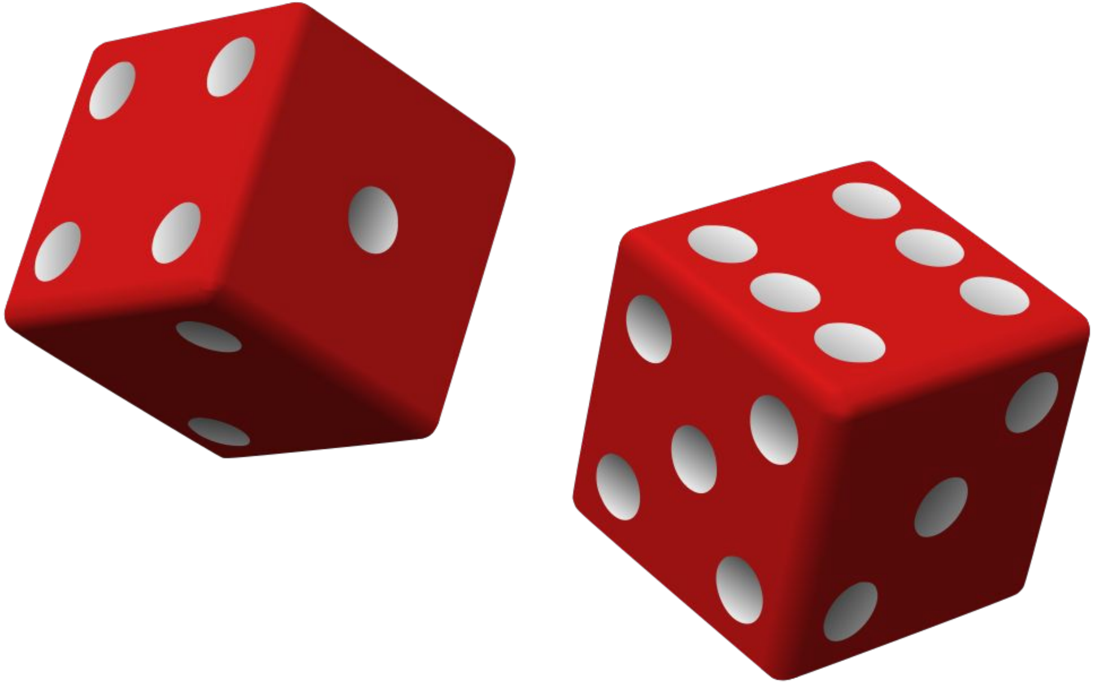
yields

Quantifying Model Uncertainty with AI

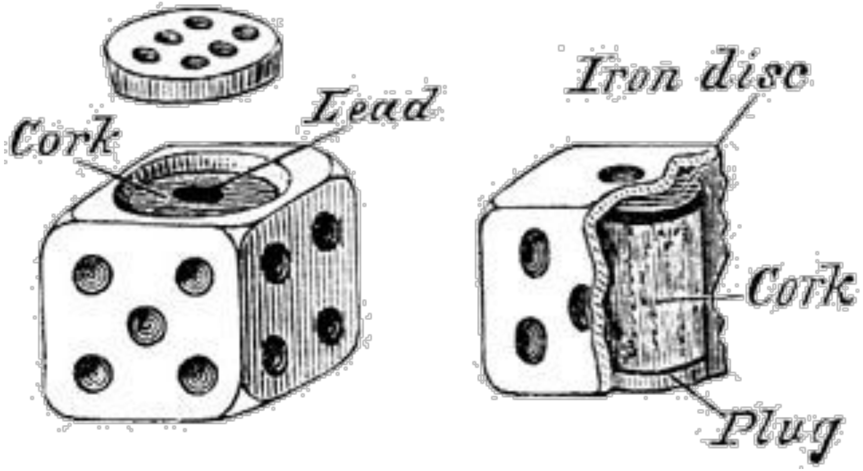
Jos Gheerardyn, CEO Yields.io, Nov 2020

yields

Why we simplify models

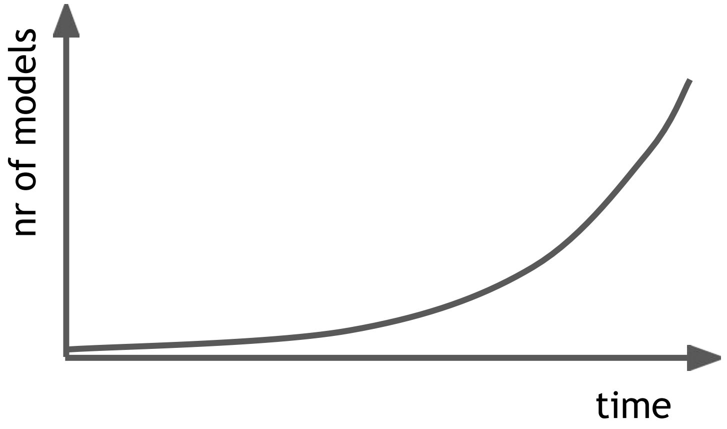


Why we validate models



The need for quantification

The number of models in financial institutions increases with 10 – 20 % yearly*



- 100 - 3000 models
- Median duration of a single model validation is >4 weeks**
  - **Tiering** (quant. & qual.)
    - materiality
    - risk exposure
    - regulatory impact
  - **Qualitative assessments** are fairly stable over time
  - **Quantitative assessments** can change quickly and allow for accurate risk management procedures
-

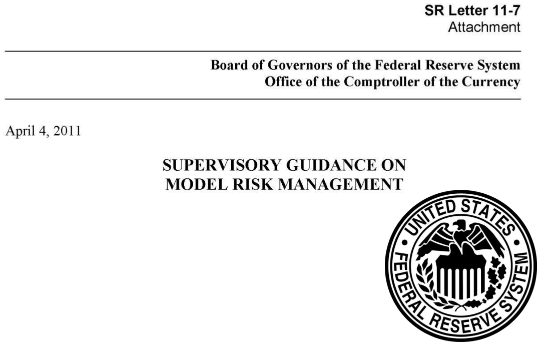
* See http://www.mckinsey.com/business-functions/risk/our-insights/the-evolution-of-model-risk-management

** See https://www2.deloitte.com/content/dam/Deloitte/dk/Documents/financial-services/deloitte-nl-global-model-practice-survey.pdf

Structure of a model validation

The highlights:

- Model dependencies
- Data
  Quality, representativeness, preprocessing, controls
- Framework and assumptions
- Model design and performance testing
  Model selection, backtesting, benchmarking, sensitivity testing, model uncertainty
- Model monitoring
- Limitations



SR Letter 11-7
