BML. 6.1 Bayesian NNs/DL

DataLab CSIC

Background

Sources

MLE-DL Intro ML Chapter 7 https://datalab-icmat.github.io/courses_stats.html

Muller, DRI (1998)
Gallego, DRI (2022)

Motivation

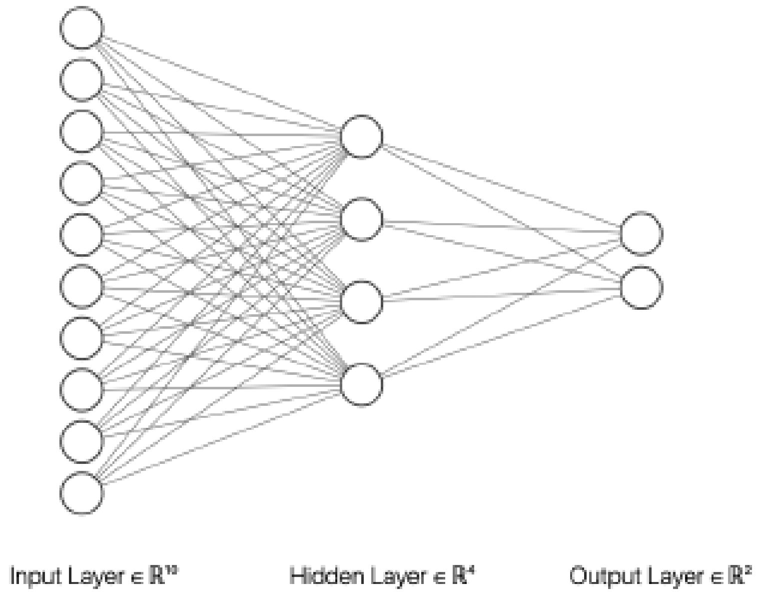
- AI is ultra cool because of deep learning/nns
- ML is very cool because of deep learning/nns
- Stats is pretty cool because of deep learning/nns
- Annex 1 of EU AI Act

- Many exciting research questions
- Many exciting computational problems
- Many exciting applications
- Probabilistic ML

Brief history of NNs

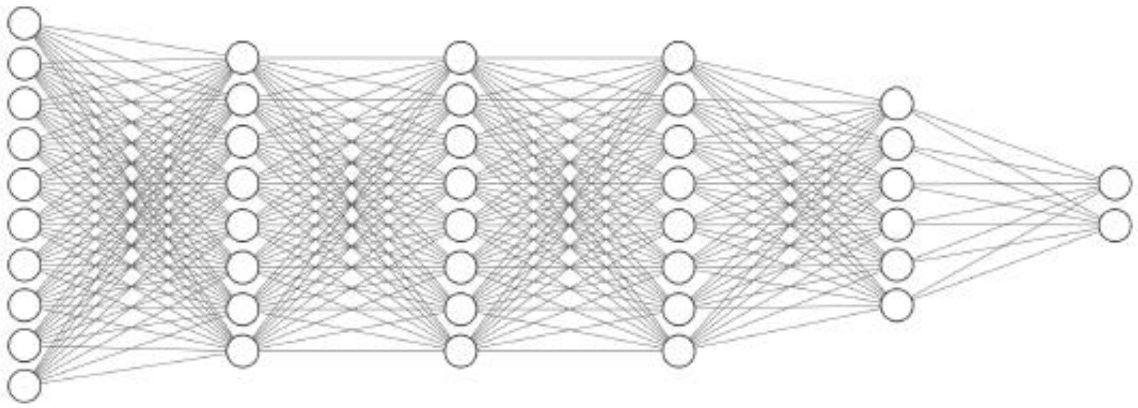
When	What	Why	Why not
End of 50’s, Beg of 60’s	Rosenblatt’s perceptron	Efficiente scheme Good branding	Minsky& Papert (1968)
End of 80’s, Beg of 90’s	Cybenko’s representation Shallow NNs	Good branding Impulse from CS comm	Tech problems (vanishing gradient) Emergence of SVM and others
2010’s on	Deep learning, variants Outstanding aplications	Massive labeled data Rediscovery of SGD GPUs ReLUs et al Domain specific architectures Winning Imagenet comp	

Concept



$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$
$$\epsilon \sim N(0, \sigma^2),$$
$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

