Deep Learning: A Bayesian Perspective

Vadim Sokolov

George Mason University
Joint work with Nick Polson

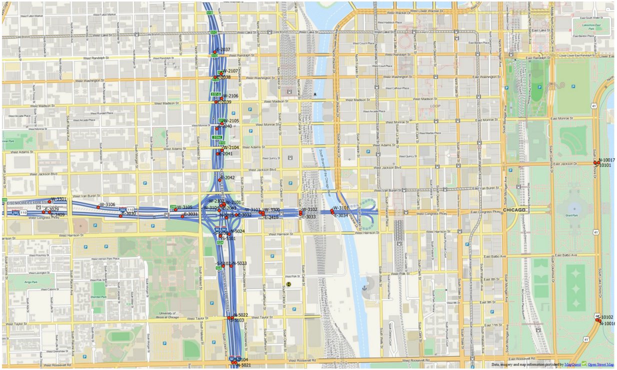
Synthesis of Statistics, Data Mining and Environmental Science
in Pursuit of Knowledge Discovery
October 31, 2017

Chicago Data

Speed, occupancy and flow, averaged over 5 minutes

1500 highway loop-detectors around Chicago area

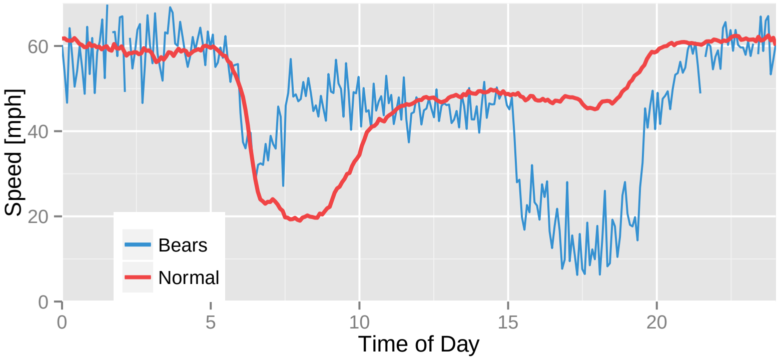
Approx 50Mb per sensor (75Gb total)



Non-recurrent traffic patterns

Chicago Bears game

Impact on I-55 north bound travel



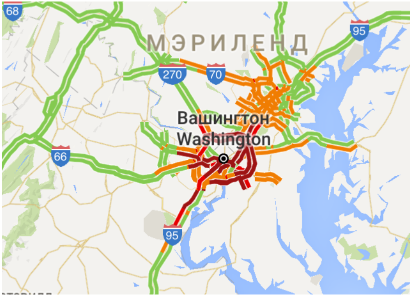
New York Giants at Bears on
Thursday October 10, 2013

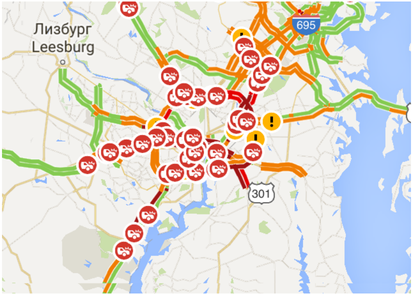
Non-recurrent traffic conditions

Weather and Accidents

Impact of light snow and accidents travel times

Snow in DC area on January 21, 2016





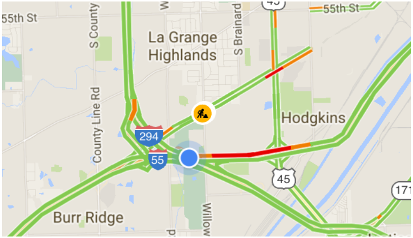
Snapshot at 12:41am (traffic flow is very light at this time of the day)