Attachment

Board of Governors of the Federal Reserve System
Office of the Comptroller of the Currency

April 4, 2011

SUPERVISORY GUIDANCE ON
MODEL RISK MANAGEMENT

Policy Statement | PS7/18

Model risk management principles for stress testing

April 2018

BANK OF ENGLAND
PRUDENTIAL REGULATION
AUTHORITY

Where can ML help?

Data

- Detecting quality issues
- Verifying representativeness
- Determining unstable model behavior

Model design and performance testing

- benchmarking
- sensitivity analysis
- scenario generation
- model uncertainty

Model monitoring

- comparison with benchmarks and/or surrogates
- automated detection of issues

Different types of uncertainty

Definition*

- X is the quantity of interest we want to model
- x_i are states that are possible outcomes of X
- P is the model
- P is the set of available models

Risk: We know the probability of each outcome x_i

Uncertainty: We do not know the probability of each outcome x_i

- Model risk: Probability measure on P
- Model uncertainty: We do now know the probabilities on P

More precisely: **Model ambiguity** means several specifications for probabilities on P**

* See Knight, F. (1921) Risk, uncertainty and profit, Boston: Houghton Mifflin.
** See Epstein, L.G. (1999) A definition of uncertainty aversion, Review of Economic Studies, 65, 579-608.