(Shallow) Neural nets



$$\{f_0, f_1, ..., f_{L-1}\}$$

$$z_{l+1} = f_l(z_l, \gamma_l).$$

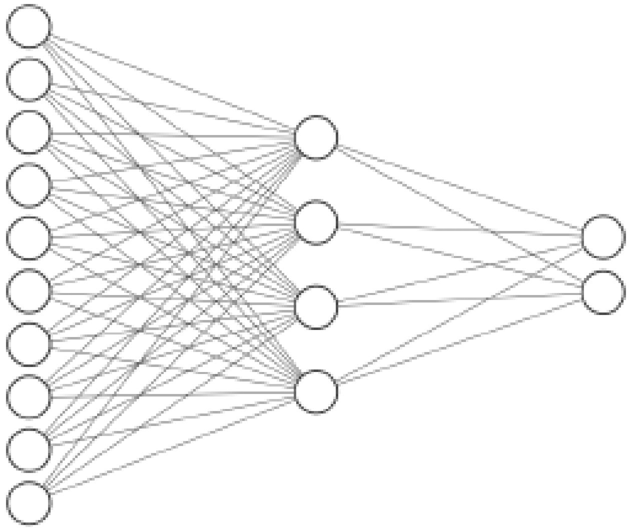
$$y = \sum_{j=1}^{m_L} \beta_j z_{L,j} + \epsilon$$
$$\epsilon \sim N(0, \sigma^2),$$

Deep neural nets

What is to be gained?

- Uncertainties in predictions
- Improved decision making based on above (risk aversion etc…)
- Some explainability via hypothesis testing
- Architecture choice

Shallow nets



$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

Linear in beta's, nonlinear in gamma's

Motivation. Cybenko's theorem

Any continuous function in the r-dimensional cube may be approximated by models of type $\sum_{j=1}^{m} \beta_j \psi(x'\gamma_j)$ when the activation function is sigmoidal

(as m goes to infty)

Training with regularisation

$$\min_{\beta,\gamma} f(\beta,\gamma) = \sum_{i=1}^{n} f_i(\beta,\gamma) = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} \beta_j \psi(x_i' \gamma_j) \right)^2$$

$$\min g(\beta,\gamma) = f(\beta,\gamma) + h(\beta,\gamma),$$

Weight decay

$$h(\beta,\gamma) = \lambda_1 \sum \beta_i^2 + \lambda_2 \sum \sum \gamma_{ji}^2$$

Ridge

Backpropagation (CASI 18, care with notation)

Algorithm 18.1 BACKPROPAGATION

1 Given a pair x, y, perform a "feedforward pass," computing the activations $a_\ell^{(k)}$ at each of the layers $L_2, L_3, \ldots, L_K$; i.e. compute $f(x; \mathcal{W})$ at x using the current $\mathcal{W}$, saving each of the intermediary quantities along the way.

2 For each output unit ℓ in layer L_K, compute

$$\begin{aligned}\delta_\ell^{(K)} &= \frac{\partial L[y, f(x, \mathcal{W})]}{\partial z_\ell^{(K)}} \\ &= \frac{\partial L[y, f(x; \mathcal{W})]}{\partial a_\ell^{(K)}} \dot{g}^{(K)}(z_\ell^{(K)}),\end{aligned} \tag{18.10}$$

where $\dot{g}$ denotes the derivative of $g(z)$ wrt z. For example for $L(y, f) = \frac{1}{2}\|y - f\|_2^2$, (18.10) becomes $-(y_\ell - f_\ell) \cdot \dot{g}^{(K)}(z_\ell^{(K)})$.

3 For layers $k = K-1,\ K-2, \ldots, 2$, and for each node ℓ in layer k, set

$$\delta_\ell^{(k)} = \left(\sum_{j=1}^{p_{k+1}} w_{j\ell}^{(k)} \delta_j^{(k+1)} \right) \dot{g}^{(k)}(z_\ell^{(k)}). \tag{18.11}$$

4 The partial derivatives are given by

$$\frac{\partial L[y, f(x; \mathcal{W})]}{\partial w_{\ell j}^{(k)}} = a_j^{(k)} \delta_\ell^{(k+1)}. \tag{18.12}$$

Bayesian NNs

Bayesian analysis of shallow neural nets (fixed arch)