Non-recurrent traffic conditions

Protesters

Impact of people protesting on a bridge over a highway

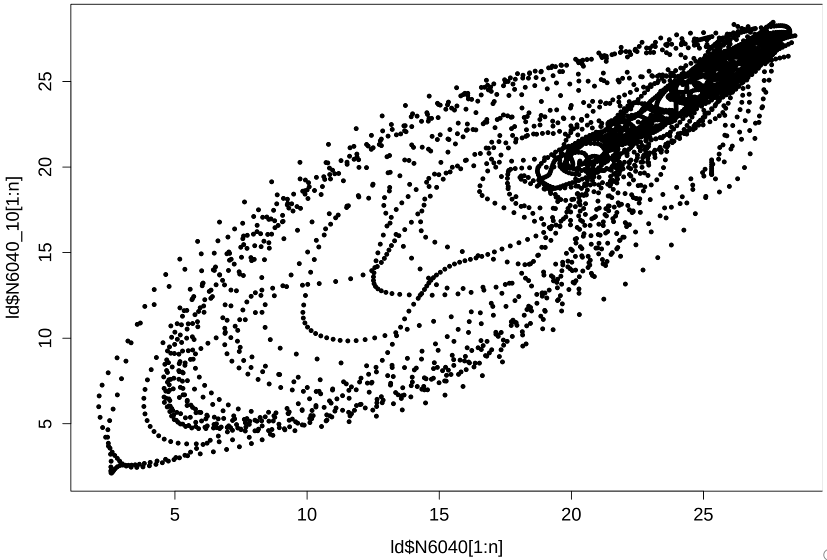
Interstate I-55, 20 miles away from Chicago on February 27, 2016



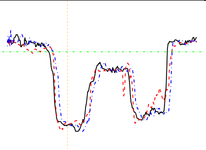
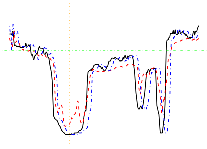
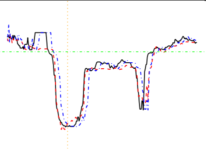
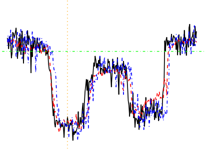
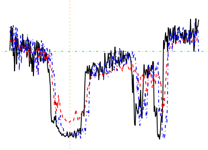
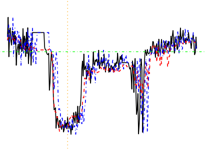
Snapshot at 2:11 PM

Relations are Highly Nonlinear

Shockwawe effect in traffic flows



Forecast Fitting

model	bears game	weather day	normal day
DL+Filter			
DL			

black = data, red = model forecast, blue = naive forecast

Why do we care about DL?

Input space (X) includes numerical, text (word2vec), images, videos Vectors,

matrices and tensors, ...

- Google's translation algorithm
- $\sim$ 1-2 billion parameters
- Alexa's speech recognition: 100 million parameters
- Networks will get larger and more efficient
- Google Waymo

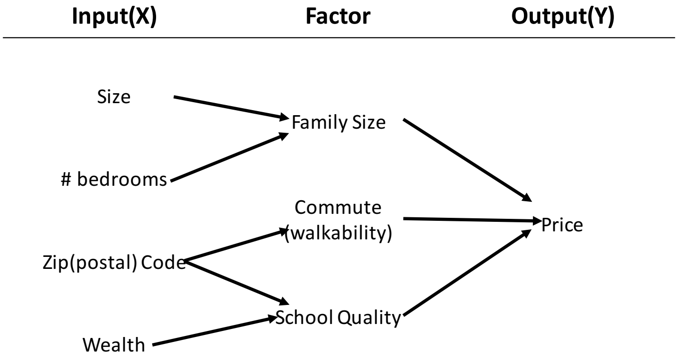
Advances in computing speed (Nvidia) lets us train and implement Deep Learning in real-time.
Google Waymo's Lidar processes 6MB Data per second ...