Two paradigms: Model averaging vs worst-case*

Bayesian model averaging

- Prior on model parameters p(θ$_i$|P$_i$)
- Prior weights on models p(P$_i$)

Posterior probability on model P$_i$

$$p(P_i \mid x) = \frac{p(x \mid P_i)p(P_i)}{\sum_i p(x \mid P_i)p(P_i)}$$

With the likelihood of the observed data under P$_i$ being $p(x \mid P_i) = \int_{E_i} p(x \mid \vartheta_i, P_i)p(\vartheta_i \mid P_i)d\vartheta_i$

Computing model-dependent quantities via $E[f(X) \mid x] = \sum_i E[f(X) \mid P_i, x]p(P_i \mid x)$

Example: option pricing**

Problem: How to choose the prior distributions over models?

* See Cont R. (2006) Model uncertainty and its impact on the pricing of derivative instruments. Mathematical Finance, Wiley

** See Bannör K. and Scherer M (2014) Model Risk and Uncertainty, Springer International Publishing

Two paradigms: Model averaging vs worst-case*

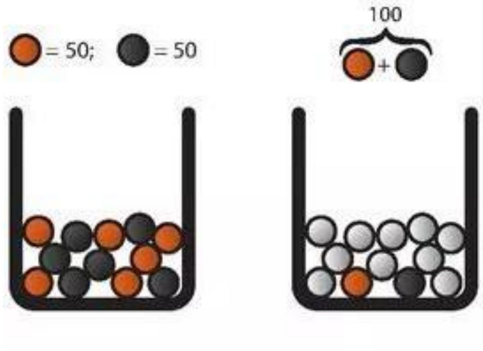
Worst case

An agent facing uncertainty maximizes his expected utility defined as the worst-case over all available models

$$\max_{X \in A} \min_{P_i \in P} E^{P_i}[U(X)]$$

Example: Ellsberg paradox (ambiguity aversion)

- Urn A: 50/50
- Urn B: assume subjective probability of distribution is 40/60



The agent will go for A

© Yields NV

* See Gilboa & Schmeidler (1989) Maxmin expected utility with non-unique prior, Journal of mathematical economics

Two paradigms: Model averaging vs worst-case*

Both methods have some key challenges in common

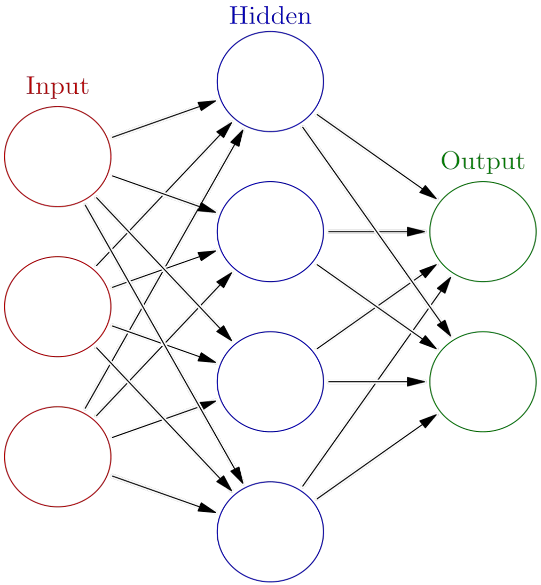
- How do we choose the candidate models?
- How to sample relevant parameters?

Machine learning can complement expert opinion to automatically generate candidate models and sample model parameters.

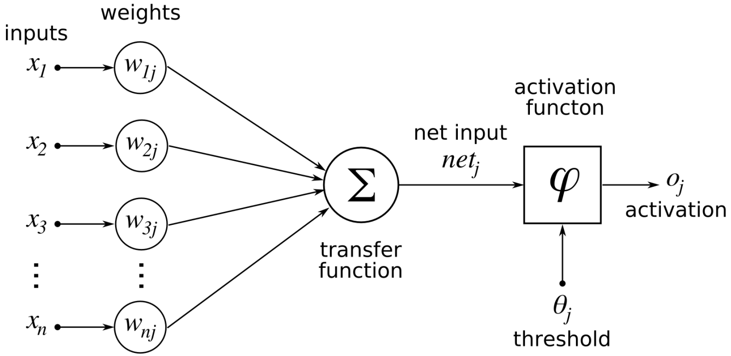
Similar to scenario generation in specification based testing of software.