$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

Bayesian analysis of shallow neural nets (fixed arch)

$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

$$\beta_i \sim N(\mu_\beta, \sigma_\beta^2) \text{ and } \gamma_i \sim N(\mu_\gamma, S_\gamma^2)$$

$$\mu_\beta \sim N(a_\beta, A_\beta),\ \mu_\gamma \sim N(a_\gamma, A_\gamma),\ \sigma_\beta^{-2} \sim \ Gamma(c_b/2, c_b C_b/2)$$

$$S_\gamma^{-1} \sim Wish(c_\gamma, (c_\gamma C_\gamma)^{-1}) \text{ and } \sigma^{-2} \sim Gamma(s/2, sS/2)$$

Objects of interest

'Indirectly', the posterior

$$p(\beta, \gamma, \nu|D) = \frac{p(\beta, \gamma, \nu)p(D|\beta, \gamma, \nu)}{\int p(\beta, \gamma, \nu)p(D|\beta, \gamma, \nu)d\beta d\gamma d\nu}$$

Directly, the predictive

$$p(y_{N+1}|D, x_{N+1}) = \int p(y_{N+1}|\beta, \gamma, \nu, x_{N+1})p(\beta, \gamma, \nu|D)d\beta d\gamma d\nu$$

Objects of interest

Lemma 2.1. *Let* $z_{ij} = z_{ij}(\gamma) = \psi(x_i'\gamma_j)$, $Z = (z_{ij})_{i=1,\ldots,N}^{j=1,\ldots,M}$, $1 = (1)_{i=1,\ldots,M}$, $A = Z'Z/\sigma^2$, $\rho = Z'y/\sigma^2$, $C = 1/\sigma_\beta^2 I$, $\delta = \mu_\beta/\sigma_\beta^2 1$. *Let* $m_b(\gamma) = (A + C)^{-1}(\rho + \delta)$ *and* $S_b(\gamma) = (A + C)^{-1}$. *Then,*

$$p(D|\gamma) = \frac{p[\beta = m_b(\gamma)]}{p[\beta = m_b(\gamma)|y, \gamma]} \prod_{i=1}^{N} p[y_i|\beta = m_b(\gamma), \gamma]$$

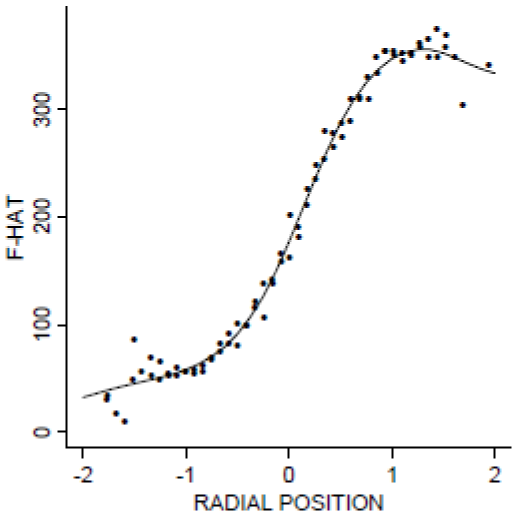
$$= p[\beta = m_b(\gamma)]|S_b(\gamma)|^{1/2} \prod_{i=1}^{N} p[y_i|\beta = m_b(\gamma), \gamma].$$

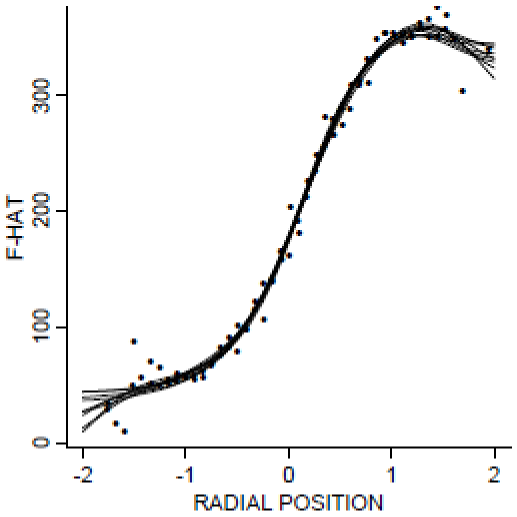
Bayesian analysis of shallow neural nets (fixed arch)