Multi-Layer Deep Models

NN models one layer!! Key is to use multi "deep" layers

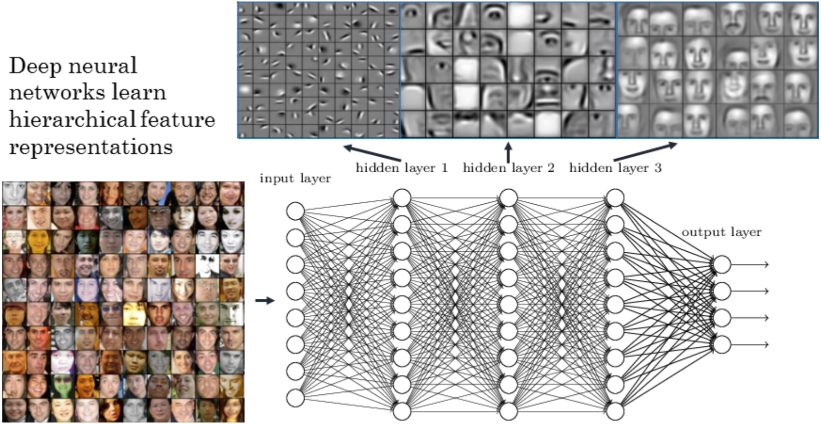
Learn weight and connections in hidden layers

Predicting House Prices ...



Multi-Layer Faces

Deep neural networks learn hierarchical feature representations



Kolmogorov-Arnold

There are no multivariate functions just superpositions of univariate ones

Let $f_1, \ldots, f_L$ be given univariate activation functions. We set

$$F(X) = (f_1 \circ \ldots \circ f_L)(X)$$

$$f_l = \sigma_l \left(\sum_{j=1}^{N_l} W_{lj} X_j + b_l \right) = \sigma_l (W_l X_l + b_l), \quad 1 \leq l \leq L,$$

Our deep predictor has hidden units N_l and depth L.
Put simply, we model a high dimensional mapping F via the superposition of univariate semi-affine functions.

Kolmogorov-Arnold Example

Interaction terms, $x_1 x_2$ and $(x_1 x_2)^2$, and max functions, $\max(x_1, x_2)$ can be expressed as nonlinear functions of semi-affine combinations. Specifically,

$$x_1 x_2 = \frac{1}{4}(x_1 + x_2)^2 - \frac{1}{4}(x_1 - x_2)^2$$

$$\max(x_1, x_2) = \frac{1}{2}|x_1 + x_2| + \frac{1}{2}|x_1 - x_2|$$

$$(x_1 x_2)^2 = \frac{1}{4}(x_1+x_2)^4 + \frac{7}{4 \cdot 3^3}(x_1-x_2)^4 - \frac{1}{2 \cdot 3^3}(x_1+2x_2)^4 - \frac{2^3}{3^3}(x_1+\frac{1}{2}x_2)^4$$

Shallow Learner

Our traditional model

$$\hat{Y} = f_1^{W_1,b_1}(f_2(W_2X + b_2)) = f_1^{W_1,b_1}(Z)$$

PCA: $Z = f_2(X) = W^\top X + b$

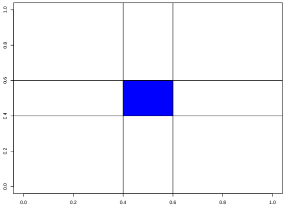
PPR: $Z = f_2(X) = \sum_{i=1}^{N_1} f_i(W_{i1}X_1 + \ldots + W_{ip}X_p)$

Examples: Principal component analysis (PCA), partial least squares (PLS), reduced rank regression (RRR), linear discriminant analysis (LDA), project pursuit regression (PPR), and logistic regression

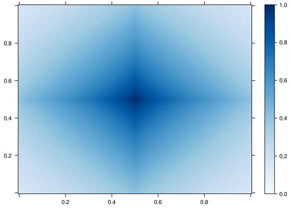
Deep Learning Predictors

Smart conditional averaging

The competitors: Trees, RF, GP.



