We need an algorithm to apply changes to the data reflecting the invariant.
- Probabilistic model, for the augmentation distribution Aug

$$p(y_i \mid x_i, \theta) \qquad \text{standard data likelihood}$$

$$p(y_i \mid x_i, \theta, \mathrm{Aug}) = \int_{x'} p(y_i \mid x_i', \theta) p(x' | x_i, \mathrm{Aug})\, \mathrm{d}x' \qquad \text{augmented data likelihood}$$

(the augmentation acts like a convolution)

- Developed as vicinal risk minimisation by Chapelle *et al.* 2001.

Data Augmentation: Implementation

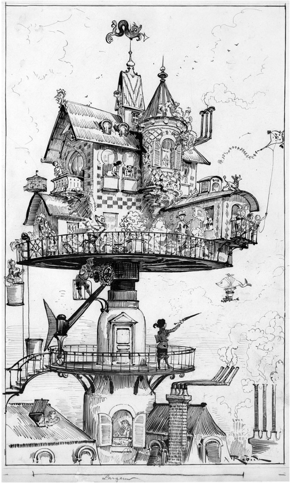
- Probabilistic model, for the augmentation distribution Aug

$$\begin{aligned} p(y_i \,|\, x_i, \theta, \mathrm{Aug}) &= \int_{x'} p(y_i \,|\, x_i', \theta) p(x' | x_i, \mathrm{Aug}) \, \mathrm{d}x' && \text{augmented data likelihood} \\ &\approx \sum_{x' \in E(x_i)} p(y_i \,|\, x_i', \theta) && \text{ensembled approximation} \end{aligned}$$

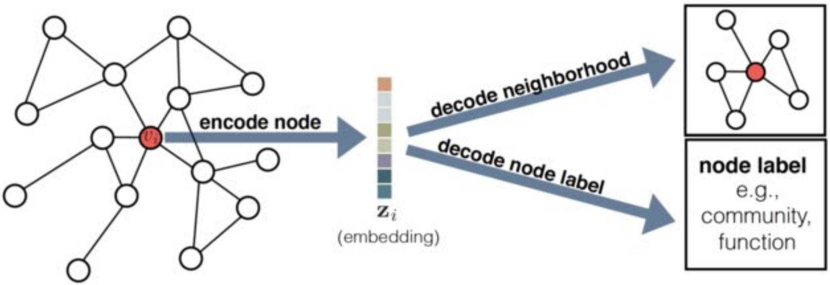
  using ensemble of augmentations $E(x_i)$ generated with $p(x'|x_i, \mathrm{Aug})$

- At learning time we train stochastically with one or more augmentations.
- At inference time we should also do augmentations.

Outline



Encoding, Decoding and Embeddings



from "Representation Learning on Graphs: Methods and Applications", Hamilton, Ying and Leskovec, 2017

Our operational framework: every piece of input structure has two embeddings, contextual and non-contextual.

Self-supervision

- to do pre-training, we need a task for learning
- embedding methods (CBOW, Word2Vec, etc.) are an early precursor
- pre-training should build lower-level features useful for subsequent target classes

Self-supervision

- to do pre-training, we need a task for learning
- embedding methods (CBOW, Word2Vec, etc.) are an early precursor
- pre-training should build lower-level features useful for subsequent target classes

Self-supervision: an artificial task created for the purposes of learning a network useful as a pre-trained network.

- called "self" supervised since the task is created automatically

Self-supervision, cont.

Self-supervision: an artificial task created for pre-training network.

- examples for image recognition:
  - color a B/W image
  - fill-in missing patches ("image in-painting")
  - object classification from very broad image class
- examples for text classification:
  - predict missing words
  - forwards and backwards order

Self-supervision, cont.

Self-supervision: an artificial task created for pre-training network.