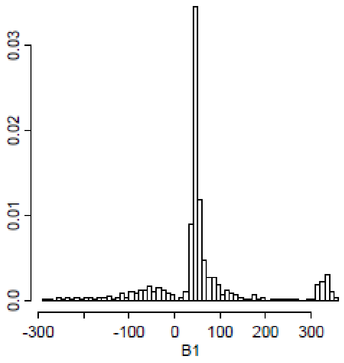
Start with arbitrary (β, γ, ν).
while *not convergence* **do**
    Given current (γ, ν), draw β from $p(\beta|\gamma, \nu, y)$ (a multivariate normal).
    for $j = 1, \ldots, m$, *marginalizing in* β *and given* ν **do**
        Generate a candidate $\tilde{\gamma}_j \sim g_j(\gamma_j)$.
        Compute $a(\gamma_j, \tilde{\gamma}_j) = \min\left(1, \frac{p(D|\tilde{\gamma}, \nu)}{p(D|\gamma, \nu)}\right)$ with $\tilde{\gamma} = (\gamma_1, \gamma_2, \ldots, \tilde{\gamma}_j, \ldots, \gamma_m)$.
        With probability $a(\gamma_j, \tilde{\gamma}_j)$ replace γ_j by $\tilde{\gamma}_j$. If not, preserve γ_j.
    end
    Given β and γ, replace ν based on their posterior conditionals:
    $p(\mu_\beta|\beta, \sigma_\beta)$ is normal; $p(\mu_\gamma|\gamma, S_\gamma)$, multivariate normal; $p(\sigma_\beta^{-2}|\beta, \mu_\beta)$, Gamma; $p(S_\gamma^{-1}|\gamma, \mu_\gamma)$, Wishart; $p(\sigma^{-2}|\beta, \gamma, y)$, Gamma.
end

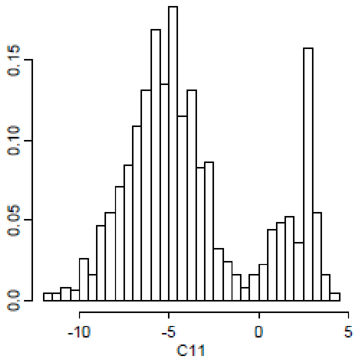
Uncertainty in predictions, explainability

$$\hat{f}(x) = \hat{E}(y_{n+1}|x_{n+1}, D) = \frac{1}{k}\sum_{t=1}^{k} E(y_{N+1}|x_{n+1}, \theta = \theta_t)$$

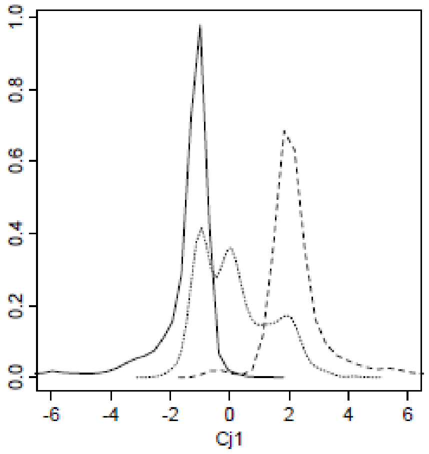


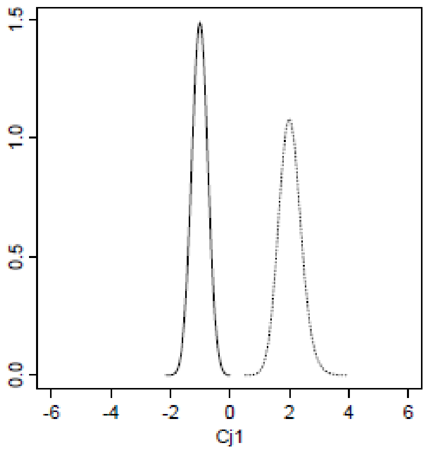






The need for architecture selection





Bayesian analysis of shallow neural nets (var arch)