(a) Tree Kernel

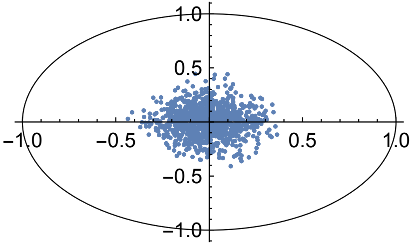


(b) Random Forest Kernel

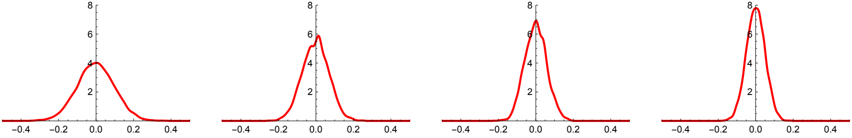
Few points will be neighbors in a high dimensional input space.

Whats wrong with Kernels?

2D image of 1000 uniform samples from a 50-dimensional ball B_{50}.



Marginal distribution shrinks as dimensionality of the space grows



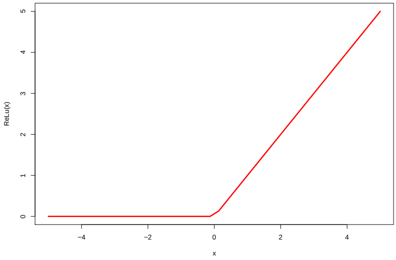
(a) $p = 100$ (b) $p = 200$ (c) $p = 300$ (d) $p = 400$

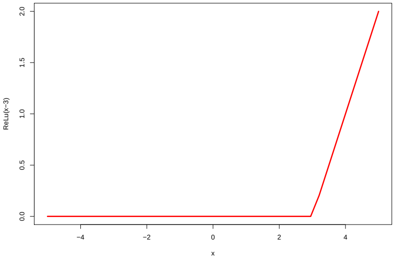
ReLU

Affine transformation defines a plane

ReLU: $f(x) = \max(0, x)$ "fires up" if point X in on the "right" side of this plane

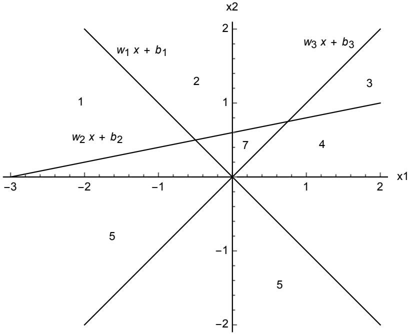
Bias terms allow for hyperplanes not to go through 0.





Example: Three-Layer Network

It takes 3 neurons to define 8 regions in 2D



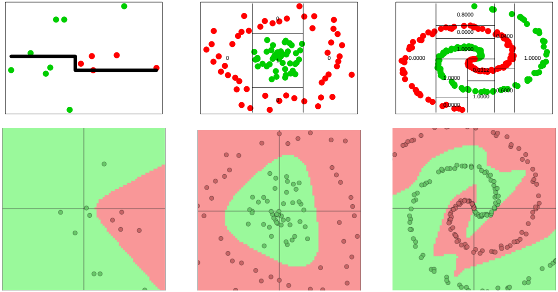
Hyperplanes defined by three neurons with ReLU activation functions

