Probabilistic Deep Learning
BYÖYO 2017

Melih Kandemir

Özyeğin University, İstanbul, Turkey

21-08-2017

Why probabilistic machine learning?

Learn a lot from few cases, just like the human brain!
`https://vimeo.com/3235882`

Why probabilistic machine learning?

Charming results!

This is the paper that made neural networks shake the world

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

A. Krizhevsky et al., NIPS, 2012

Why probabilistic machine learning?

with unbelievably good results on the very challenging ILSRVC data set

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 1: Comparison of results on ILSVRC-2010 test set. In *italics* are best results achieved by others.

Why probabilistic machine learning?

using a method called **Dropout** introduced by yet another paper

Journal of Machine Learning Research 15 (2014) 1929-1958 Submitted 11/13; Published 6/14

Dropout: A Simple Way to Prevent Neural Networks from Overfitting

Nitish Srivastava NITISH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
Alex Krizhevsky KRIZ@CS.TORONTO.EDU
Ilya Sutskever ILYA@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
10 Kings College Road, Rm 3302
Toronto, Ontario, M5S 3G4, Canada.

Editor: Yoshua Bengio

Abstract

Deep neural nets with a large number of parameters are very powerful machine learning systems. However, overfitting is a serious problem in such networks. Large networks are also slow to use, making it difficult to deal with overfitting by combining the predictions of many different large neural nets at test time. Dropout is a technique for addressing this problem. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different "thinned" networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights. This significantly reduces overfitting and gives major improvements over other regularization methods. We show that dropout improves the performance of neural networks on supervised learning tasks in vision, speech recognition, document classification and computational biology, obtaining state-of-the-art results on many benchmark data sets.

Keywords: neural networks, regularization, model combination, deep learning

N. Srivastava, Journal of Machine Learning Research, 2014

Why probabilistic machine learning?

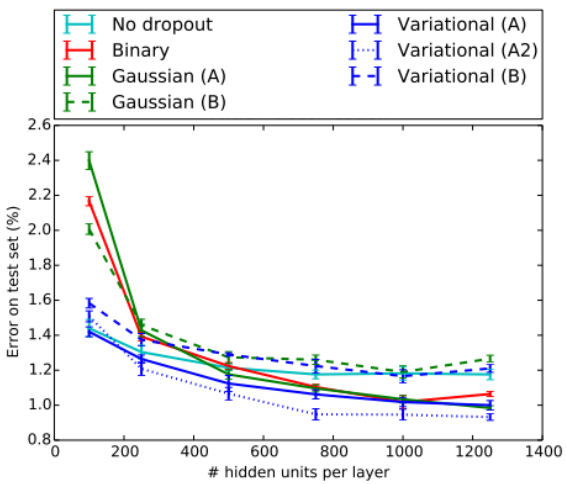
which builds on *uncertainty* of a neuron to be active or inactive

$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p), \\
\widetilde{\mathbf{y}}^{(l)} &= \mathbf{r}^{(l)} * \mathbf{y}^{(l)}, \\
z_i^{(l+1)} &= \mathbf{w}_i^{(l+1)} \widetilde{\mathbf{y}}^{l} + b_i^{(l+1)}, \\
y_i^{(l+1)} &= f(z_i^{(l+1)}).
\end{aligned}
$$

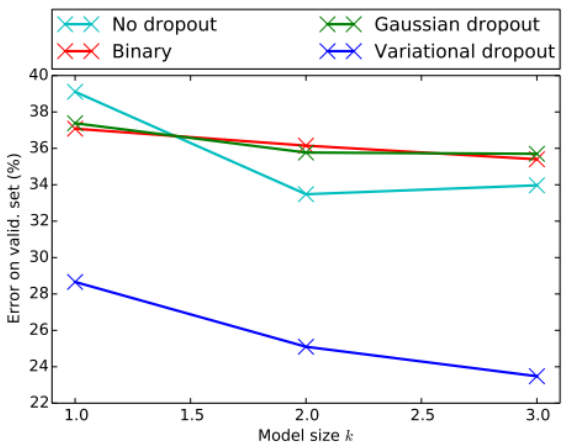
- **Mystery:** Why such a simple trick works that well
- **NOT mystery:** Why accounting for uncertainty leverages more robust predictions $\rightarrow$ **Probability Theory!**

Why probabilistic machine learning?

The more advanced the uncertainty account is, the better the predictions are



(a) Classification error on the MNIST dataset



(b) Classification error on the CIFAR-10 dataset

D. Kingma et al., NIPS, 2015

What is probability?

What is probability?

Is probability an **objective** or a **subjective** measure?

What if probability is objective?

$$p(e) = \lim_{n \to +\infty} \frac{n_e}{n}$$

- n_e: Number of times the event of interest occurs
- n: Number of trials

Can probability really be purely objective?

- How shall we handle $+\infty$? Sample set is limited.
- How do we know that our sample set is not biased?
- How do we know that the dice is fair?
- Is not making fairness or biasedness assumptions a subjective guess?
- Then why not quantify subjectivity?

asks a Bayesian, like de Finetti:

The classical view, based on physical considerations of symmetry, in which one should be *obliged* to give the same probability to such *symmetric* cases. But which symmetry? And, in any case, why? The original sentence becomes meaningful if reversed: the symmetry is probabilistically significant, in someone's opinion, if it leads him to assign the probabilities to such events.

de Finetti, 1970/74, Preface,xi-xii

Thomas Bayes the legend (1701-1761)

$$p(H|X) = \frac{p(X|H)p(H)}{p(X)}$$

H: Hypothesis
X: Measurement

Bayes' Theorem

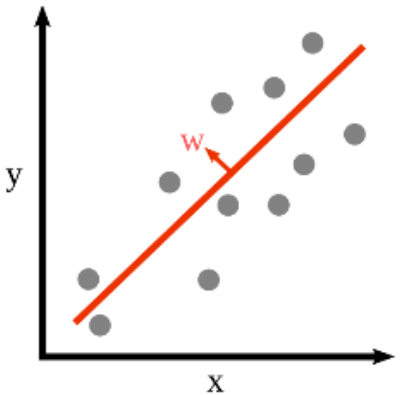
- $$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}$$

  $x \in \mathcal{X}$ is an observable in the sample space $\mathcal{X}$.
- θ is a set of model parameters. It is an index to a frequentist, and a random variable for a Bayesian.
- $p(x|\theta)$: likelihood (how do model parameters describe data?)
- $p(\theta)$: prior (what is our prior belief about model parameters?)
- $p(x)$: evidence (what is the likelihood of data *regardless of* the model parameters?)
- $p(\theta|x)$: posterior (how do model parameters distribute after observations are taken into account?)

Prior? What does it really mean?

Who do you expect to win the tennis game and why?

What does it mean to be Bayesian in machine learning?



	Model	Learning
Classical ML	$y_n = \mathbf{w}^T \mathbf{x}_n + \epsilon$	$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmin}} \; \|\|\mathbf{y} - \mathbf{w}^T \mathbf{X}\|\|_2^2$
Probabilistic ML	$p(y_n \|\mathbf{w}, \mathbf{x}_n) = \mathcal{N}(y_n \|\mathbf{w}^T \mathbf{x}_n, \sigma^2)$	$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}} \prod_{n=1}^{N} p(y_n \|\mathbf{w}, \mathbf{x}_n)$
Bayesian ML	$p(\mathbf{y} \|\mathbf{w}, \mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(y_n \|\mathbf{w}^T \mathbf{x}_n, \sigma^2)$ $p(\mathbf{w})$ ← Prior belief	$\underbrace{p(\mathbf{w} \|\mathbf{X}, \mathbf{y})}_{\text{Posterior}} = \frac{\overbrace{p(\mathbf{y} \|\mathbf{w}, \mathbf{X})}^{\text{Likelihood}} \times \overbrace{p(\mathbf{w})}^{\text{Prior}}}{\underbrace{p(\mathbf{y} \|\mathbf{X})}_{\text{Evidence}}}$

Motivation 1: De Finetti's Theorem

A sequence of random variables $(x_1, x_2, \cdots, x_N)$ is infinitely exchangeable iff, for any N,

$$p(x_1, x_2, \cdots, x_N) = \int \prod_{i=1}^{N} p(x_i|\theta) P(d\theta)$$

Here, $P(d\theta) = p(\theta)d\theta$ if θ has a density.
Implications:

- Exchangeability can be checked from right hand side.
- There must exist a parameter θ!
- There must exist a likelihood $p(x|\theta)$!
- There must exist a distribution P on θ

Motivation 2: Posterior predictive distribution

Given a posterior $p(\theta|x)$ and a new observation x^*, the posterior predictive distribution is

$$\begin{aligned} p(x^*|x) &= \int p(x^*|\theta)p(\theta|x)d\theta \\ &= \mathbb{E}_{p(\theta|x)}[p(x^*|\theta)] \end{aligned}$$

This distribution takes into account all possible values of θ with importance proportional to the probability of their occurrence. This virtue is called **model averaging** and exists *only* in Bayesian models!

The model selection problem

We are given two hypotheses that claim to explain a certain data set.

- Both give similar prediction quality. Which one should we choose?
- Which one explains the data better?

Motivation 3: Bayes quantifies model selection

Hypothesis 1 ($\mathcal{H}_1$): Likelihood: $p_{\mathcal{H}_1}(x|\theta_1)$, Prior: $p_{\mathcal{H}_1}(x|\theta_1)$
Hypothesis 2 ($\mathcal{H}_2$): Likelihood: $p_{\mathcal{H}_2}(x|\theta_2)$, Prior: $p_{\mathcal{H}_2}(x|\theta_2)$

We can alternatively treat the hypothesis as a random variable $\mathcal{H} = \{1, 2\}$ that determines the type of the distribution $p(\cdot)$:

$$p_{\mathcal{H}_1}(x|\theta_1) = p(x|\theta_1, \mathcal{H} = 1)$$
$$p_{\mathcal{H}_2}(x|\theta_2) = p(x|\theta_2, \mathcal{H} = 2)$$

Let us place a prior on also on the hypothesis variable. Unless we have a good reason, we are agnostic to both hypotheses: $P(\mathcal{H} = 1) = P(\mathcal{H} = 2)$.

Motivation 3: Bayes quantifies model selection

Now let us take into account all possible model parameter realizations for both hypotheses (i.e. calculate the evidence):

$$p(x|\mathcal{H}=1) = \int p(x|\theta_1, \mathcal{H}=1)p(\theta_1|\mathcal{H}=1)d\theta_1$$

$$p(x|\mathcal{H}=2) = \int p(x|\theta_2, \mathcal{H}=2)p(\theta_2|\mathcal{H}=2)d\theta_2$$

This operation is called **MARGINALIZING OUT**!
Nuisance Variable: A variable that we are not interested for our current analysis of interest.
Rule of Thumb: Marginalize out nuisance variables as much as you can!

Motivation 3: Bayes quantifies model selection

- Now apply Bayes theorem to calculate the posterior on hypotheses

$$P(\mathcal{H}|x) = \frac{p(x|\mathcal{H})P(\mathcal{H})}{p(x)}$$

- Choose the hypothesis with higher posterior probability. Compare $p(\mathcal{H} = 1|x)$ and $p(\mathcal{H} = 2|x)$.
  - Since $p(x)$ does not depend on $\mathcal{H}$, its magnitude does not have an effect on the comparison.
  - Since we chose a **uniform prior** on the hypotheses ($P(\mathcal{H} = 1) = P(\mathcal{H} = 2)$), the magnitude of $P(\mathcal{H})$ also does not have an effect.
- Hence, it suffices to calculate $p(x|\mathcal{H} = 1)/p(x|\mathcal{H} = 2)$. This metric is called the **Bayes factor** [Kass and Raftery, 1995]. Choose $\mathcal{H}_1$ if Bayes factor is greater than 1, choose $\mathcal{H}_2$ otherwise.
- The model **evidence** serves as a **quantitative** metric for model selection in the Bayesian setting.

Wait... all is well but...

How can we calculate the posterior

$p(\theta|\mathcal{X}) = \prod_{n=1}^{N} p(x_n|\theta)p(\theta)/p(\mathcal{X})$

especially $p(\mathcal{X})$?

In many cases you cannot. You can only approximate it.

And this is what this course is all about!

Maximum likelihood estimation

Given observed data X and the assumption that $X \sim p(X|\theta)$, the *maximum likelihood estimate (MLE)* is

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \; p(X|\theta)$$

Since $\log(\cdot)$ is monotonically increasing, we can equivalently solve

$$\hat{\theta} = \underset{\theta}{\text{argmax}} \; \log \; p(X|\theta)$$

What are priors for?

- To incorporate prior beliefs
- To avoid overfitting
  - Controlling model complexity:
    1. inducing sparsity=regularization
    2. marginal likelihood
  - Marginalization of model parameters (represented as a distribution, not a point estimate.
- To attain posterior uncertainty, which is essential for
  - active learning
  - decision making (medicine, finance, etc.)