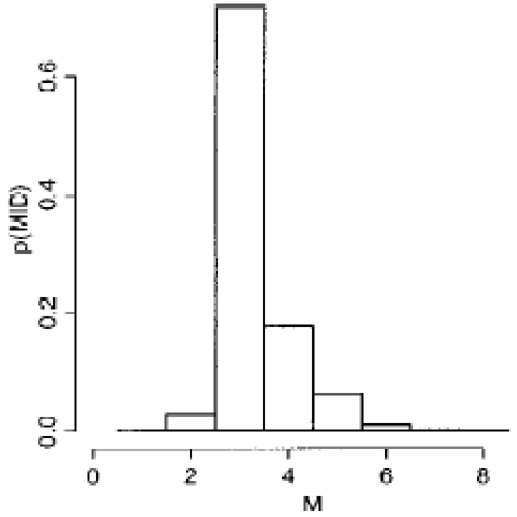
$$\begin{aligned} y &= x_i'a + \sum_{j=1}^{m^*} d_j \beta_j \psi(x'\gamma_j) + \epsilon \\ & \epsilon \sim N(0, \sigma^2), \\ & \psi(\eta) = \exp(\eta)/(1+\exp(\eta)), \\ Pr(d_j = k | d_{j-1} = 1) &= (1-\alpha)^{1-k} \times \alpha^k, k \in \{0,1\} \end{aligned}$$

$$\beta_i \sim N(\mu_b, \sigma_\beta^2),\ a \sim N(\mu_a, \sigma_a^2),\ \gamma_i \sim N(\mu_\gamma, \Sigma_\gamma).$$

Reversible jump algo

Reversible jump algo

1. $\gamma_{jk}|\gamma_{-jk}, M, \nu, D, \quad j = 1, \ldots, M+1, \quad k = 0, \ldots, p.$
2. $d_j|d_{-j}, \gamma_1, \ldots, \gamma_M, \gamma_{M+1}, \nu, D, \quad j = 1, \ldots, M+1.$
3. $\beta_1, \ldots, \beta_M, \lambda|\gamma_1, \ldots, \gamma_M, M, \nu, D.$
4. $\nu|\beta_1, \gamma_1, \ldots, \beta_M, \gamma_M, \lambda, D.$



Other

Non-linear autoregression

Semi-parametric regression

(Gaussian process)

$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$

$$\epsilon \sim N(0, \sigma^2),$$

$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

HMC

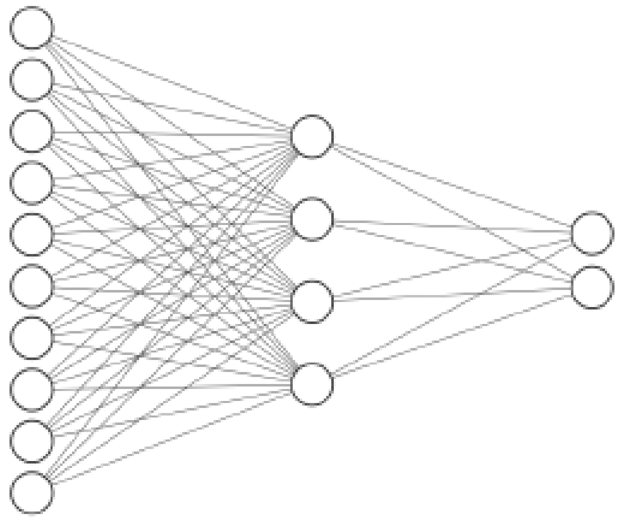
$$U(\theta) = \tau_\beta \sum_{j=1}^{m} \beta_j^2/2 + \tau_\gamma \sum_{j=1}^{d}\sum_{k=1}^{m} \gamma_{j,k}^2/2 + \tau \sum_{i=1}^{n} (y_i - f_i(\beta,\gamma))^2/2, \qquad H(\theta, r) = U(\theta) + \frac{1}{2}\sum_{j=1}^{l} q_j^2$$

Start with arbitrary $\theta_0 = (\beta_0, \gamma_0)$.
while *not convergence* **do**
  Given current θ_t and $q_t \sim \mathcal{N}(0, I)$, perform one or more leapfrog integration steps

$$q_{t+\frac{1}{2}} = q_t - \frac{\epsilon}{2}\nabla U(\theta_t)$$
$$\theta_{t+1} = \theta_t + \epsilon q_{t+\frac{1}{2}}$$
$$q_{t+1} = q_{t+\frac{1}{2}} - \frac{\epsilon}{2}\nabla U(\theta_{t+1})$$