$$\hat{Y}(X) = \sum_{k \in K} w_k(X)\hat{Y}_k(X),$$

Tree vs DL example

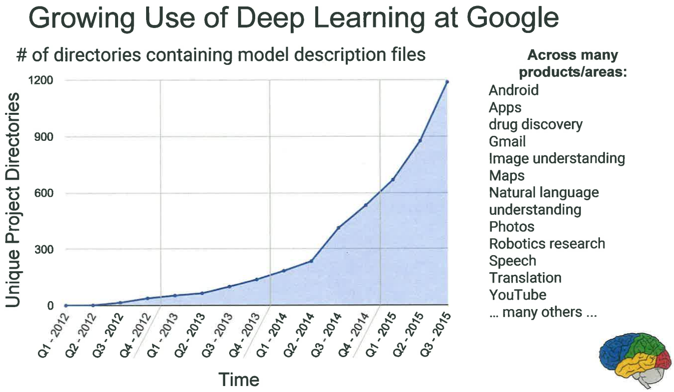
$$Y = \mathrm{softmax}(w^0 Z^2 + b^0)$$

$$Z^2 = \tanh(w^2 Z^1 + b^2) \quad Z^1 = \tanh(w^1 X + b^1).$$



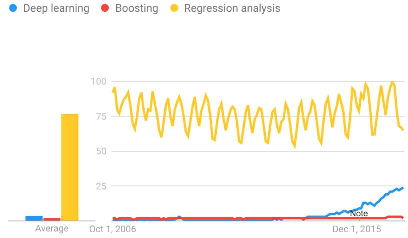
An advantage of deep architectures is that the number of hyper-planes grow exponentially with the number of layers.

Academic Curiosity? ... but it works so well!!



Still a niche? ... becoming mainstream

Worldwide. 9/3/06 - 9/3/17.



Worldwide. 9/3/06 - 9/3/17.

Training, Validation, and Testing

Given the training dataset $D = \{Y^{(i)}, X^{(i)}\}_{i=1}^{T}$ of input-output pairs and a loss function $\mathcal{L}(Y, \hat{Y})$, we compute

$$\hat{W} = (\hat{W}_0, \ldots, \hat{W}_L) \text{ and } \hat{b} = (\hat{b}_0, \ldots, \hat{b}_L)$$

by solving

$$\arg\min_{W,b} \frac{1}{T} \sum_{i=1}^{T} \mathcal{L}(Y_i, \hat{Y}^{W,b}(X_i)) .$$

For the L_2-norm for a traditional least squares

$$\mathcal{L}(Y_i, \hat{Y}(X_i)) = \| Y_i - \hat{Y}(X_i) \|_2^2 ,$$

our target function becomes the mean-squared error (MSE).

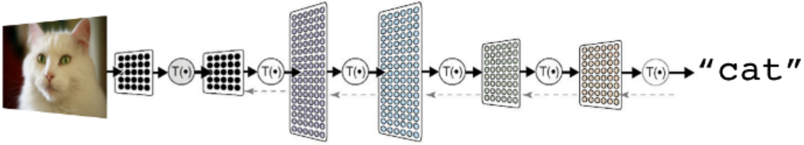
Back-Propagation

Stochastic gradient descent adapted to a deep learning setting.

Proximal Newton Algorithm: $\nabla\mathcal{L}$ available for deep learners.

One caveat of back-propagation is the multi-modality of the system to be solved (and the resulting slow convergence properties).

Deep learning methods heavily rely on the availability of large computational power: `NVIDIA GPU` and `Google's TPU`.



Tensor Processing Unit

The problem: Deep Learning is typically applied to large datasets.

A driverless car processes 6GB data per second.

Applications need computational speed

The solution: A specialized processor called Tensor Processing Unit (TPU, GPU, CPU)

Processing advances tied to TPU not CPU

Google TPU 2.0 and Nvidia TeslaV100

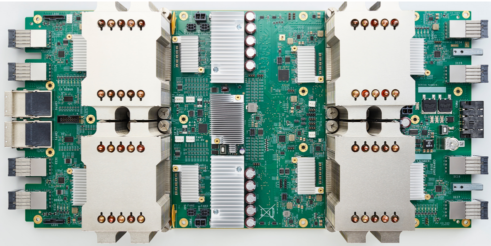
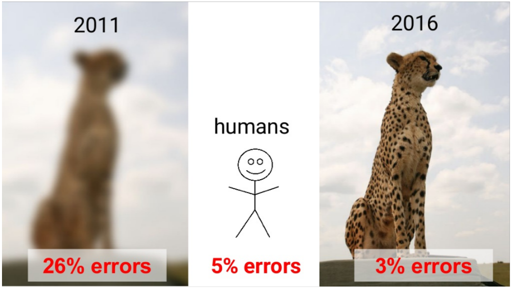
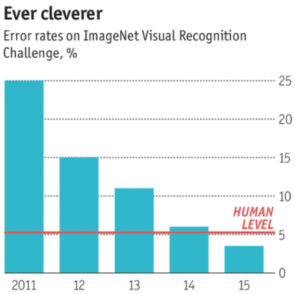