- examples for image recognition:
  - color a B/W image
  - fill-in missing patches ("image in-painting")
  - object classification from very broad image class
- examples for text classification:
  - predict missing words
  - forwards and backwards order
- generally, the self-supervision task should be (1) richer and more refined than or (2) similar to subsequent target tasks

Pseudo-likelihood and Self-supervision

A pseudo-likelihood (Besag 1975) is an approximation to the joint probability distribution of a collection of random variables using univariate conditionals:

$$
\begin{aligned}
p(X|M,\boldsymbol{\Theta}) &= \prod_{i=1}^{I} p(x_i \,|\, x_1, \ldots, x_{i-1}, M, \Theta) && \text{(product rule of probability)} \\
&= p(x_1 \,|\, M, \Theta) p(x_2 \,|\, x_1, M, \Theta) \cdots \underbrace{p(x_I \,|\, x_1, \ldots, x_{I-1}, M, \Theta)}_{\text{repeat this term}}
\end{aligned}
$$

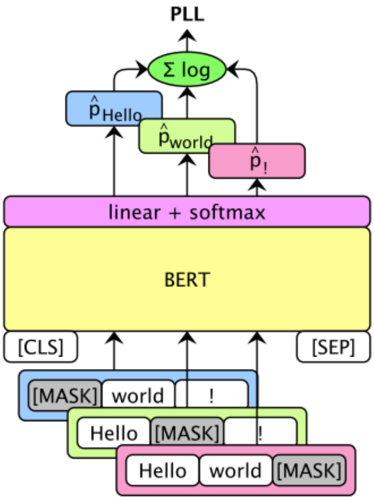
$$
\tilde{p}(X|M,\Theta) \equiv \prod_{i=1}^{I} \underbrace{p(x_i \,|\, X_{-i}, M, \Theta)}_{\text{each term is a single prediction}} \qquad \text{where } X_{-i} = X \setminus \{x_i\}
$$

Pseudo-likelihood, cont.

$$\tilde{p}(X|M,\Theta) \equiv \prod_{i=1}^{I} p(x_i \,|\, X_{-i}, M, \Theta) \qquad \text{where } X_{-i} = X \setminus \{x_i\}$$

- it is easily computed because the individual conditionals can usually be easily marginalised.
- maximising pseudo-likelihood is known to be consistent with maximising likelihood in the limit of infinite data.
- Pseudo-likelihood can be viewed as a simplified theoretical view of self-supervision.
- But an alternative view is of representation learning.

Self-supervision with Natural Language



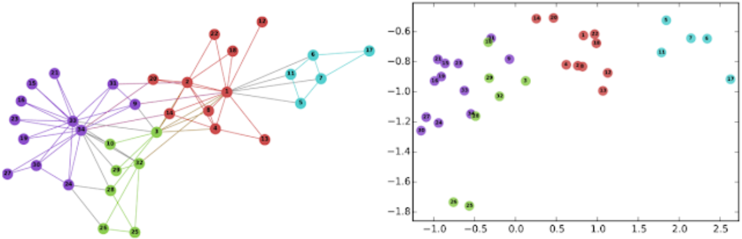
- stochastic pseudo-likelihood training on

$$PLL(W;\theta) = \sum_{t=1}^{|W|} \log p(w_t | W_{-t}, \theta)$$

- may add other prediction objectives about sentence ordering, etc.
- when the model is multi-layer transformers, is the basis of the biggest revolution in NLP history (BERT, XLNET, ...)

figure from Salazar, Liang et al., ACL 2020

Graph Neural Networks



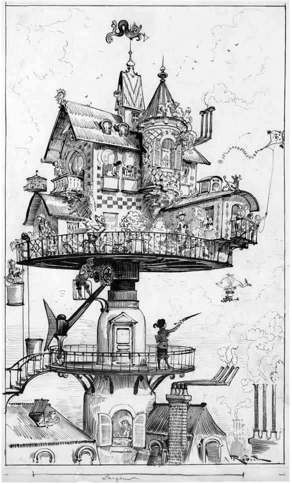
from "Representation Learning on Graphs: Methods and Applications", Hamilton, Ying and Leskovec, 2017