Neural networks

NN Topology

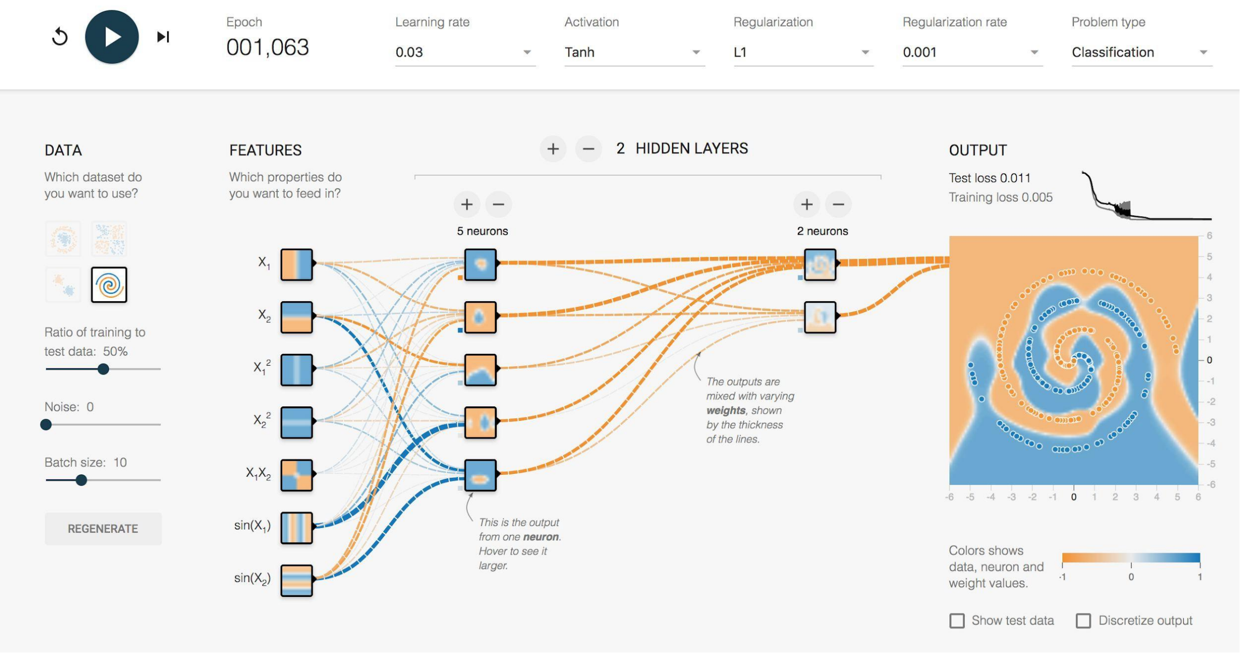


Neuron



$$\sum_i w_{ij} x_i + w_{0j} \qquad \tanh(v_j - \theta_j)$$

NN in action



Epoch 001,063

Learning rate 0.03

Activation Tanh

Regularization L1

Regularization rate 0.001

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

2 HIDDEN LAYERS

5 neurons

2 neurons

This is the output from one **neuron**. Hover to see it larger.