Image recognition has improved





Machines are becoming better than humans

Application: Identifying Skin Cancer

Dataset: 130,000 images of skin lesions/2,000 different diseases

Test data: 370 high-quality, biopsy-confirmed images

Better performance than 23 Stanford dermatologists

10,000 hours no match for deep learning and large datasets

Application: Training A New Rembrandt

Analyze all 346 of Rembrandt's paintings

Identify all geometric patterns used by Rembrandt.

Reassemble into a fully formed face and bust

Google: αGo

Supervised and Reinforcement Learning

Value Function and Tree Search

Convenient

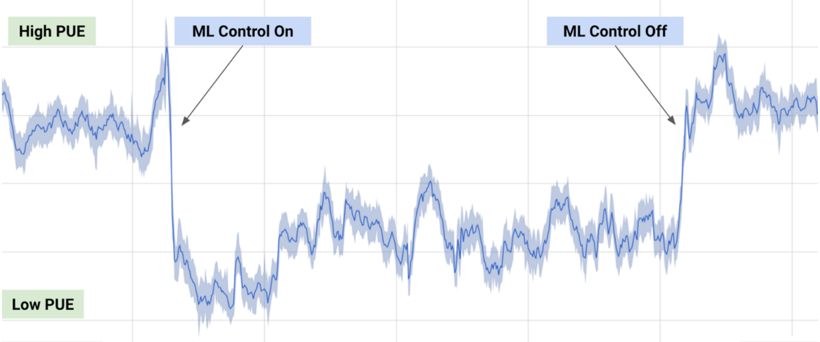
- Fullyobserved
- Discrete action space
- Perfectsimulator
- Relativelyshort game
- Trial-and errorexperience
- Largehuman datasets

Inconvenient

- Actions executed awkwardly
- Incomplete information
- Imperfectsimulator
- Longer tasks, hard to assess value
- Hard to practice millions of times
- Small human data sources

Google Data Center Cooling Costs Reduced by 40%

Monitoring real-time conditions and adjusting data center climate control based on past experience



What is Wrong with DL

Point estimates

No model selection mechanism

No regularization mechanism

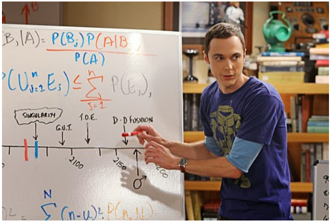
What is Bayes?

Incorporate prior knowledge about unknown θ before data X is observed

Understand uncertainty about θ after data is observed

$$p(\theta|X) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}$$

Posterior $p(\theta|X)$ has all the information about θ we can extract from X, given the prior



Bayesian Learning

Given training data $D = (X, Y)$, the goal is to build a model $p(y|x, \theta, X, Y)$

Define prior $p(\theta)$

Find posterior (training)

$$p(\theta|X, Y) = \frac{p(Y|\theta, X)p(\theta)}{\boxed{\int p(x|\theta, Y)p(\theta)d\theta}}$$

Predict using total probability

$$\boxed{p(y_{\text{new}}|x_{\text{new}}, X, Y) = \int p(y_{\text{new}}|x_{\text{new}}, \theta)p(\theta|X, Y)d\theta}$$

Bayes predictor averages over all of the models parametrized by θ

$\boxed{\textit{equation}}$ = intractable

Probabilistic Interpretation

In a traditional probabilistic setting, view the output Y as a random variable generated by a probability model $p(Y|Y^{W,b}(X))$ with conditioning is on the predictor $\hat{Y}(X)$.