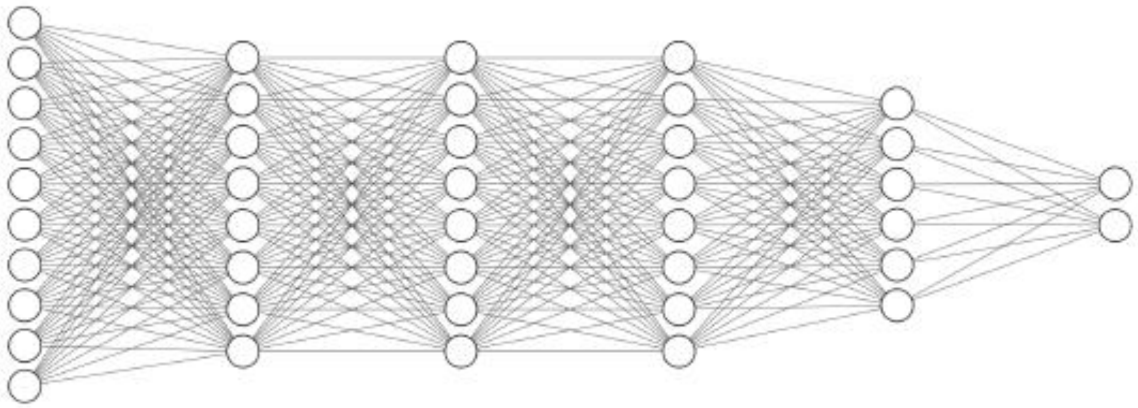
  to reach θ^* and q^*.
  Compute $\alpha(\theta_t, \theta^*) = \min\left\{1, \frac{\exp H(\theta^*, r^*)}{\exp H(\theta_t, r_t)}\right\}$.
  Accept θ^* as θ_{t+1} with probability $\alpha(\theta_t, \theta^*)$, else discard it.
end

From shallow to deep….



$$y = \sum_{j=1}^{m} \beta_j \psi(x'\gamma_j) + \epsilon$$
$$\epsilon \sim N(0, \sigma^2),$$
$$\psi(\eta) = \exp(\eta)/(1 + \exp(\eta))$$

(Shallow) Neural nets



$$\{f_0, f_1, ..., f_{L-1}\}$$

$$z_{l+1} = f_l(z_l, \gamma_l).$$

$$y = \sum_{j=1}^{m_L} \beta_j z_{L,j} + \epsilon$$
$$\epsilon \sim N(0, \sigma^2),$$

Deep neural nets

From shallow to deep

Chapter 7 https://datalab-icmat.github.io/courses_stats.html

- Convolutional
- Recurrent. LSTM and Transformers
- Autoencoders, VAE
- GANs
- …

From shallow to deep….

Start with arbitrary (β, γ, ν).
while *not convergence* **do**
  Given current (γ, ν), draw β from $p(\beta|\gamma, \nu, y)$ (a multivariate normal).
  for $j = 1, \ldots, m$, *marginalizing in* β *and given* ν **do**
    Generate a candidate $\tilde{\gamma}_j \sim g_j(\gamma_j)$.
    Compute $a(\gamma_j, \tilde{\gamma}_j) = \min\left(1, \frac{p(D|\tilde{\gamma}, \nu)}{p(D|\gamma, \nu)}\right)$ with $\tilde{\gamma} = (\gamma_1, \gamma_2, \ldots, \tilde{\gamma}_i, \ldots, \gamma_m)$.
    With probability $a(\gamma_j, \tilde{\gamma}_j)$ replace γ_j by $\tilde{\gamma}_j$. If not, preserve γ_j.
  end
  Given β and γ, replace ν based on their posterior conditionals:
  $p(\mu_\beta|\beta, \sigma_\beta)$ is normal; $p(\mu_\gamma|\gamma, S_\gamma)$, multivariate normal; $p(\sigma_\beta^{-2}|\beta, \mu_\beta)$, Gamma; $p(S_\gamma^{-1}|\gamma, \mu_\gamma)$, Wishart; $p(\sigma^{-2}|\beta, \gamma, y)$, Gamma.
end