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

OUTPUT

Test loss 0.011

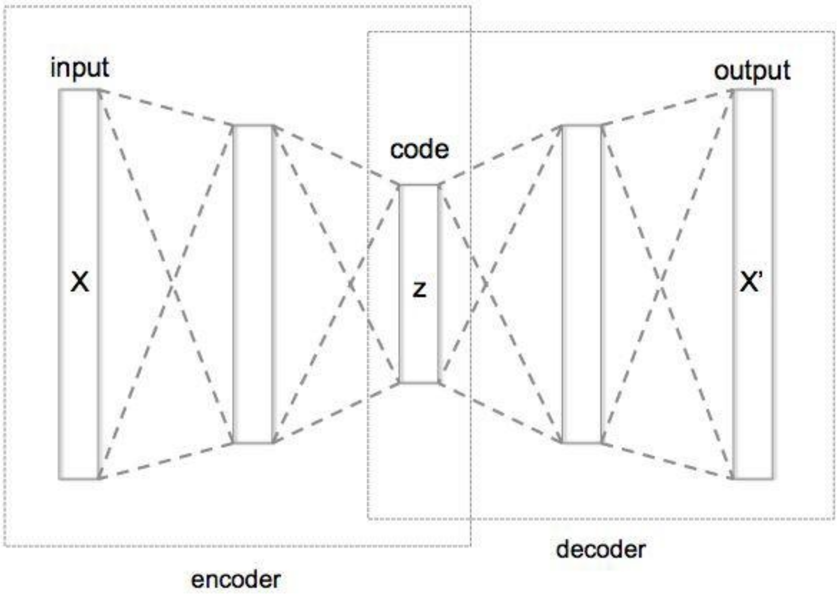
Training loss 0.005

Colors shows data, neuron and weight values.

□ Show test data

□ Discretize output

Autoencoder



- Minimize reconstruction error |x - x`|
- Linear autoencoder = PCA*
- Fast
- Non-linear activation

Used for dimensional reduction and outlier detection

* See e.g. https://www.cs.toronto.edu/~urtasun/courses/CSC411/14_pca.pdf

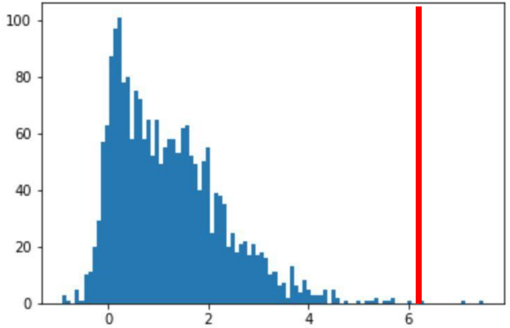
Anomaly detection with autoencoder

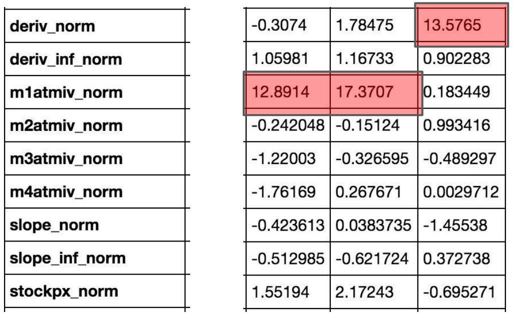
1. Train an autoencoder on the data and compute the Mahalanobis distance histogram

$$M = \sqrt{(x-\mu)^T S^{-1} (x-\mu)}$$

2. Fit a heavy-tailed distribution to the histogram to determine a cut-off parameter. Samples with Mahalanobis distance above this value are considered anomalies

3. The resulting vol surfaces are indeed outliers as can be seen via the z-scores of their parameters

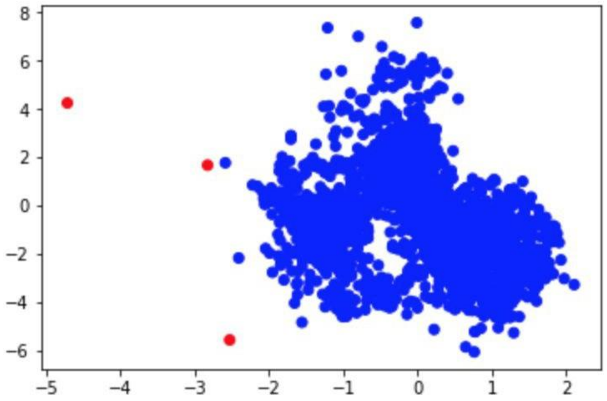




deriv_norm	-0.3074	1.78475	13.5765
deriv_inf_norm	1.05981	1.16733	0.902283
m1atmiv_norm	12.8914	17.3707	0.183449
m2atmiv_norm	-0.242048	-0.15124	0.993416
m3atmiv_norm	-1.22003	-0.326595	-0.489297
m4atmiv_norm	-1.76169	0.267671	0.0029712
slope_norm	-0.423613	0.0383735	-1.45538
slope_inf_norm	-0.512985	-0.621724	0.372738
stockpx_norm	1.55194	2.17243	-0.695271