Yin-Yang in statistics

Regularization: Frequentist way of being Bayesian
Non-informative priors: Bayesian way of being Frequentist

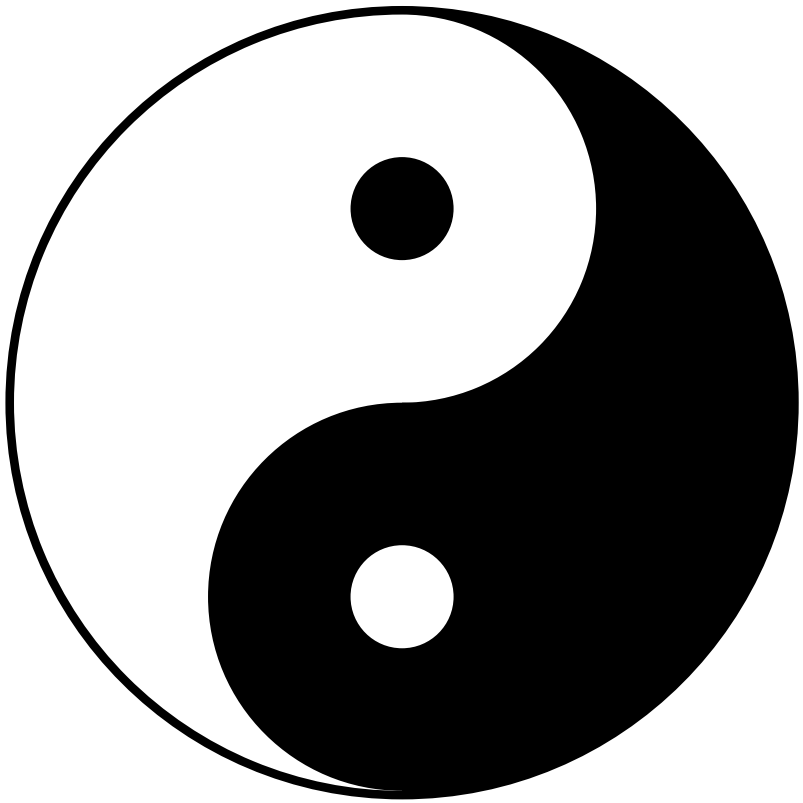


Image:
`https://en.wikipedia.org/wiki/Yin_and_yang`

Conjugate Prior [1]

If $p(\theta|\mathcal{D})$ is in the same family as $p(\theta)$, then $p(\theta)$ is called a *conjugate prior* for $p(\mathcal{D}|\theta)$.

[1] H. Raiffa and R. Schlaifer, Applied Statistical Decision Theory,1961.

Examples

- Beta $\times$ Binomial $=$ Beta
- Dirichlet $\times$ Multinomial $=$ Dirichlet
- Normal $\times$ Normal Mean $=$ Normal
- Gamma $\times$ Normal Variance $=$ Gamma
- Wishart $\times$ Normal Covariance $=$ Wishart

Regularization in linear regression

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(y_n - \sum_{d=1}^{D} w_d x_{nd} \right)^2$$

$$s.t \sum_{d=1}^{D} w_d^2 < t$$

Regularization in linear regression

$$\underset{\mathbf{w}}{\operatorname{argmin}} \underset{\lambda}{\operatorname{argmax}} \sum_{n=1}^{N} \Big(y_n - \sum_{d=1}^{D} w_d x_{nd} \Big)^2 + \lambda \Big(\sum_{d=1}^{D} w_d^2 - t \Big)$$

Regularization in linear regression

$$\underset{\mathbf{w}}{\operatorname{argmin}} \cancel{\operatorname{argmax}_{\lambda}} \sum_{n=1}^{N} \Big(y_n - \sum_{d=1}^{D} w_d x_{nd} \Big)^2 + \lambda \Big(\sum_{d=1}^{D} w_d^2 - \cancel{t} \Big)$$

l_2-norm regularization $\rightarrow$ ridge regression

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(y_n - \mathbf{w}^T \mathbf{x}_n \right)^2 + \lambda \underbrace{||\mathbf{w}||_2^2}_{\mathbf{w}^T \mathbf{w}}$$

l_2-norm regularization $\rightarrow$ ridge regression

$$\underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(y_n - \mathbf{w}^T \mathbf{x}_n \right)^2 + \lambda \underbrace{||\mathbf{w}||_2^2}_{\mathbf{w}^T\mathbf{w}}$$

Solution:

$$\begin{aligned}
\nabla_{\mathbf{w}} \Big\{ (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \Big\} &= 0 \\
\mathbf{w}^T \mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{y}^T \mathbf{X} \mathbf{w} + \mathbf{y}^T \mathbf{y} + \lambda \mathbf{w}^T \mathbf{w} &= 0 \\
2\mathbf{X}^T \mathbf{X} \mathbf{w} - 2\mathbf{X}^T \mathbf{y} + 2\lambda \mathbf{w} &= 0 \\
(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}) \mathbf{w} &= \mathbf{X}^T \mathbf{y} \\
(\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y} &= \hat{\mathbf{w}}
\end{aligned}$$

Thanks to $\lambda \mathbf{I}$, the matrix inverse exists even though $N < D$.

Maximum A Posteriori (MAP) estimation

What if we are not interested in model uncertainty? Then it suffices to seek for the mode of the posterior:

$$\begin{aligned}\underset{\mathbf{w}}{\operatorname{argmax}}\ \log\ p(\mathbf{w}|\mathbf{X},\mathbf{y}) &= \underset{\mathbf{w}}{\operatorname{argmax}}\ \log \frac{p(\mathbf{y}|\mathbf{w},\mathbf{X})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\ \log\ p(\mathbf{y}|\mathbf{w},\mathbf{X}) + \log p(\mathbf{w}) \underbrace{- \log p(\mathbf{y}|\mathbf{X})}_{\text{const}} \\ &= \underset{\mathbf{w}}{\operatorname{argmax}}\ \log\ p(\mathbf{y}|\mathbf{w},\mathbf{X}) + \log p(\mathbf{w})\end{aligned}$$

The found mode is called the *Maximum A Posteriori (MAP)* estimate of the model. This is a technique a true Bayesian largely avoids, though there are specific cases where it is useful.

MAP Example

Let us take again the Bayesian linear regression case.

$$\mathbf{y}|\mathbf{w}, \mathbf{X} \sim \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I}),$$
$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}).$$

Our aim is to solve

$$\begin{aligned}
&\operatorname*{argmax}_{\mathbf{w}} \log \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I}) + \log \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}) \\
&= \operatorname*{argmax}_{\mathbf{w}} \left\{ -\frac{\beta}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} \right\} \\
&= \operatorname*{argmax}_{\mathbf{w}} \left\{ -\frac{1}{2}\mathbf{w}^T\Big(\beta\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\Big)\mathbf{w} + \beta\mathbf{y}^T\mathbf{X}\mathbf{w} \right\}
\end{aligned}$$

MAP Example

Set the gradient of the variable of interest to zero:

$$\nabla_{\mathbf{w}} \left\{ -\frac{1}{2}\mathbf{w}^T \Big(\beta \mathbf{X}^T\mathbf{X} + \alpha \mathbf{I}\Big)\mathbf{w} + \beta \mathbf{y}^T \mathbf{X}\mathbf{w} \right\} \triangleq 0$$

$$-\Big(\beta \mathbf{X}^T\mathbf{X} + \alpha \mathbf{I}\Big)\mathbf{w} + \beta \mathbf{X}^T\mathbf{y} \triangleq 0$$

Solving for $\mathbf{w}$ and rearranging β in the same way as above gives

$$\hat{\mathbf{w}} \triangleq \left(\mathbf{X}^T\mathbf{X} + \frac{\alpha}{\beta}\mathbf{I}\right)^{-1} \mathbf{X}^T\mathbf{y}.$$

Once more we recapitulate the ridge regression. This will have consequences far up to how dropout relates to complicated Bayesian models!

KL divergence is a dissimilarity measure

Considering that $-\log x$ is a convex function,

$$\begin{aligned}\mathbb{KL}[p||q] &= -\int p(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x} \\ &\geq -\log \underbrace{\int p(\mathbf{x}) \frac{q(\mathbf{x})}{p(\mathbf{x})} d\mathbf{x}}_{1} = 0.\end{aligned}$$

Because $-\log x$ is a *strictly* convex function (i.e. equality holds only at intersection points),

$$p(\mathbf{x}) = q(\mathbf{x}) \iff \mathbb{KL}[p||q] = 0.$$

Hence, KL divergence is a dissimilarity *metric* between two densities. Note that $\mathbb{KL}[p||q] \neq \mathbb{KL}[q||p]$.

Calculus of variations

Typically we have scalars or vectors as variables. Then we operate on mappings from these variables to other entities. For instance in $f(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$, the vector $\mathbf{x}$ is our variable of interest and $f(\cdot)$ is a *function* of it.

There are some cases where we take *functions* as *variables* of interest and operate on mappings from functions to other entities:

$$\mathbb{F} : f(\mathbf{x}) \to \mathbb{R}.$$

Such mappings are called **functionals**. One example is the *KL divergence*. The branch of mathematics that has functionals in its focus is named as the *calculus of variations*.

What if we have non-conjugate priors?

Assume we are given a data set $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ and a Bayesian model

$$\mathbf{X}|\boldsymbol{\theta} \sim \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\theta}),$$
$$\boldsymbol{\theta} \sim p(\boldsymbol{\theta}).$$

with a non-conjugate prior $p(\boldsymbol{\theta})$ on the set of *latent variables* wrt likelihood $p(\mathbf{x}_n|\boldsymbol{\theta})$. We are interested in the posterior

$$p(\boldsymbol{\theta}|\mathbf{X}),$$

which does not have a closed-form solution. What shall we do then?

Approximating the posterior

Choose a $q(\boldsymbol{\theta}|\boldsymbol{\gamma})$, a density parameterized by $\boldsymbol{\gamma}$, and construct an optimization problem to make $q(\boldsymbol{\theta}|\boldsymbol{\gamma})$ as similar as possible to the true posterior $p(\boldsymbol{\theta}|\mathbf{X})$.

But what sort of an optimization problem would be suitable?

Hint: Put the pieces together.

How about this?

$$\underset{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}{\operatorname{argmin}} \mathbb{KL}[p(\boldsymbol{\theta}|\mathbf{X})||q(\boldsymbol{\theta}|\boldsymbol{\gamma})]$$

Did we solve the problem now?

How about this?

Not quite!

$$\mathbb{KL}[p(\boldsymbol{\theta}|\mathbf{X})||q(\boldsymbol{\theta}|\boldsymbol{\gamma})] = \int p(\boldsymbol{\theta}|\mathbf{X}) \log \frac{p(\boldsymbol{\theta}|\mathbf{X})}{q(\boldsymbol{\theta}|\boldsymbol{\gamma})} d\boldsymbol{\theta}.$$

The loss function depends on $p(\boldsymbol{\theta}|\mathbf{X})$, which we do not know. We ended up with the point we started from!

How about the other way around?

$$\underset{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}{\operatorname{argmin}} \mathbb{KL}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\mathbf{X})]$$

- At least worthwhile going forward. Approximating the posterior by solving this optimization problem is called **Variational Bayes**!
- Actually, there are ways to go forward from $\mathbb{KL}[p(\boldsymbol{\theta}|\mathbf{X})||q(\boldsymbol{\theta}|\boldsymbol{\gamma})]$ as well by introducing further approximations. This is called *Expectation Propagation*. We will cover that approach towards the end of the semester.

Variational Bayes

$$\begin{aligned}
\mathbb{KL}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\mathbf{X})] &= \int q(\boldsymbol{\theta}|\boldsymbol{\gamma}) \log \frac{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}{\underbrace{p(\boldsymbol{\theta}|\mathbf{X})}_{\frac{p(\boldsymbol{\theta},\mathbf{X})}{p(\mathbf{X})}}} d\boldsymbol{\theta} \\
&= \int q(\boldsymbol{\theta}|\boldsymbol{\gamma}) \log \frac{q(\boldsymbol{\theta}|\boldsymbol{\gamma})p(\mathbf{X})}{p(\boldsymbol{\theta},\mathbf{X})} d\boldsymbol{\theta} \\
&= \int q(\boldsymbol{\theta}|\boldsymbol{\gamma}) \log q(\boldsymbol{\theta}|\boldsymbol{\gamma}) d\boldsymbol{\theta} \\
&+ \int q(\boldsymbol{\theta}|\boldsymbol{\gamma}) \log p(\mathbf{X}) d\boldsymbol{\theta} \\
&- \int q(\boldsymbol{\theta}|\boldsymbol{\gamma}) \log p(\boldsymbol{\theta},\mathbf{X}) d\boldsymbol{\theta}
\end{aligned}$$

Variational Bayes

$$\mathbb{KL}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\mathbf{X})] = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log q(\boldsymbol{\theta}|\boldsymbol{\gamma})]}_{-\mathbb{H}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\boldsymbol{\theta}]} + \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\mathbf{X})]}_{\log p(\mathbf{X})} - \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\boldsymbol{\theta}, \mathbf{X})]$$

Arranging the terms, we get the interesting outcome below

$$\underbrace{\log p(\mathbf{X})}_{\text{const}} = \underbrace{\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\boldsymbol{\theta}, \mathbf{X})] + \mathbb{H}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\boldsymbol{\theta}]}_{\mathcal{L}} + \underbrace{\mathbb{KL}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\mathbf{X})]}_{\geq 0}.$$

Hence, $\mathcal{L}$ is a lower bound to the log of the evidence. Hence it is called the *Evidence Lower Bound (ELBO)*. ELBO equals to the log-evidence iff $q(\boldsymbol{\theta}|\boldsymbol{\gamma}) = p(\boldsymbol{\theta}|\mathbf{X})$.