The loss function is then

$$\mathcal{L}(Y, \hat{Y}) = -\log p(Y|Y^{\hat{W},\hat{b}}(X)),$$

the negative log-likelihood.

When predicting the probability of congestion, we have a multinomial logistic regression model with cross-entropy loss function.

Bayes + DL

Bayesian inference for DL: reparameterization

Calculate Monte Carlo gradients using variational inference.

The variation inference approximates the posterior $p(\theta \mid X, Y)$ with a variation distribution $q(\theta \mid \phi)$, $\theta = (W, b)$.

$$\mathrm{KL}(q \,||\, p) = \int q(\theta \mid D, \phi) \log \frac{q(\theta \mid D, \phi)}{p(\theta \mid D)} d\theta.$$

Variational Inference

KL requires intractable $\log p(\theta \mid D)$

Useful identity

$$\log p(D) = \text{ELBO}(\phi) + \text{KL}(q \mid\mid p)$$

The sum does not depend on ϕ, thus minimizing $\text{KL}(q \mid\mid p)$ is the same that maximizing

$$\text{ELBO}(\phi) = \int q(\theta \mid D, \phi) \log \frac{p(Y \mid X, \theta) p(\theta)}{q(\theta \mid D, \phi)} d\theta$$

$\text{ELBO}(\phi) \to \max_{\phi}$ is solved using stochastic gradient descent.

Gradient of ELBO

To calculate the gradient, it is convenient to write the ELBO as

$$\mathrm{ELBO}(\phi) = \int q(\theta \mid D, \phi) \log p(Y \mid X, \theta) d\theta - \int q(\theta \mid D, \phi) \log \frac{q(\theta \mid D, \phi)}{p(\theta)} d\theta$$

$$\nabla_{\phi} \int q(\theta \mid D, Y, \phi) \log p(Y \mid X, \theta) d\theta = \nabla_{\phi} E_{\theta \sim q} \log p(Y \mid X, \theta)$$

Is not a expectation!

Reparametrization

Reparametrization trick represents θ as a value of a deterministic function, $\theta = g(\epsilon, X, \phi)$, where $\epsilon \sim r(\epsilon)$ does not depend on ϕ. Now, the derivative is given by

$$\nabla_\phi E_q \log p(Y \mid X, \theta) = \int r(\epsilon) \nabla_\phi \log p(Y \mid g(\epsilon, x, \phi)) d\epsilon = E_\epsilon[\nabla_g \log p(Y \mid g(\epsilon, X, \phi)) \nabla_\phi g(\epsilon, X, \phi)].$$

The reparametrization is trivial in the case when $q(\theta \mid D, \phi) = N(\theta \mid \mu(D, \phi), \Sigma(D, \phi))$, than $\theta = \mu(D, \phi) + \epsilon \Sigma(D, \phi),\ \epsilon \sim N(0, I)$.

Bayesian Regularisation

Typically we find MAP (poor man's version of Bayes) estimator via

$$\log p(Y|X,\theta) + \log p(\theta) \to \max_{\theta}$$

Via VI we search for distribution over θ

$$\int q(\theta|D,\phi) \log p(Y|X,\theta) d\theta - \mathrm{KL}(q(\theta|\phi)||p(\theta)) \to \max_{\phi}$$

Equivalent to adding noise to the DL parameters θ at each iteration

Normal Dropout

Dropout is a model selection technique designed to avoid over-fitting in the training process.

Normal dropout add normal noise to θ at each iteration.