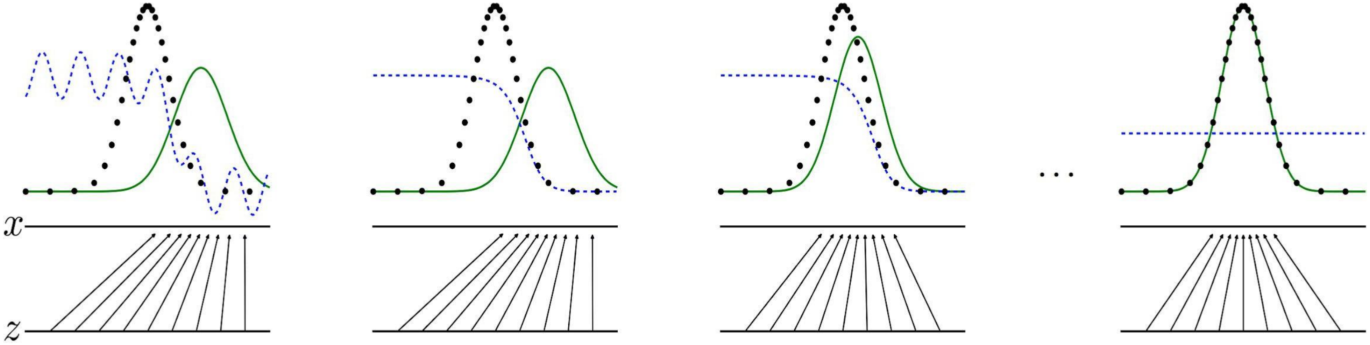
Anomaly detection with unlabelled data

The bottleneck of the autoencoder can be used to reduce the dimensionality of the problem (to e.g. plot the dataset in 2D). The red dots are the detected outliers.



Generative adversarial networks*

- Contains a generator and a discriminator
- Generative model can serve as a source for test data



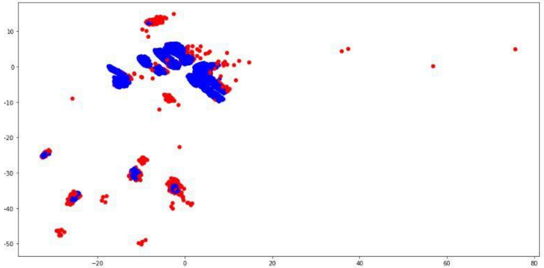
* See Goodfellow I. et al, Generative adversarial networks, https://arxiv.org/abs/1406.2661

Sampling parameters

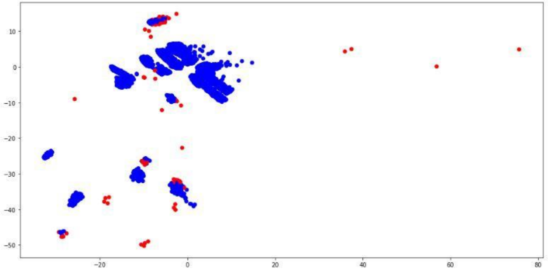
Using autoencoder to partition data

PD model (logistic regression) calibrated on Lending Club Loan Data*
900k rows, 75 columns, discrete and continuous