Variational Bayes: Inference $\rightarrow$ Optimization

$$\underset{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}{\operatorname{argmin}}\ \mathbb{KL}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta}|\mathbf{X})] \equiv \underset{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}{\operatorname{argmax}}\ \mathcal{L}$$

Inference as optimization

Let us take a closer look at the generic form and contemplate on the feasibility of the approach

$$\begin{aligned}
&\underset{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}{\operatorname{argmax}}\ \mathcal{L} \\
&= \underset{\boldsymbol{\gamma}}{\operatorname{argmax}} \left\{ \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\mathbf{x}_n|\boldsymbol{\theta})] + \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\boldsymbol{\theta})] + \mathbb{H}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\boldsymbol{\theta}] \right\} \\
&= \underset{\boldsymbol{\gamma}}{\operatorname{argmax}} \left\{ \sum_{n=1}^{N} \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\mathbf{x}_n|\boldsymbol{\theta})] - \mathbb{KL}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta})] \right\}
\end{aligned}$$

Calculate $\mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\log p(\mathbf{x}_n|\boldsymbol{\theta})]$ and look up $\mathbb{H}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[\boldsymbol{\theta}]$ or alternatively $\mathbb{KL}_{q(\boldsymbol{\theta}|\boldsymbol{\gamma})}[q(\boldsymbol{\theta}|\boldsymbol{\gamma})||p(\boldsymbol{\theta})]$. Take the gradient of the ELBO wrt $\boldsymbol{\gamma}$ and optimize.

Devise a BNN

$$p(\mathbf{Y}|\mathbf{X},\mathbf{w}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{y}_n|f(\mathbf{x}_n,\mathbf{w}),\tau^{-1}\mathbf{I}),$$

$$p(\mathbf{w}) = \prod_{l=1}^{L}\prod_{i=1}^{I_l}\prod_{j=1}^{I_{l+1}} \left(\pi\mathcal{N}(w_{ij}^{(l)}|0,\sigma_1^2) + (1-\pi)\mathcal{N}(w_{ij}^{(l)}|0,\sigma_2^2)\right).$$

Choosing $\sigma_1 > \sigma_2$ provides a heavy tail. If $\sigma_2 << 1$, we attain a variant of a *spike-and-slab* prior: Activate a neuron if really required, push it hard towards zero otherwise. Fitting σ_1 and σ_2 also to data is not necessarily beneficial.

Apply Variational Bayes

Maximize the ELBO

$$\mathcal{L} = \sum_{n=1}^{N} \mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w})] - \mathbb{KL}[q(\mathbf{w})||p(\mathbf{w})]$$

by taking gradient steps

$$\begin{aligned}\nabla_\phi \mathcal{L} &= \sum_{n=1}^{N} \nabla_\phi \mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w})] - \nabla_\phi \mathbb{E}\Big[\log q(\mathbf{w})||p(\mathbf{w})\Big] \\ &= \nabla_\phi \mathbb{E}_{q(\mathbf{w})}\left[-\frac{\tau^{-1}}{2}\sum_{n=1}^{N}||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w})||_2^2 - \log\frac{q(\mathbf{w})}{p(\mathbf{w})}\right],\end{aligned}$$

where

$$q(\mathbf{w}|\phi) = \prod_{l=1}^{L}\prod_{i=1}^{I_l}\prod_{j=1}^{I_{l+1}} \mathcal{N}(w_{ij}^{(l)}|\mu_{ij}^{(l)}, \gamma_{ij}^{(l)})$$

Apply Variational Bayes

$$\nabla_\phi \mathcal{L} = \sum_{n=1}^{N} \nabla_\phi \mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{y}_n|\mathbf{x}_n, \mathbf{w})] - \nabla_\phi \mathbb{E}\Big[\log q(\mathbf{w})||p(\mathbf{w})\Big]$$

$$= \nabla_\phi \mathbb{E}_{q(\mathbf{w})} \Bigg[\underbrace{-\frac{\tau^{-1}}{2} \sum_{n=1}^{N} ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w})||_2^2 - \log \frac{q(\mathbf{w})}{p(\mathbf{w})}}_{f(\mathbf{w},\theta)} \Bigg],$$

$q(\epsilon) = \mathcal{N}(\epsilon|0, 1)$, and $t(\theta, \epsilon) \rightarrow w = \mu + \sqrt{\gamma}\,\epsilon$, resulting in

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{q(\epsilon)} \Bigg[\nabla_\phi \Bigg\{ -\frac{\tau^{-1}}{2} \sum_{n=1}^{N} ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w})||_2^2 + \log p(\mathbf{w}) - \log q(\mathbf{w}) \Bigg\} \Bigg],$$

The gradient

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{q(\epsilon)} \Bigg[-\frac{\tau^{-1}}{2} \sum_{n=1}^{N} \nabla_\phi ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w})||_2^2 + \nabla_\phi \log p(\mathbf{w}) - \nabla_\phi \log q(\mathbf{w}) \Bigg].$$

Here,

$$\nabla_\phi ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w})||_2^2 \leftarrow \text{Standard Backprop},$$

$$\nabla_\phi \log p(\mathbf{w}) \leftarrow \frac{\partial \log p(\mathbf{w})}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \phi},$$

$$\nabla_\phi \log q(\mathbf{w}|\phi) \leftarrow \frac{\partial \log q(\mathbf{w}|\phi)}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \phi} + \frac{\partial \log q(\mathbf{w}|\phi)}{\partial \phi}.$$

Eventually, we arrived at a regularized version of the standard backprop gradient!

Resolve the integral with Monte Carlo samples

$$\nabla_\phi \mathcal{L} = \mathbb{E}_{q(\epsilon)}\Bigg[-\frac{\tau^{-1}}{2}\sum_{n=1}^{N}\nabla_\phi ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w}^{(s)})||_2^2 + \nabla_\phi \log p(\mathbf{w}) - \nabla_\phi \log q(\mathbf{w}|\phi)\Bigg].$$

$$\approx \frac{1}{S}\sum_{s=1}^{S}\Bigg[-\frac{\tau^{-1}}{2}\sum_{n=1}^{N}\nabla_\phi ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w}^{(s)})||_2^2 + \nabla_\phi \log p(\mathbf{w}^{(s)}) - \nabla_\phi \log q(\mathbf{w}^{(s)}|\phi)\Bigg],$$

where $\phi = \{\mu, \gamma = \log(1+\exp(\rho))^2\}$ and

$$\mathbf{w}^{(s)} = \mu + \sqrt{\gamma}\,\epsilon^{(s)}, \quad \epsilon^{(s)} \sim \mathcal{N}(\epsilon^{(s)}|0,1).$$

The Bayes by Backprop Algorithm

Set (i.e. define in a deep learning algorithm as functions):

$$f(\mathbf{w}, \phi) = -\frac{\tau^{-1}}{2} \sum_{n=1}^{N} ||\mathbf{y}_n - f_{NN}(\mathbf{x}_n, \mathbf{w})||_2^2 + \log p(\mathbf{w}) - \log q(\mathbf{w}|\phi),$$

$$w_{ij}^{(l)} = \mu_{ij}^{(l)} + \log(1 + \exp(\rho_{ij}^{(l)}))\epsilon, \quad \phi_{ij}^{(l)} = (\mu_{ij}^{(l)}, \rho_{ij}^{(l)}), \quad \forall i, j, l$$

For every neuron $w_{ij}^{(l)}$ in the network, repeat:

- Sample $\epsilon \sim \mathcal{N}(\epsilon|0, 1)$.
- Calculate gradients

$$\nabla_{\mu_{ij}^{(l)}} = \frac{\partial f(\mathbf{w}, \phi)}{\partial \mathbf{w}} + \frac{\partial f(\mathbf{w}, \phi)}{\partial \mu_{ij}^{(l)}},$$

$$\nabla_{\rho_{ij}^{(l)}} = \frac{\partial f(\mathbf{w}, \phi)}{\partial \mathbf{w}} \frac{\epsilon}{1 + \exp(-\rho_{ij}^{(l)})} + \frac{\partial f(\mathbf{w}, \phi)}{\partial \rho_{ij}^{(l)}}.$$

- Take gradient steps: $\mu_{ij}^{(l)} \leftarrow \mu_{ij}^{(l)} + \alpha \nabla_{\mu_{ij}^{(l)}}$, $\rho_{ij}^{(l)} \leftarrow \rho_{ij}^{(l)} + \alpha \nabla_{\rho_{ij}^{(l)}}$.

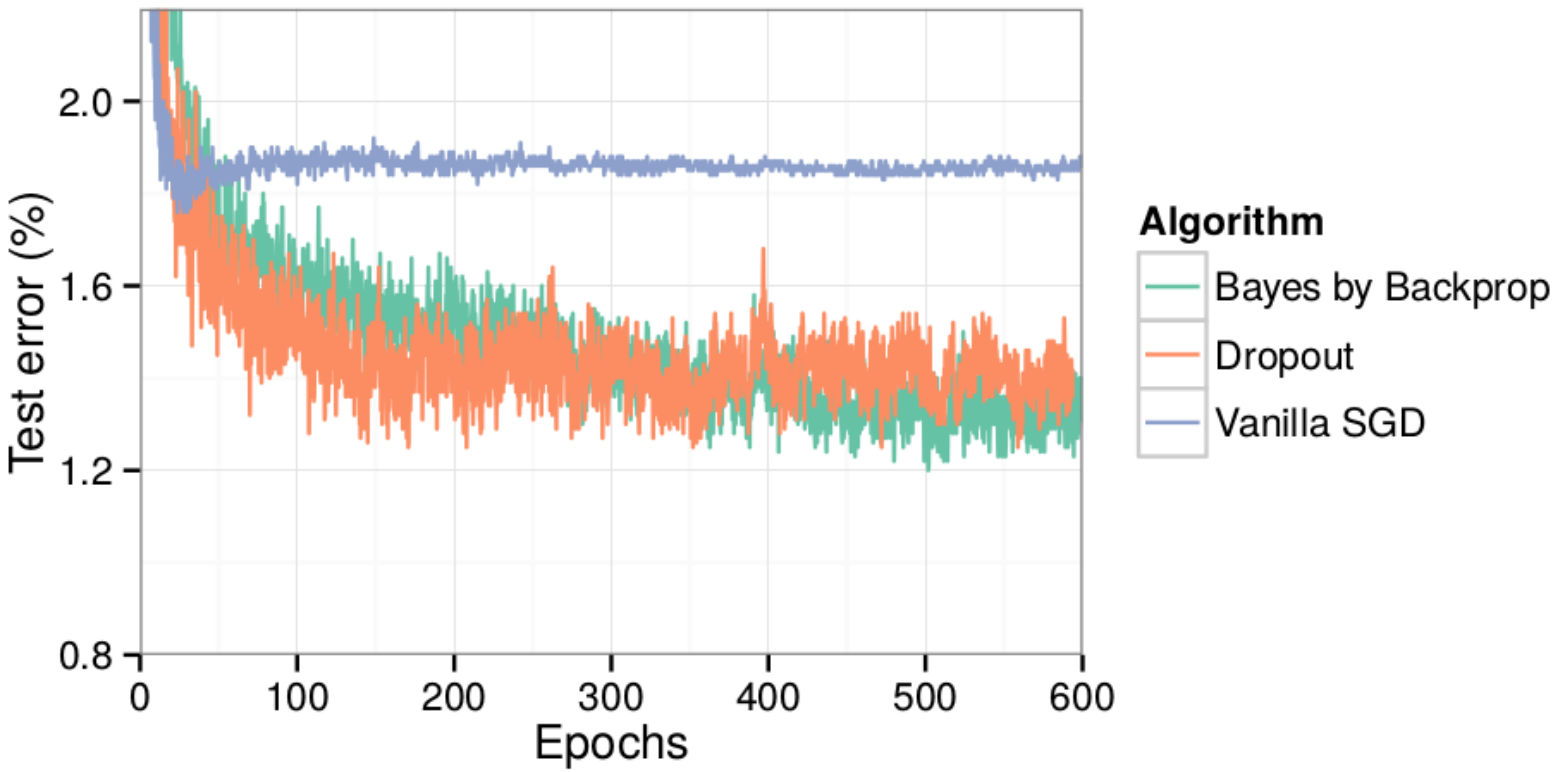
Results (Blundell et al., ICML, 2015)

Table 1. Classification Error Rates on MNIST. $\star$ indicates result used an ensemble of 5 networks.