- techniques similar to word embeddings have been developed for graphs
- nodes in graphs also have extensive side information
  - "hospital patient" node in a graph may have their socio-economic data attached
- pseudo-likelihood style self-supervision applies equally as well

Summary

- Arbitary graph structures with embedded text can be modelled using this framework.
- Self-supervision works but:
  - pays to be clever about the pseudo-likelihood prediction tasks
- These self-supervision encoder-decoder networks form ideal pre-trained networks for subsequent prediction tasks.
  - can be done for written code and comments

Outline



Context

Old ML and Stats versus New ML

A Catalogue of Deep Learning Ideas

- Ensembling
- Data Augmentation
- Self-Supervision
- Probabilistic Programming

Bayes infer. Using Gibbs Sampling

BUGS, Spiegelhalter, Thomas, Best, Gilks, 1996

Modelling language:

```
model{
  # model priors
  beta0 ~ dnorm(0, 0.001)
  eta1 ~ dnorm(0, 0.001)
  tau ~ dgamma(0.1, 0.1)
  sigma <- 1/sqrt(tau)
  # data model, linear regression
  for( i in 1:n) {
     mu[i] <- beta0+ beta1*x[i]
     y[i] ~ dnorm(mu[i] , tau)
 }
}
```

Bayes infer. Using Gibbs Sampling

BUGS, Spiegelhalter, Thomas, Best, Gilks, 1996

Modelling language:

```
model{
  # model priors
  beta0 ~ dnorm(0, 0.001)
  eta1 ~ dnorm(0, 0.001)
  tau ~ dgamma(0.1, 0.1)
  sigma <- 1/sqrt(tau)
  # data model, linear regression
  for( i in 1:n) {
     mu[i] <- beta0+ beta1*x[i]
     y[i] ~ dnorm(mu[i] , tau)
 }
}
```

- Simple Bayesian linear regression using Gaussian model $\vec{x} = \beta_0 + \beta_1\vec{y}$.
- All constants, parameters and data are defined in the language.

Probabilistic Programming

- used to define probability models for automatic algorithm generation
- provide statistical constructs that users can directly call and use
- predominantly declarative - focus on what needs to be done over how
- can provide support for a plethora of operations
  - Gibbs, Hamiltonian Monte Carlo and other Gibbs
  - variational algorithms
  - black-box optimisation

this section largely drawn from PhD thesis of Dr. Sachith Seneviratne

Probabilistic Programming: Examples

BUGS uses a directed acyclic graph (DAG) to represent a model (1995)

AutoBayes uses Schema to compose models (1998)

Stan uses Hamiltonian Monte Carlo and targets continuous sampling (2012)

Edward CPU/GPU support with various inference schemes (2016)

Google TensorFlow Probability scale performance across CPUs, GPUs, and TPU, within `TensorFlow`

Probabilistic Programming: Examples

BUGS uses a directed acyclic graph (DAG) to represent a model (1995)

AutoBayes uses Schema to compose models (1998)

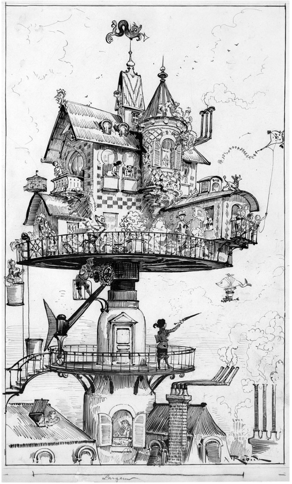
Stan uses Hamiltonian Monte Carlo and targets continuous sampling (2012)

Edward CPU/GPU support with various inference schemes (2016)

Google TensorFlow Probability scale performance across CPUs, GPUs, and TPU, within `TensorFlow`

Initially developed independently of Deep Learning, but has similar goals, and similar techniques, and there is now convergence.