Random train / test split: AUC = .896



Training: 99% inner points AUC
on test set = .873



Training: 99.9% inner points AUC
on test set = .845

* See https://www.kaggle.com/wendykan/lending-club-loan-data

Sampling parameters – ctu’ed

Use GAN’s to generate realistic datasets

- As an alternative to e.g. bootstrapping
- To deal with sparse data

Example: Model performance analysis

1. Generate labeled datasets (1000)
2. Calibrate model on training set
3. Measure out of sample performance

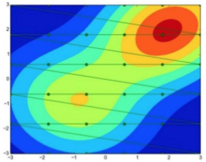
	AUC	-2 log (L)
Bootstrap	.87 - .92	.143 - .176
GAN	.84 - .93	.120 - .179

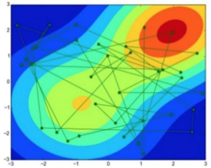
Sampling models

Hyperparameter “de-tuning”

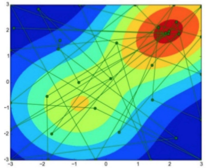
ML in general and (deep) neural network algorithms have many degrees of freedom

- Number of layers, number of nodes and connections
- Activation functions
- Learning rates
- Etc.





Grid Random



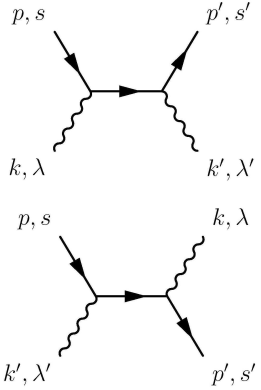
SMAC*

Genetic programming**

* See https://www.cs.ubc.ca/~hutter/papers/10-TR-SMAC.pdf
** See https://github.com/EpistasisLab/tpot

Documenting neural networks

"NN are black-box or at least hard to understand"



Similar problem in quantum field theory was solved by developing a beautiful graphical language

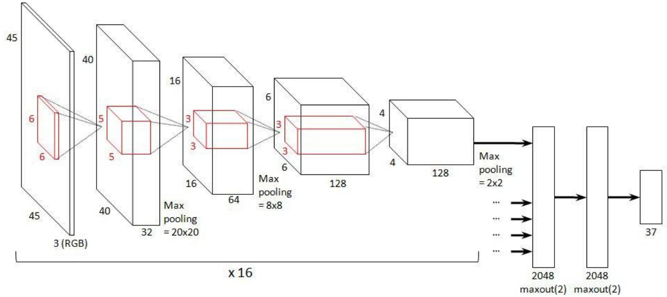
Feynman diagrams

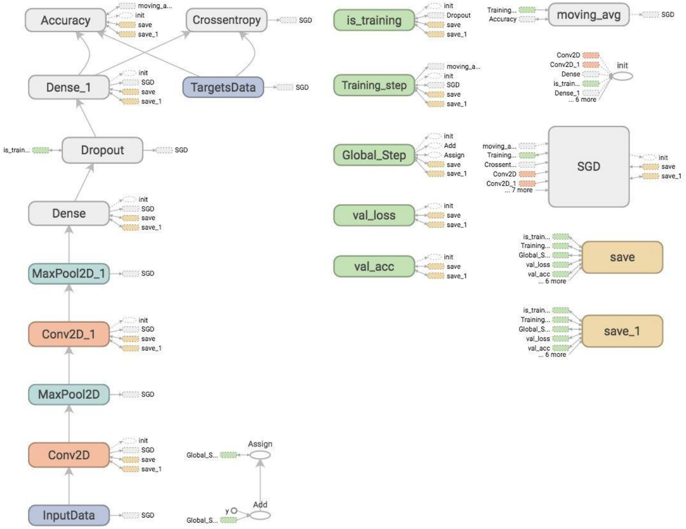
$$M_{fi} = (ie)^2 \overline{u}(\vec{p}\,', s')\not\epsilon\,'(\vec{k}\,', \lambda')^* \frac{\not p + \not k + m_e}{(p+k)^2 - m_e^2} \not\epsilon(\vec{k}, \lambda) u(\vec{p}, s) + (ie)^2 \overline{u}(\vec{p}\,', s')\not\epsilon(\vec{k}, \lambda) \frac{\not p - \not k' + m_e}{(p-k')^2 - m_e^2} \not\epsilon\,'(\vec{k}\,', \lambda')^* u(\vec{p}, s).$$