Method	# Units/Layer	# Weights	Test Error
SGD, no regularisation (Simard et al., 2003)	800	1.3m	1.6%
SGD, dropout (Hinton et al., 2012)			$\approx$ 1.3%
SGD, dropconnect (Wan et al., 2013)	800	1.3m	**1.2**%*
SGD	400	500k	1.83%
	800	1.3m	1.84%
	1200	2.4m	1.88%
SGD, dropout	400	500k	1.51%
	800	1.3m	1.33%
	1200	2.4m	1.36%
Bayes by Backprop, Gaussian	400	500k	1.82%
	800	1.3m	1.99%
	1200	2.4m	2.04%
Bayes by Backprop, Scale mixture	400	500k	1.36%
	800	1.3m	1.34%
	1200	2.4m	**1.32**%

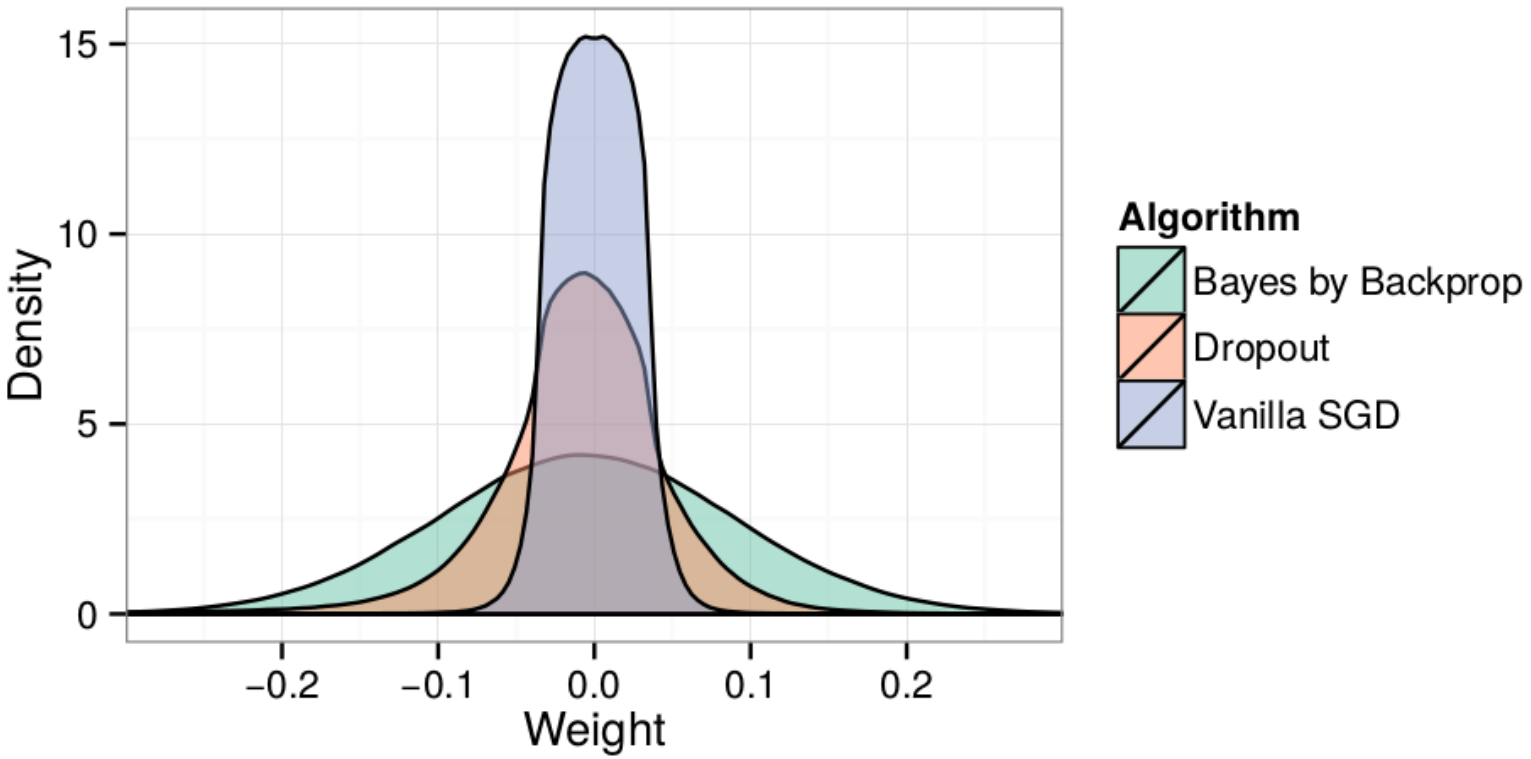
Results (Blundell et al., ICML, 2015)

- Standard Stochastic Gradient Descent jumps fast into a local minimum, but gets stuck there.
- Bayes by Backprop seeks for more stable solutions, thanks to the additional stochasticity on the weights.



Results (Blundell et al., ICML, 2015)

- Bayes by Backprop enhances diversity in weights (i.e. the gradients steps manage to reach at further locations from the initialization point).



Results (Blundell et al., ICML, 2015)

- Sort neurons by Signal-to-Noise Ratio (SNR):

$$|\mu_{ij}^{(l)}|/\log(1+\exp(\rho)).$$

- Prune the lowest portion.
- No drop in performance until 75%.
- Not surprisingly, the majority of the neurons are redundant!

Table 2. Classification Errors after Weight pruning

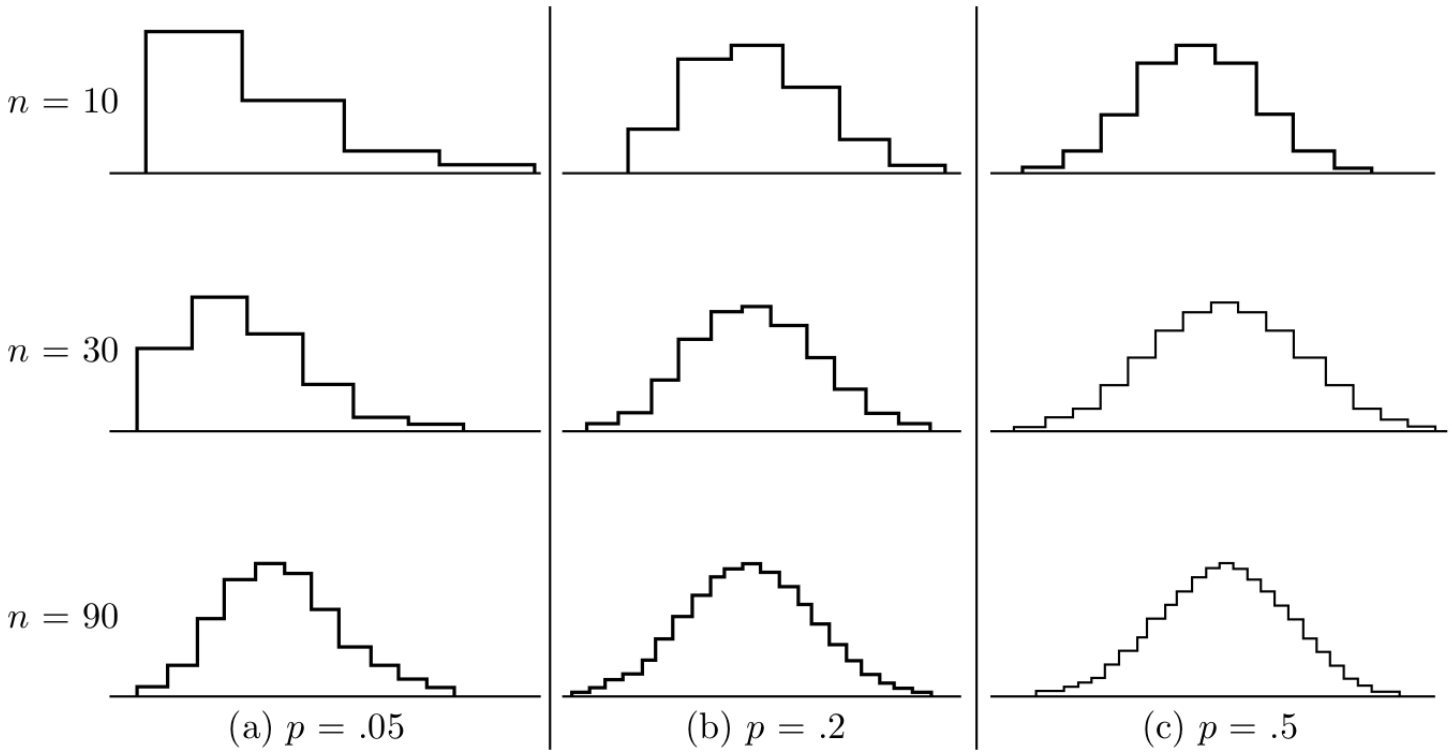
Proportion removed	**# Weights**	**Test Error**
0%	2.4m	1.24%
50%	1.2m	1.24%
75%	600k	1.24%
95%	120k	1.29%
98%	48k	1.39%

The Central Limit Theorem

Theorem 1 (CLT). Let $X_i, i = 1, 2, \cdots,$ be i.i.d. with $E(X_i) = \xi$, $Var(X_i) = \sigma^2 < \infty$, and empirical mean $\bar{X}$. Then

$$\sqrt{n}(\bar{X} - \xi)/\sigma \xrightarrow{L} \mathcal{N}(0, 1).$$

CLT Illustrated



Histograms of $\sqrt{n}\left(\frac{X}{n} - p\right) \Big/ \sqrt{pq}$ for $X \sim Bernoulli(p)$ and $q = 1 - p^2$.

[2] **Figure.** E.L. Lehmann, Elements of Large-Sample Theory, Springer, 1998

The dropout sequence

Let b and y be linear and non-linear activations of a hidden neuron, respectively. The standard dropout generates a random sequence b_s and y_s by the process below

$$\begin{aligned}\mathbf{z}^{(s)} &\sim \prod_{d=1}^{d} Bernoulli(z_d^{(s)}|1-p),\\ b_s &= f(\mathbf{z}^{(s)}),\\ y_s &= \sigma(b_s),\end{aligned}$$

where $f(\mathbf{z}) = \sum_{d=1}^{D} w_d z_d x_d, \quad \mathbf{x} = [x_1, \cdots, x_D]$ is the vector of inputs to the neuron coming from the previous layer, $\sigma(\cdot)$ is any activation function, and p is the dropout rate. Then we have

$$y_s \xrightarrow{L} E_{\mathbf{z}}[y].$$

The dropout sequence at infinity

As both b_s and y_s satisfy the conditions of the CLT with weak Lyapunov conditions, applying Theorem 3, we get a convergent-in-probability sequence applying the following process

$$\begin{aligned}\epsilon^{(s)} &\sim \mathcal{N}(0,1),\\ b_s &= E_{\mathbf{z}}[f(\mathbf{z})] + \sqrt{Var_{\mathbf{z}}[f(\mathbf{z})]}\epsilon_d^{(s)},\\ y_s &= \sigma(b_s),\end{aligned}$$

where

$$E_{\mathbf{z}}[f(\mathbf{z})] = \sum_{d=1}^{D} w_d(1-p)x_d,$$

$$Var_{\mathbf{z}}[f(\mathbf{z})] = \sum_{d=1}^{D} Var_{z_d}[w_d z_d x_d] = \sum_{d=1}^{D}(w_d x_d)^2(1-p)p.$$

Remark: Now we draw one single sample *per output*, not *per input dimension* as before!