Outline



Context

Old ML and Stats versus New ML

A Catalogue of Deep Learning Ideas

Conclusion

Example Uses

- Model/Spec driven black-box algorithms ease the workload of developers.
  - machine learning without statistics!
- Porting down to GPUs or multi-core allows real speed.
- Learning representations and discovering higher order concepts.
  - convolutions, structures, sequences, ...
  - high capacity makes them very flexible in fitting and does implicit parallel search
- self-supervision

  i.e., pre-training networks with fabricated but useful tasks
- Allows "modelling in the large":
  - multi-task learning, imitation learning, meta-learning
  - components like convolutions, structures, sequences, ...

→ using variants

→ pretraining on code

Code is Different to Natural Language

- NLP is “natural” and extremely hard to parse, with new or erroneous tokens appearing
  - majority of deep nets treat it like a sequence, not a structure
- code is easy to parse with precise structures, except for
  - the language buried in comments,
  - and variable names
- we even have auxiliary information like modification time, author, etc.

code has rich structures and auxiliary information as well as embedded text

Pre-training for Code

- use graphs to represent structure of code, model with graph NNs
- harness standard language models for the comments
- harness standard tokenisation models for variable names (`WordPiece`, etc.)
- develop clever self-supervision tasks:
  - predict comments
  - predict control flow
  - predict expressions

Variants in Probabilistic Programming

Sachith Seneviratne's PhD thesis, 2020

- high level code generated from models
- variant algorithms (part Gibbs, part variational, different probability formulations)
- variant implementations (different parts in multi-core, GPU)
- automated testing harness to "race" techniques
  - ala "Using machine learning to focus iterative optimization," Agakov, Bonilla, *et al.*, *Int. Symp. on Code Gen. and Optim.*, 2006

Variants in Probabilistic Programming

Sachith Seneviratne's PhD thesis, 2020

- high level code generated from models
- variant algorithms (part Gibbs, part variational, different probability formulations)
- variant implementations (different parts in multi-core, GPU)
- automated testing harness to "race" techniques
  - ala "Using machine learning to focus iterative optimization," Agakov, Bonilla, *et al.*, *Int. Symp. on Code Gen. and Optim.*, 2006

Needs the code generation and automated testing infrastructure to make work.

Another Application

TODAY'S HIGHLIGHTS

Why Full-time Programmers Are Decreasing Faster Than Ever

My guess is the number will become half after 10 years

Entreprogrammer in Level Up Coding · 4 min read ★

20 Machine Learning Projects That Will Get You Hired in 2021

The AI and Machine Learning industry is booming like never before. As of 2021, the increase in AI usage…

Khushbu Shah in ProjectPro · 17 min read

4 Pandas Functions That I Wish I Knew Earlier

You don't need to reinvent the wheel.

Yong Cui in Towards Data Science · 4 min read ★

The 3 Worst Pieces of Programming Advice I've Ever Heard

3. Build as quickly as possible

generating clickbait for developers

Another Application

TODAY'S HIGHLIGHTS

Why Full-time Programmers Are Decreasing Faster Than Ever

My guess is the number will become half after 10 years

Entreprogrammer in Level Up Coding · 4 min read ★

20 Machine Learning Projects That Will Get You Hired in 2021

The AI and Machine Learning industry is booming like never before. As of 2021, the increase in AI usage…

Khushbu Shah in ProjectPro · 17 min read

4 Pandas Functions That I Wish I Knew Earlier

You don't need to reinvent the wheel.

Yong Cui in Towards Data Science · 4 min read ★

The 3 Worst Pieces of Programming Advice I've Ever Heard

3. Build as quickly as possible

generating clickbait for developers

who can tell me where this image is grabbed from?

Questions?

Figure source: http://milewalk.com/mwblog/4-worst-job-interview-questions/