Documenting neural networks ctu’ed

Krizhevsky-style

Tensorboard - style





© Yields NV

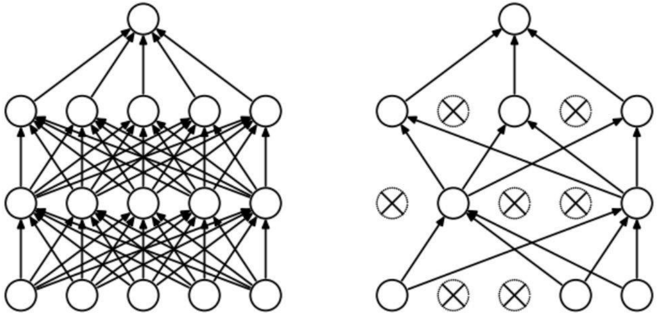
Model uncertainty in deep learning*

- **Why does my model work?**

  E.g. dropouts: avoids over-fitting and improves performance, but why?

- **What does my model know?**
  I.e. understanding the degree of certainty in the model

  E.g. train model to recognize dog breeds and present a cat
  E.g.2 train model on Dutch mortgages and present a Belgian client



(a) Standard Neural Net

(b) After applying dropout.

training

testing

* See http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf

Why does dropout work?

- Place prior distribution on the weights p(w)
- Given dataset (x: input, y: label), the posterior is p(w|x,y)
- Define simple distribtion $q_M(w)$
- Approximate posterior by q_M via minimization of the KL divergence

This is **approximate variational inference**

One can prove that

$$KL(q_M(w)|p(w|x,y)) \approx -\int q_M(w)\underbrace{\log p(y|x,w)dw}_{\boxed{\text{loss}}} + \underbrace{KL(q_M(w)|p(w))}_{\boxed{\text{L2 regularization}}}$$

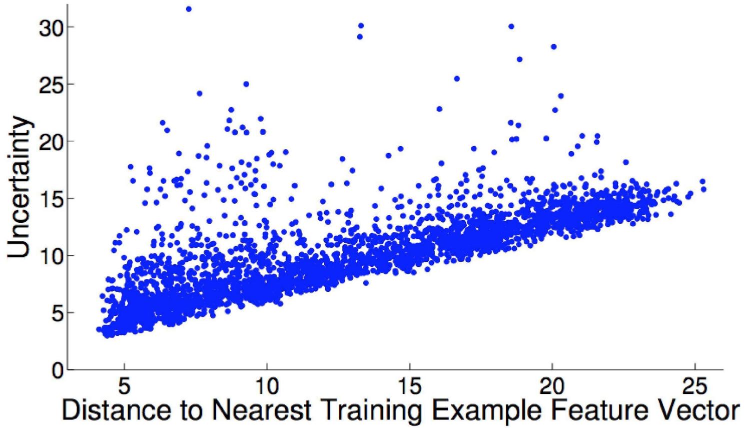
Now take $q_M(w)$ = M * diag Bernouilli

Sampling from $q_M(w)$ is randomly putting columns in M to zero = randomly setting nodes to zero = dropout

Hence, **dropout is approximately integrating over model parameters**

What does my model know?

With a Bayesian NN we can compute the uncertainty on the output by looking at the second moment



Recap

- We have introduced classical concepts of model uncertainty and model risk
- We discussed a Bayesian approach (risk; model averaging) and maxmin approaches (uncertainty; robust expected utility)
- We can use ML to generate candidate models and sample candidate model parameters automatically
- We can use Bayesian neural networks to integrate over model parameters (dropouts) and measure uncertainty

Thank you!

Yields NV

Brugmannlaan 63,
1190
Forest,
Belgium

+32 479 527
261

info@yields.io

Yields.io Ltd

8 Northumberland Ave,
Westminster, London
WC2N 5BY, UK

+44 7584 068899

info@yields.io