The dropout sequence illustrated at large sample sizes

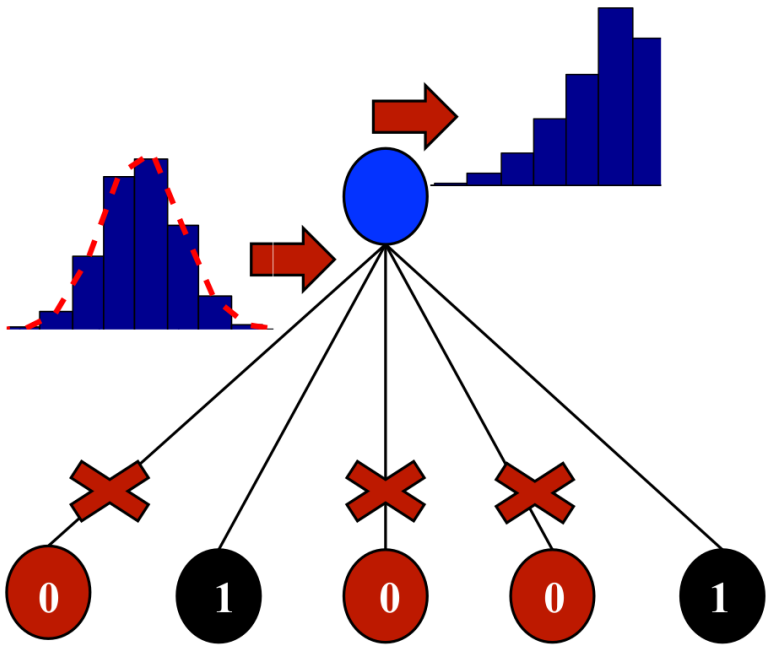


Figure. S.I. Wang, C.D. Manning, Fast Dropout Training, ICML, 2013

Time complexity of Gaussian Dropout

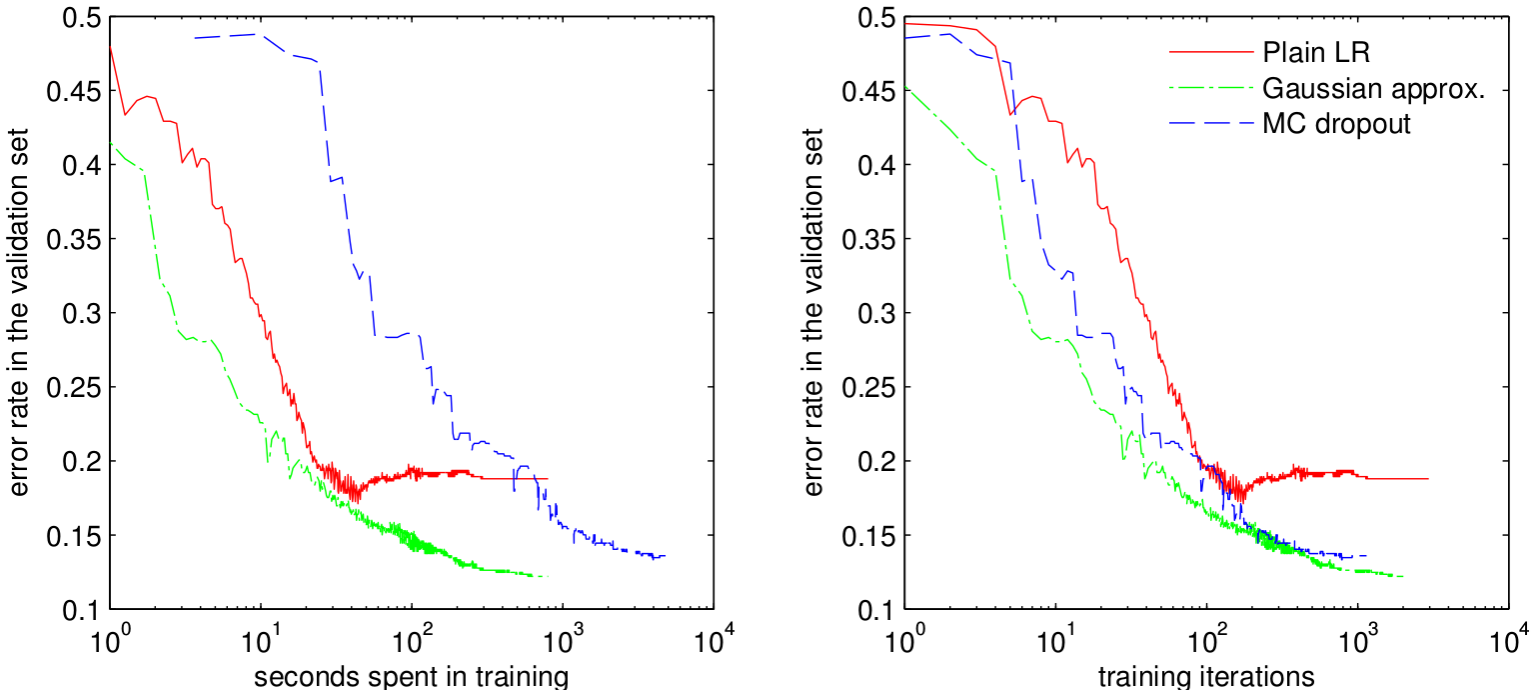


Figure. S.I. Wang, C.D. Manning, Fast Dropout Training, ICML, 2013

Convergence of Gaussian Dropout

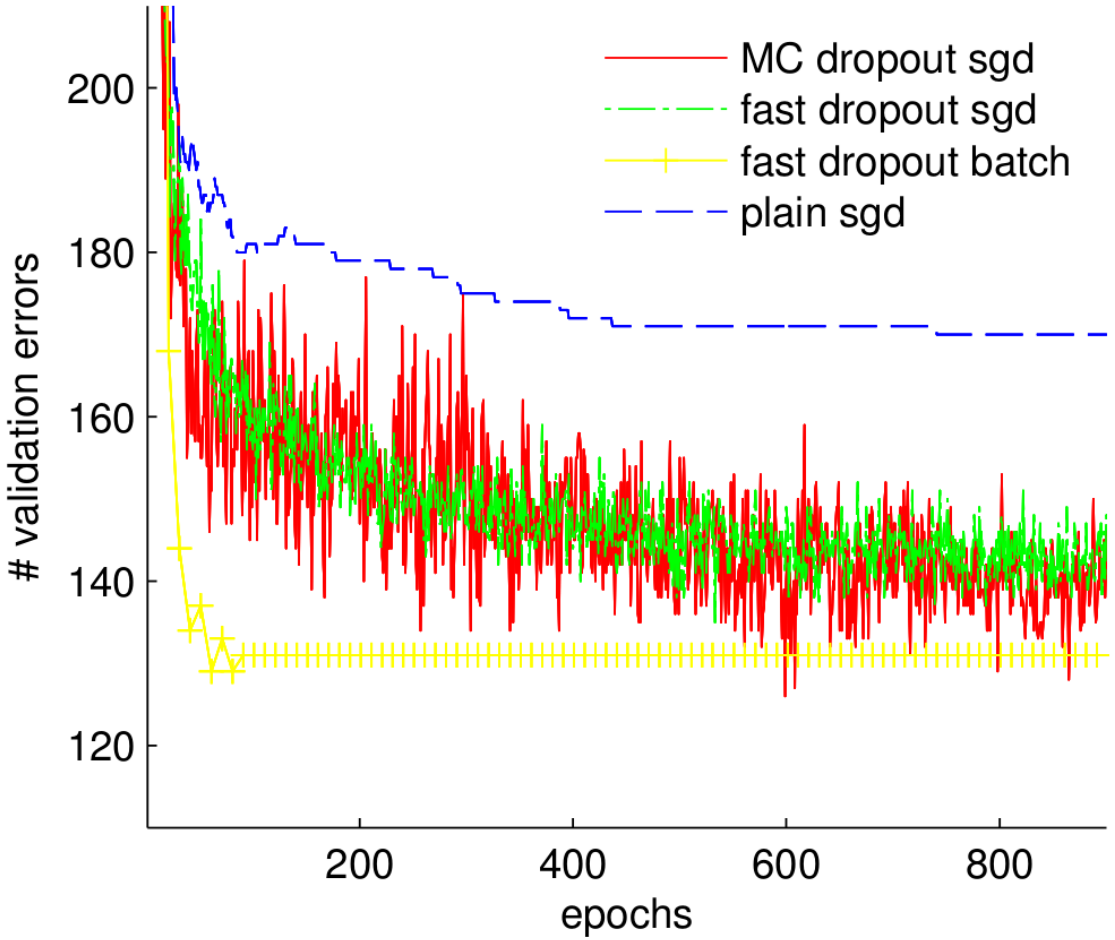


Figure. S.I. Wang, C.D. Manning, Fast Dropout Training, ICML, 2013

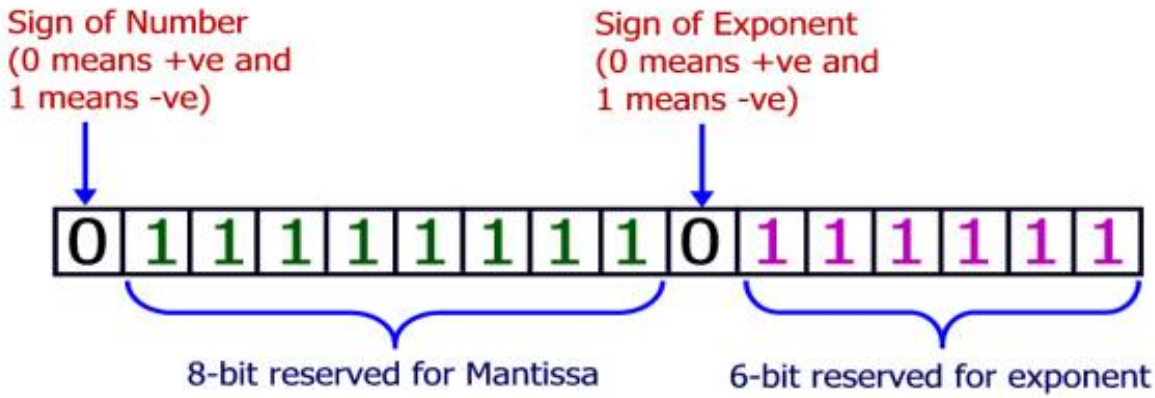
Linking Gaussian Dropout to Variational Bayes

Assign each v a factor distribution $q(w)$ and apply the reparameterization trick for fast sampling. Also choose a uniform (improper) prior:

$$p(w) \sim \exp(c).$$

Linking Gaussian Dropout to Variational Bayes

Applying the floating point arithmetic,



we equivalently get

$$
\begin{aligned}
w &= s_i a_i, \\
s_i &\sim Bernoulli(0.5) \; on \; \{-1, +1\}, \\
p(log|a_i|) &\propto c.
\end{aligned}
$$

Figure.
`www.bnuoj.com/v3/problem_show.php?pid=20227`

The implied KL divergence term

Doing the math (details less essential for our scope), we attain the following KL divergence term, which eventually depends on α

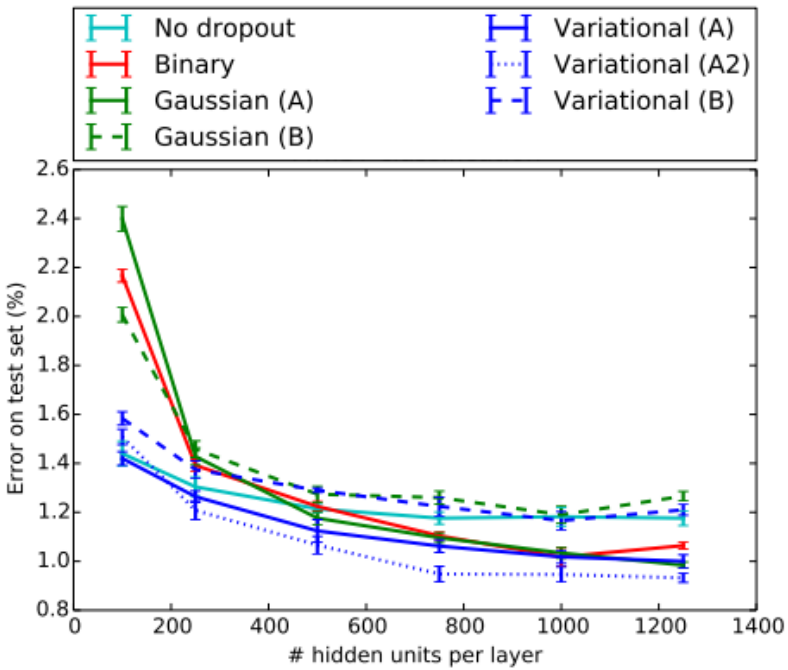
$$-\mathbb{KL}[q(w)||p(w)] \approx \text{const} + 0.5\log(\alpha) + c_1\alpha + c_2\alpha^2 + c_3\alpha^3,$$

where

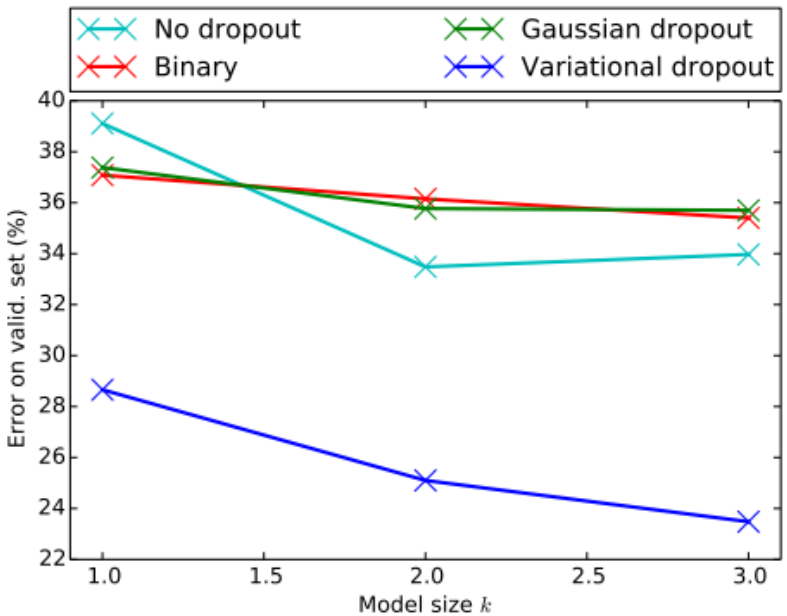
$$\begin{aligned} c_1 &= 1.16145124, \\ c_2 &= 1.50204118, \\ c_3 &= 0.58629921. \end{aligned}$$

Adding this tiny term to the loss function, we can learn the dropout rate also from data!!!

Performance of Variational Dropout



(a) Classification error on the MNIST dataset



(b) Classification error on the CIFAR-10 dataset

Figure. D. Kingma, T. Salimans, M. Welling, Variational Dropout and the Local Reparameterization Trick, NIPS, 2015

Problems with standard MCMCs

Metropolis and Metropolis-Hastings:

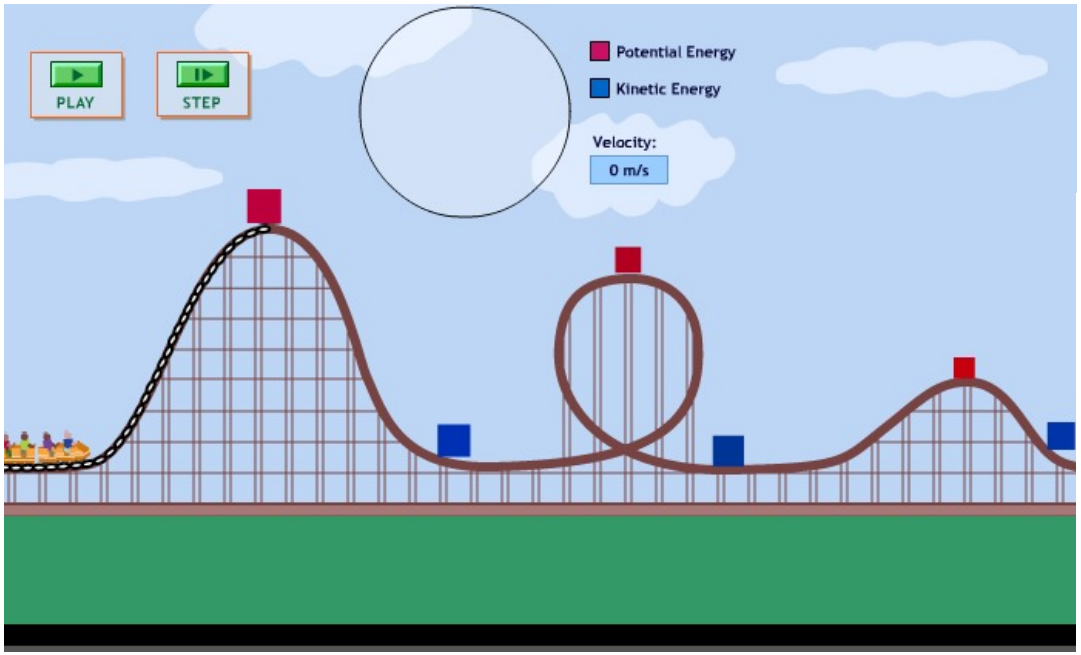
- The proposal distribution is agnostic about the target model.
- The proposals tend to perform random walk, hence shoot blindly.
- The outcome often is an unacceptably low acceptance rate.