Start with arbitrary $\theta_0 = (\beta_0, \gamma_0)$.
while *not convergence* **do**
  Given current θ_t and $q_t \sim \mathcal{N}(0, I)$, perform one or more leapfrog integration steps

$$q_{t+\frac{1}{2}} = q_t - \frac{\epsilon}{2}\nabla U(\theta_t)$$
$$\theta_{t+1} = \theta_t + \epsilon q_{t+\frac{1}{2}}$$
$$q_{t+1} = q_{t+\frac{1}{2}} - \frac{\epsilon}{2}\nabla U(\theta_{t+1})$$

  to reach θ^* and q^*.
  Compute $\alpha(\theta_t, \theta^*) = \min\left\{1, \frac{\exp H(\theta^*, r^*)}{\exp H(\theta_t, r_t)}\right\}$.
  Accept θ^* as θ_{t+1} with probability $\alpha(\theta_t, \theta^*)$, else discard it.
end

From shallow to deep

- From gradient descent to stochastic descent
- From MCMC to variational Bayes (ADVI)
  SG-MCMC
  hybrids (e.g. VIS)

Algorithm 4 **(A refined variational approximation sampler).**

Refinement phase:

while *not convergence* **do**

  Sample initial set of particles, $\theta_0 \sim q_{0,\phi}(\theta|D)$.

  Refine particles through sampler, $\theta_T \sim Q_{\eta,T}(\theta|\theta 0)$.

  Compute the ELBO objective (Equation 12).

  Update parameters ϕ, η through automatic differentiation on objective.

end

Inference phase, based on learned sampler parameters ϕ^*, η^*:

Sample an initial set of particles, $\theta_0 \sim q_{0,\phi^*}(\theta|D)$.

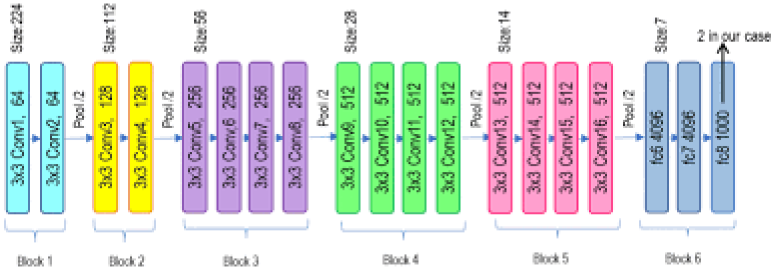
Use the MCMC sampler $\theta_T \sim Q_{\eta^*,T}(\theta|\theta_0)$ as $T \to \infty$.

Code at https://github.com/vicgalle/nn-review.

VGG-19 vs 3 hidden layer with 200 nodes with ReLU vs multinomial regression

Independent Gaussian priors over parameters (plus strong prior in VGG-19)

CIFAR-10 60000 32x32 color images 10 classes



Model	Test accuracy
Linear	38.10%
MLP	50.03%
VGG-19	93.29%

Code

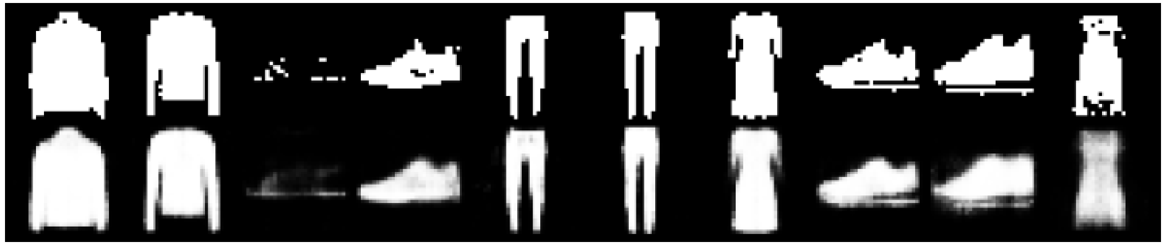
https://github.com/chris-nemeth/sgmcmc-review-paper

R package sgmcmc (Edward in Python)

https://github.com/vicgalle/nn-review.

https://github.com/vicgalle/vis

VIS examples



Systems with VI

Venture, WebPPL, Edward, **Stan**, PyMC3, Infer.net, Anglican

Challenges

Challenges

The limits of VIS

New hybrids

New bright ideas from physics

PPLs

Decision support

Bayesian Transformers

Bayesian graduation in LLMs