The dropout architecture becomes

$$
\begin{aligned}
D_i^{(l)} &\sim \mathrm{N}(1, \sigma^2), \\
W^{(l)} &= W^{(l)} \star D^{(l)}, \\
Z_i^{(l)} &= W_i^{(l)} X^{(l)} + b_i^{(l)}. \\
&\dots
\end{aligned}
$$

Bayesian Regularization for DL

Take $q(\theta|\alpha,\gamma) = N(\theta|\alpha,\gamma\alpha^2)$

Then Bayesian regularization (ELBO)

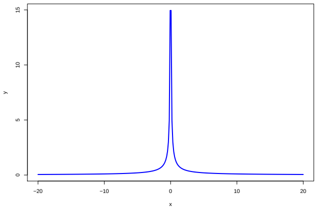
$$\int N(\theta|\alpha,\gamma\alpha^2)\log p(Y \mid X,\theta)d\theta - \mathrm{KL}(\mathrm{q} \mid\mid \mathrm{p}(\theta)) \rightarrow \max_{\alpha}$$

First term is the objective function of the DL + Normal

Dropout training procedure

We have additional KL term!

Need to find $p(\theta)$ so that KL does not depend on α



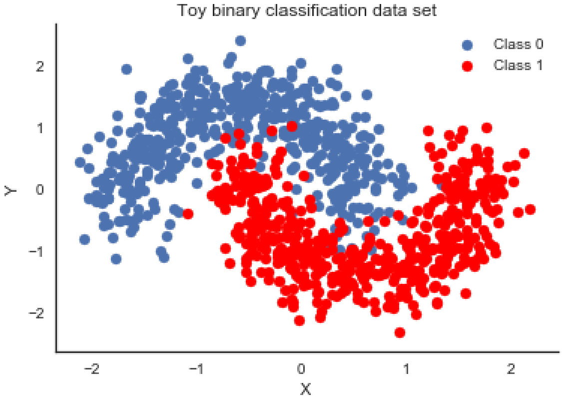
$$p(\theta) \propto \frac{1}{|\theta|}$$

Bayes DL Classification

2-layer network (MLP) with tanh activation

5-neurons

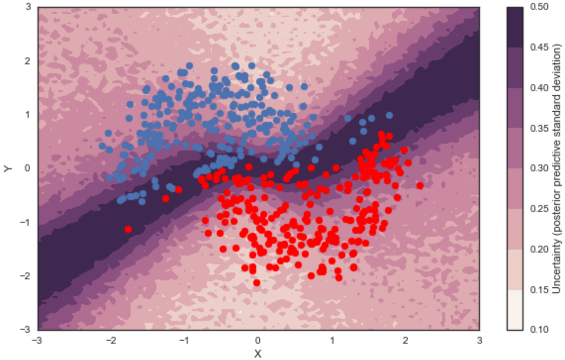
1000 observations (.5 for training)



Prediction

Used automated variational inference (AVI)

Can calculate uncertainty in predicted value!

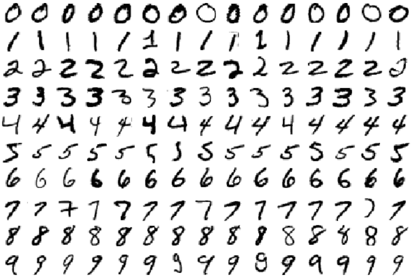


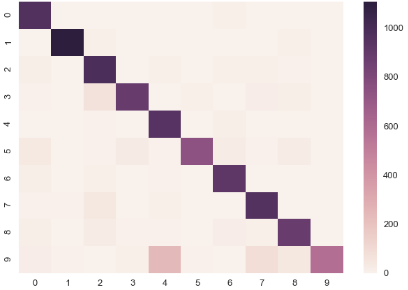
Bayes DL Classification II

2-layer network (MLP) with tanh activation

5-neurons

60k observations (60k for training)





Discussion

Many successful applications. Extremely high dimensionality

SGD is very powerful tool to obtain point estimates

Recently: first steps towards Bayes + DL: Dropout + VI

Still baby steps, methods are not scalable (4 hours to train DL for MNIST vs 2 minutes for Chicago Traffic)

Uncertainty assertion for deep predictors?

Decision making and policy under uncertainty?