Gibbs:

- Requires conditional distributions available in closed form (not tractable even for logistic regression).

Remedy $\Rightarrow$ Incorporate model curvature into the sampling scheme.

Back to the past: Potential and Kinetic Energy



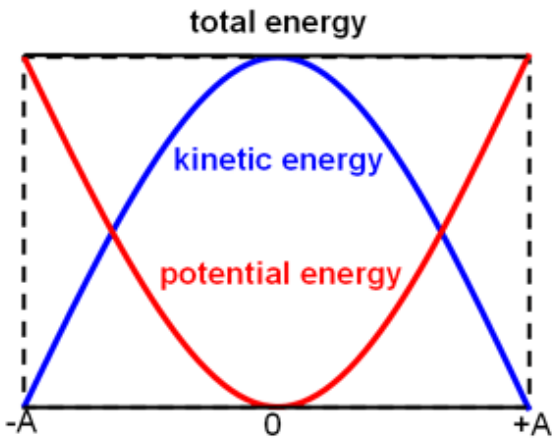


Figure. `http://99daveva31893.blogspot.com.tr/2013/05/potential-and-kinetic-energy.html.jpg`

For physicists

- **System state:** Position of the roller coaster θ.
- **Potential energy:** A score proportional to the height of the roller coaster $U(\theta)$.
- **Kinetic energy:** A score proportional to the speed (actually momentum) r of the roller coaster $K(r)$.
- **Total energy:** Rule that governs how potential and kinetic energies are related $H(\theta, r) = K(r) + U(\theta)$.

Here $H(\theta, r)$ is also called the *Hamiltonian function*.

For us machine learners

- **System state:** Position on the explored space of latent variables θ.
- **Potential energy:** A score *proportional* (i.e. no normalization constant) to the posterior we aim to sample from $U(\theta)$.
- **Kinetic energy:** Auxiliary variables r that animate the system state as fast as an auxiliary score $K(r)$.
- **Total energy:** Rule $H(\theta, r) = K(r) + U(\theta)$ assuring that the animated system is a Markov chain which has the posterior as the stationary distribution.

Hamiltonian dynamics

A physically-inspired way to model the Markov chain dynamics:
For ith latent variable (particle for physicists), we have

$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial r_i}, \tag{1}$$

$$\frac{dr_i}{dt} = -\frac{\partial H}{\partial \theta_i}, \tag{2}$$

meaning

- (1) A particle moves as fast as the change in kinetic energy.
- (2) The momentum of the particle increases as fast as the decrease in its height.

We choose

$$K(r) = \frac{1}{2} r^T M^{-1} r$$

$$U(\theta) = -\log p(\theta) - \log p(\mathcal{D}|\theta) = -\log p(\theta) - \sum_{x \in \mathcal{D}} \log p(x|\theta).$$

Understanding Hamiltonian dynamics: The hockey puck analogy

$$\frac{d\theta_i}{dt} = \frac{\partial H}{\partial r_i}, \qquad \frac{dr_i}{dt} = -\frac{\partial H}{\partial \theta_i},$$

- Assume a hockey puck (θ) placed on a rugged surface of *frictionless* ice ($U(\theta)$).
- We let the puck move by pushing it towards an arbitrary direction $K(r)$.
- As the surface is frictionless, the puck will keep moving forever (so we can sample as much as we want!).
- On a flat surface, the puck will keep constant speed.
- Under positive slope $\partial H/\partial \theta_i > 0$ it will climb and then lose speed and vice versa.
- The puck swings between steep regions (modes) and keeps speed on plateaus!

How to solve the Hamiltonian system of differential equations

No analytically tractable solution for interesting models.
Approximate by finite difference.

Way 1: Euler's method:

$$r_{t+\epsilon} \leftarrow r_t + \epsilon \frac{dr_t}{dt} = r_t - \epsilon \nabla_{\theta_t} U(\theta)$$
$$\theta_{t+\epsilon} \leftarrow \theta_t + \epsilon \frac{d\theta_t}{dt} = \theta_t + \epsilon M^{-1} r_t$$

- **Advantage**: Ultimately trivial.
- **Disadvantage**: The finite difference approximation will diverge from the true gradient at every time step.

How to solve the Hamiltonian system of differential equations

Way 2: *Modified* Euler's method:

$$r_{t+\epsilon} \leftarrow r_t + \epsilon \frac{dr_t}{dt} = r_t - \epsilon \nabla_{\theta_t} U(\theta)$$
$$\theta_{t+\epsilon} \leftarrow \theta_t + \epsilon \frac{d\theta_t}{dt} = \theta_t + \epsilon M^{-1} r_{t+\epsilon}$$

- Coupling the two updates brings about a charming improvement in accuracy, yet does not solve all the problems.
- The finite difference approximation is still not sufficiently accurate.
- Stronger coupling is required.

How to solve the Hamiltonian system of differential equations

Way 3: The **Leapfrog** method:

$$r_{t+\epsilon/2} \leftarrow r_t + (\epsilon/2)\frac{dr_t}{dt} = r_t - (\epsilon/2)\nabla_{\theta_t} U(\theta)$$

$$\theta_{t+\epsilon} \leftarrow \theta_t + \epsilon\frac{d\theta_t}{dt} = \theta_t + \epsilon M^{-1} r_{t+\epsilon/2}$$

$$r_{t+\epsilon} \leftarrow r_{t+\epsilon/2} + (\epsilon/2)\frac{dr_t}{dt} = r_{t+\epsilon/2} - (\epsilon/2)\nabla_{\theta_t} U(\theta)$$

Here is how the Leapfrog method takes one step ahead:

- Take half a step with the right leg.
- Take a full step with the left leg.
- Take half a step with the right leg.

Euler's method and Leapfrog

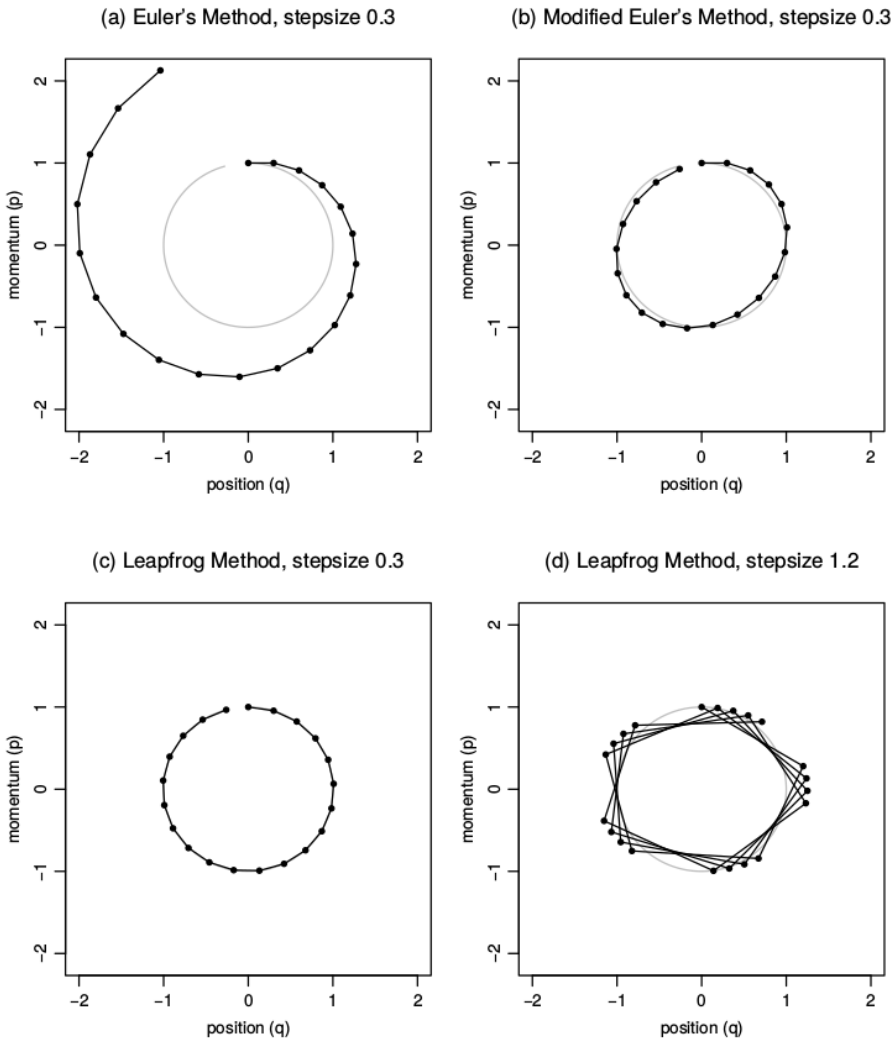


Figure. R. Neal, MCMC using Hamiltonian Dynamics, 2011

Hamiltonian Monte Carlo Sampling

Algorithm 1: Hamiltonian Monte Carlo

Input: Starting position $\theta^{(1)}$ and step size ϵ
for $t = 1, 2 \cdots$ **do**
  Resample momentum r
  $r^{(t)} \sim \mathcal{N}(0, M)$
  $(\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$
  Simulate discretization of Hamiltonian dynamics in Eq. (4):
  $r_0 \leftarrow r_0 - \frac{\epsilon}{2} \nabla U(\theta_0)$
  for $i = 1$ ***to*** m **do**
    $\theta_i \leftarrow \theta_{i-1} + \epsilon M^{-1} r_{i-1}$
    $r_i \leftarrow r_{i-1} - \epsilon \nabla U(\theta_i)$
  end
  $r_m \leftarrow r_m - \frac{\epsilon}{2} \nabla U(\theta_m)$
  $(\hat{\theta}, \hat{r}) = (\theta_m, r_m)$
  Metropolis-Hastings correction:
  $u \sim \text{Uniform}[0, 1]$
  $\rho = e^{H(\hat{\theta}, \hat{r}) - H(\theta^{(t)}, r^{(t)})}$
  if $u < \min(1, \rho)$, **then** $\theta^{(t+1)} = \hat{\theta}$
end

Figure. T. Chen et al. Stochastic Gradient Hamiltonian Monte Carlo, ICML, 2014

The Metropolis Correction

The numerical inaccuracies resulting from the finite difference approximation accumulate and cause the chain diverge from the target distribution. We can avoid this by accessing the true posterior every now and then.
Define the Hamiltonian joint distribution

$$\pi(\theta, r) \propto \exp\Big(- U(\theta) - \frac{1}{2} r^T M^{-1} r \Big).$$

Applying the Metropolis criterion, we get the acceptance probability

$$\min\left(1, \frac{\pi(\hat{\theta}, \hat{r})}{\pi(\theta_0, r_0)}\right) = \min\left(1, \underbrace{e^{H(\hat{\theta},\hat{r}) - H(\theta_0, r_0)}}_{\rho}\right).$$

Remark: Accessing the posterior $p(\theta|\mathcal{D})$ is nice, but could be unacceptably expensive for some models, such as deep neural nets!

Random walk versus Hamiltonian dynamics

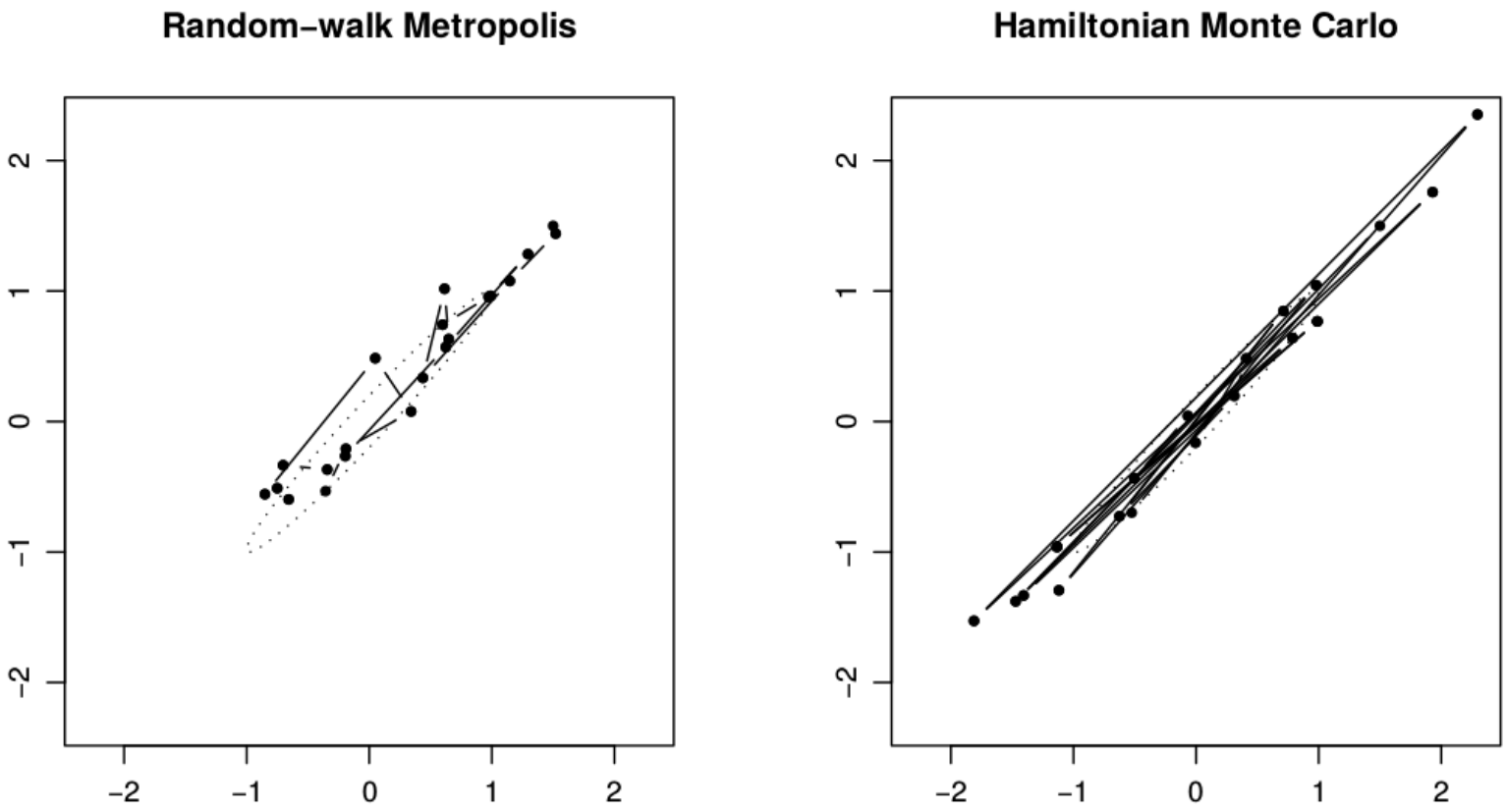


Figure. R. Neal, MCMC using Hamiltonian Dynamics, 2011

Random walk versus Hamiltonian dynamics

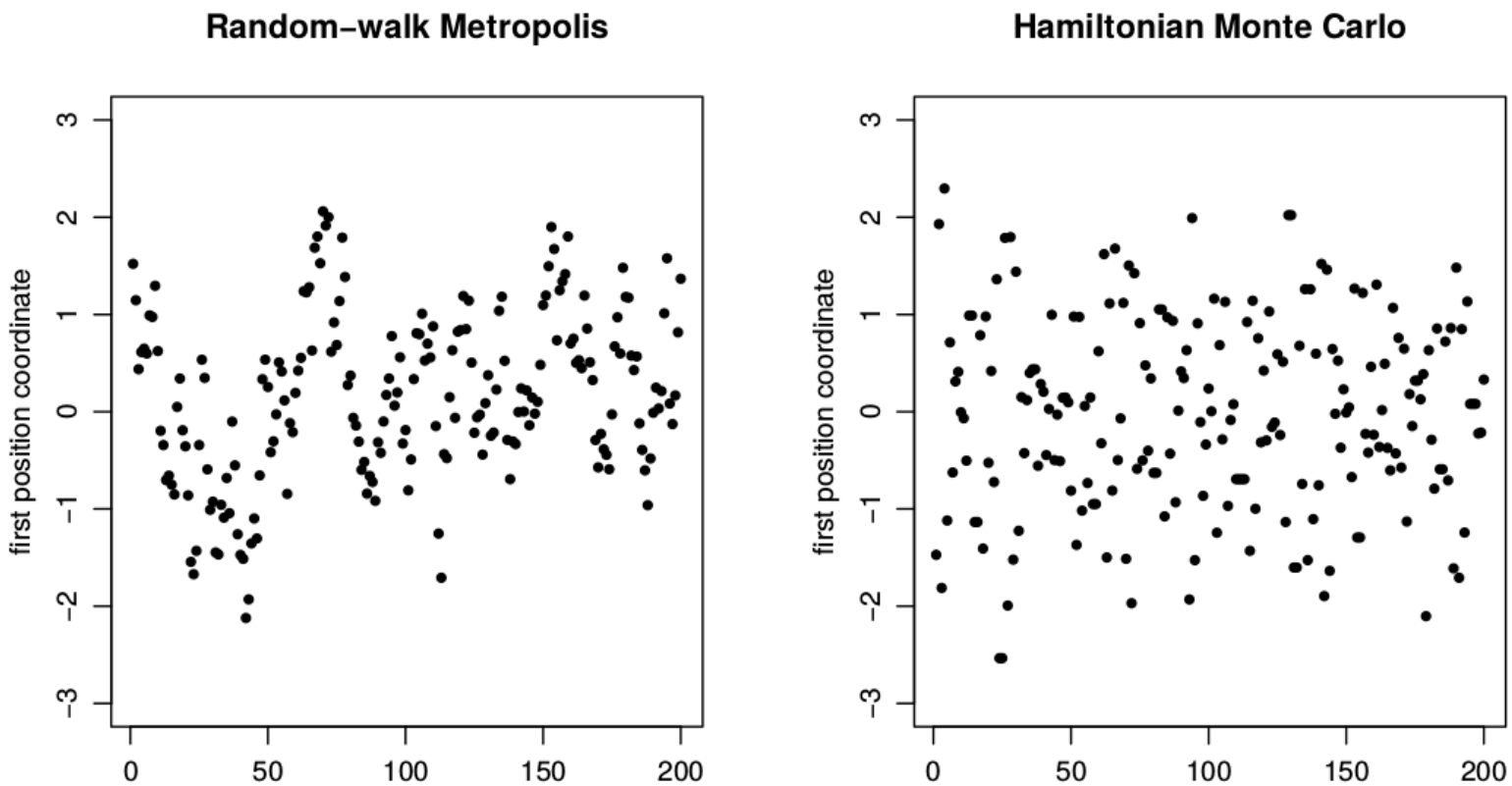


Figure. R. Neal, MCMC using Hamiltonian Dynamics, 2011

Random walk versus Hamiltonian dynamics

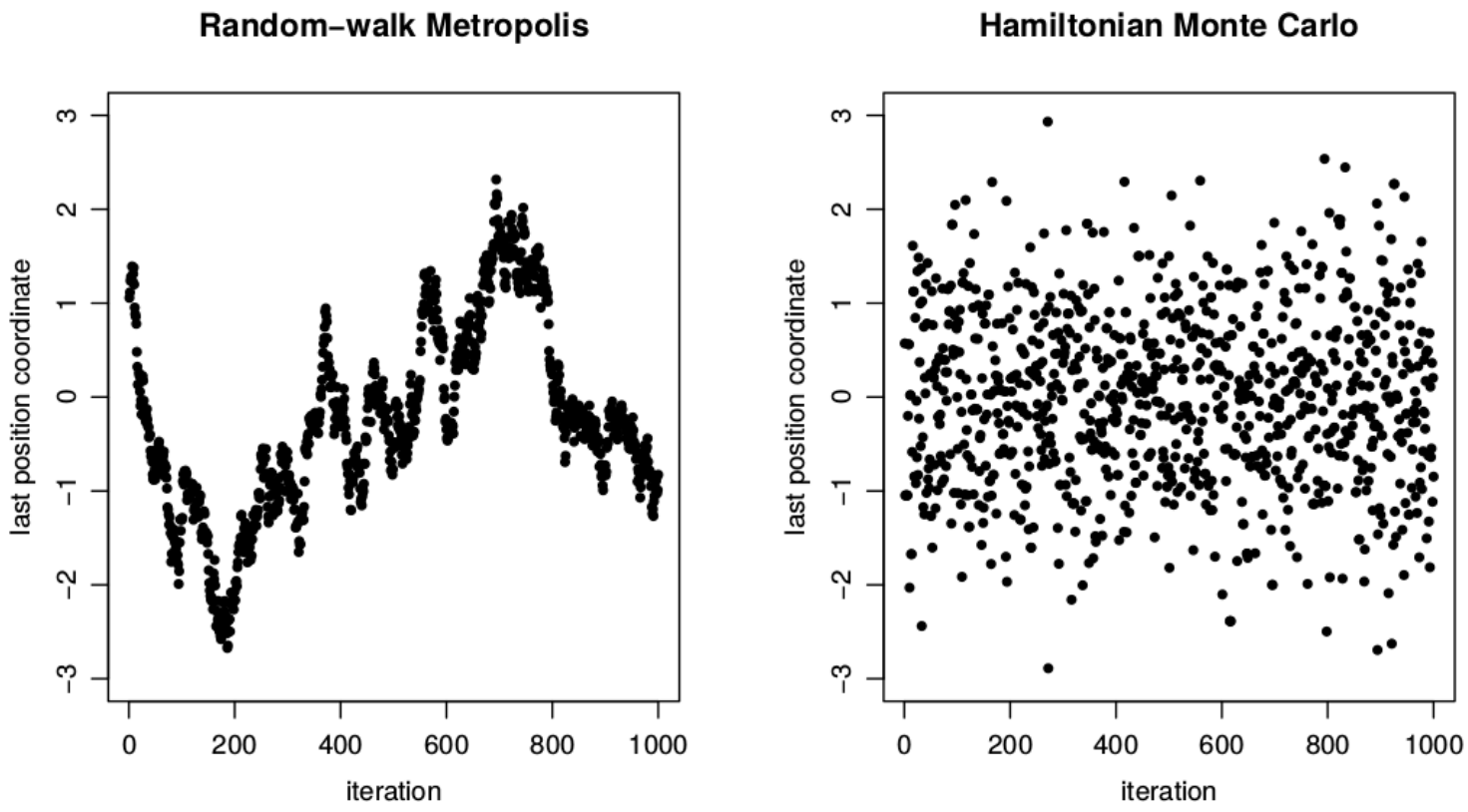


Figure. R. Neal, MCMC using Hamiltonian Dynamics, 2011

Stochastic Gradient HMC

- HMC is all great for accurate posterior inference but every jump requires a full pass on the the data, which is no longer practical in the present age.
- The naive way out is to switch from exact gradient to stochastic gradient for the potential energy (i.e. the model):

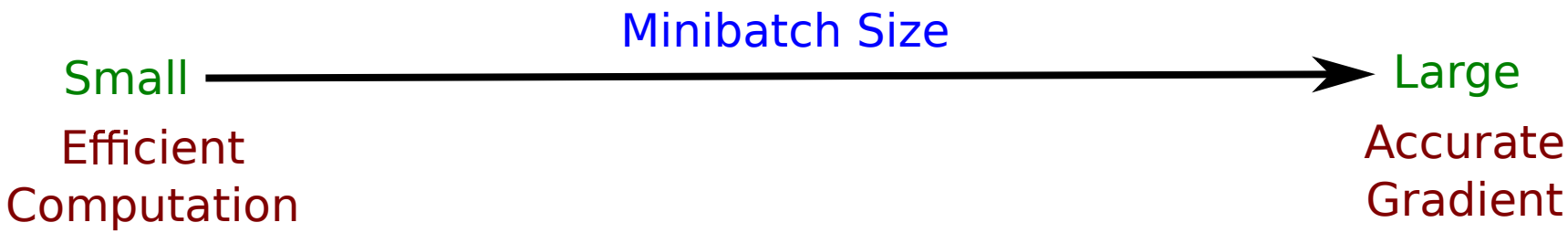
$$\nabla \tilde{U}(\theta) = -\frac{|\mathcal{D}|}{|\tilde{\mathcal{D}}|} \sum_{x \in \tilde{\mathcal{D}}} \nabla \log p(x|\theta) - \nabla \log p(\theta),$$

  where $\tilde{\mathcal{D}}$ is a random *minibatch*.
- The stochastic gradient trick is mostly used for very large data sets, which results in excessively many iterations. When iterated large enough times, the Central Limit Theorem will govern the stochastic gradient noise

$$\nabla \tilde{U}(\theta) - \nabla U(\theta) \sim \mathcal{N}(0, V),$$

  where V is the noise covariance.

The notorious efficiency-accuracy tradeoff



Naive Stochastic Gradient HMC

With a little sloppy notation, let us write

$$\nabla\tilde{U}(\theta) - \nabla U(\theta) \sim \mathcal{N}(0, V) \Rightarrow \nabla\tilde{U}(\theta) = \nabla U(\theta) + \mathcal{N}(0, V).$$

With the added stochastic gradient noise, the ϵ-discretized momentum update turns into

$$\begin{aligned}
\Delta r = -\epsilon\nabla\tilde{U}(\theta) &= -\epsilon(\nabla U(\theta) + p), \quad p = Ls, \quad s \sim \mathcal{N}(0, 1), \\
&= -\epsilon\nabla U(\theta) + \epsilon p, \quad \epsilon p = \epsilon L s, \quad s \sim \mathcal{N}(0, 1), \\
&= -\epsilon\nabla U(\theta) + \mathcal{N}(0, \epsilon^2 V).
\end{aligned}$$

where $V = LL^T$ is the Cholesky decomposition of V.

Naive Stochastic Gradient HMC

Casting $\epsilon \to 0$, we attain the dynamical system below

$$d\theta = M^{-1} r dt,$$
$$dr = -\nabla U(\theta) dt + \mathcal{N}(0, 2B(\theta) dt),$$

where $B(\theta) = \frac{1}{2}\epsilon V(\theta)$.

- The hockey puck on the ice surface is now under random wind!
- The Hamiltonian system preserves entropy when under exact gradients.
- The extra entropy coming from the stochastic gradient breaks the balance and accumulates entropy at every iteration.
- Consequently, the dynamic system above converges to a uniform distribution!

Stochastic Gradient HMC with Friction

A new term is added to the system to counter the stochastic gradient entropy:

$$d\theta = M^{-1} r dt,$$
$$dr = -\nabla U(\theta) dt - BM^{-1} r dt + \mathcal{N}(0, 2B(\theta) dt).$$

- The hockey puck is now on an ice surface, under random wind, and is exerted some friction!
- With the friction term added, the system is again able to preserve entropy, hence the equilibrium distribution is the posterior.
- The dynamical system above is known by physicists as *second-order Langevin dynamics* [Wang & Uhlenbeck, 1945].

Stochastic Gradient HMC

Algorithm 2: Stochastic Gradient HMC

for $t = 1, 2 \cdots$ **do**
  optionally, resample momentum r *as*
  $r^{(t)} \sim \mathcal{N}(0, M)$
  $(\theta_0, r_0) = (\theta^{(t)}, r^{(t)})$
  simulate dynamics in Eq.(13)*:*
  for $i = 1$ ***to*** m **do**
    $\theta_i \leftarrow \theta_{i-1} + \epsilon_t M^{-1} r_{i-1}$
    $r_i \leftarrow r_{i-1} - \epsilon_t \nabla \tilde{U}(\theta_i) - \epsilon_t C M^{-1} r_{i-1}$
      $+\mathcal{N}(0, 2(C - \hat{B})\epsilon_t)$
  end
  $(\theta^{(t+1)}, r^{(t+1)}) = (\theta_m, r_m)$, no M-H step
end

Figure. T. Chen et al. Stochastic Gradient Hamiltonian Monte Carlo, ICML, 2014.

Naive versus principled SGHMC

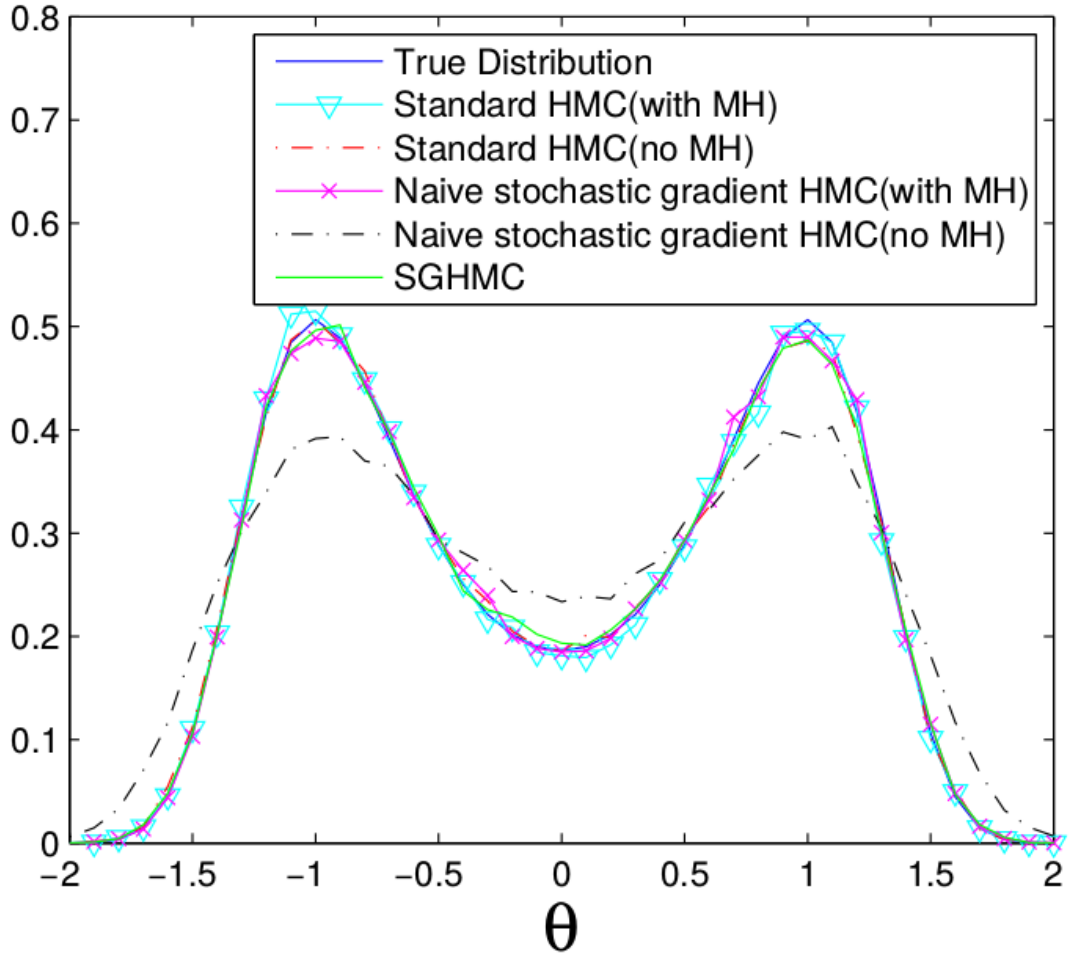


Figure. T. Chen et al. Stochastic Gradient Hamiltonian Monte Carlo, ICML, 2014.

Naive versus principled SGHMC

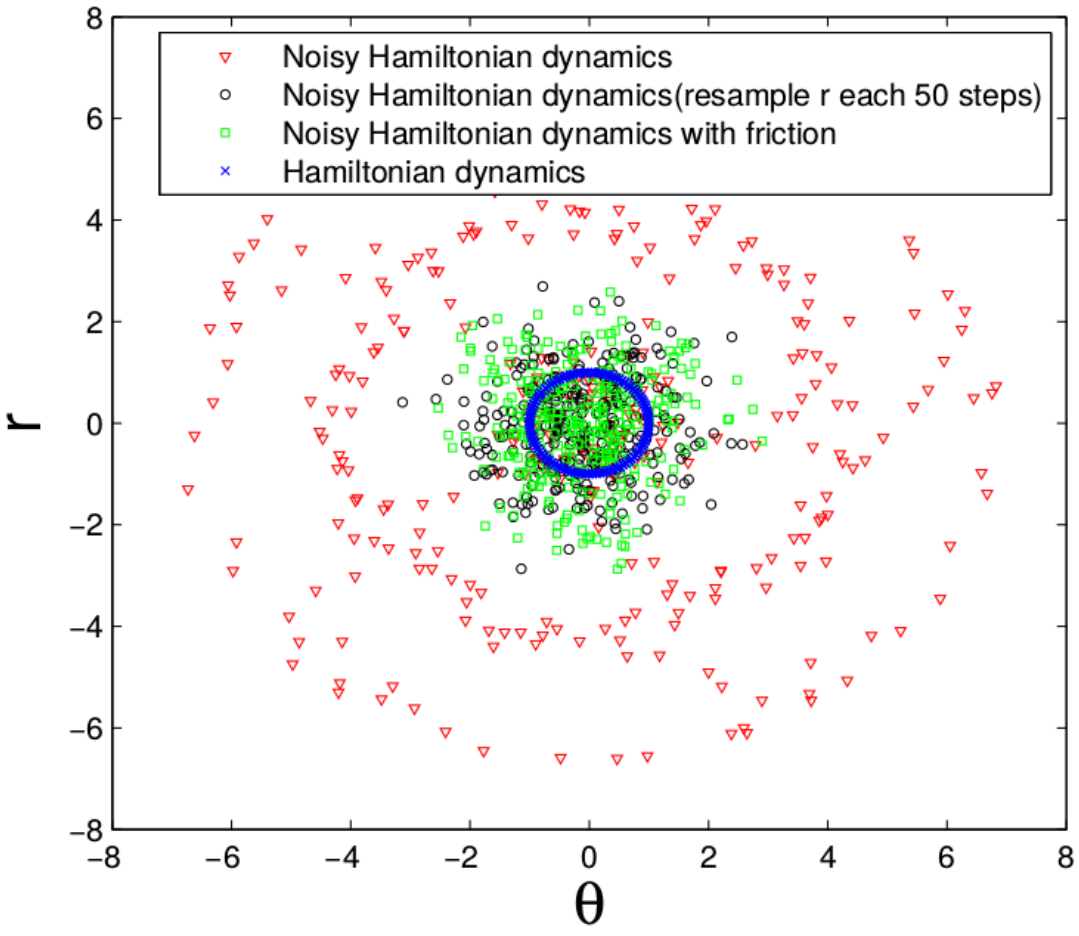


Figure. T. Chen et al. Stochastic Gradient Hamiltonian Monte Carlo, ICML, 2014.

SGHMC for deep learning

A Bayesian multilayer perceptron with 100 hidden neurons evaluated on MNIST.

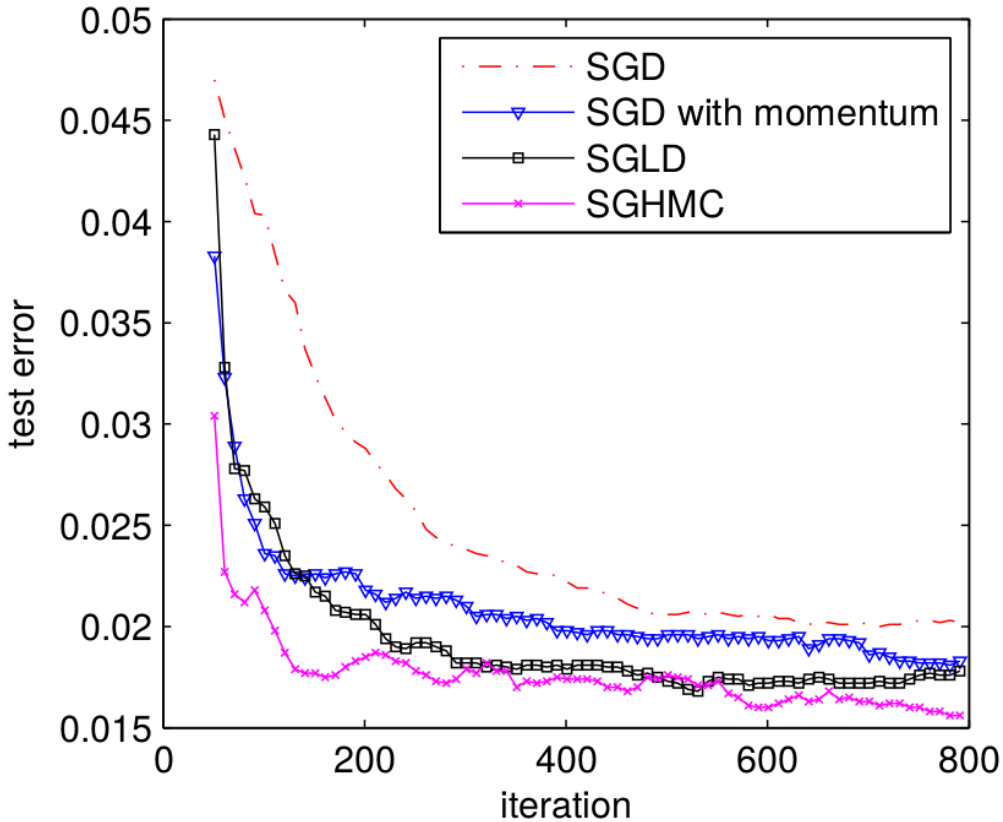


Figure. T. Chen et al. Stochastic Gradient Hamiltonian Monte Carlo, ICML, 2014.

Maximum A-Posteriori Estimation

For a Bayesian model $p(\theta)\ p(\mathcal{D}|\theta)$ with $p(\mathcal{D}|\theta) = \prod_{x\in\mathcal{D}} p(x|\theta)$, the mode of the posterior is called the *Maximum A-Posteriori Estimate (MAP)*. The MAP of a model can be found by

$$\operatorname*{argmin}_{\theta} \Big\{ -\log p(\theta) - \sum_{x\in\mathcal{D}} \log p(x|\theta) \Big\}.$$

When closed-form solution is not available, we do gradient descent

$$\theta_{t+1} \leftarrow \theta_t - \epsilon_t \Big(-\nabla \log p(\theta_t) - \sum_{x\in\mathcal{D}} \nabla \log p(x|\theta) \Big).$$

The second term demands a full pass on the training set, which is not feasible in many cases.

Stochastic Gradient Descent (SGD)

Robbins-Monro Theorem [Robbins & Monro 1951] says that the approximate gradient

$$\frac{|\tilde{\mathcal{D}}|}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \nabla \log p(x|\theta)$$

obtained by randomly sampling a minibatch $\tilde{\mathcal{D}}$, hence assuming that the training data consists of $\frac{|\tilde{\mathcal{D}}|}{\mathcal{D}}$ replications of $\tilde{\mathcal{D}}$ will converge to the exact gradient after infinitely many iterations if the learning rate follows a series satisfying

$$\sum_{t=1}^{\infty} \epsilon_t = \infty, \qquad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty.$$

Hence, the learning rate should never drop down to zero (left), but should still keep decreasing over time (right).

(First-order) Langevin Dynamics

$$\Delta\theta_t = \frac{\epsilon}{2}\left(\nabla \log p(\theta_t) + \sum_{x\in\mathcal{D}} \nabla \log p(x|\theta_t)\right) + \eta_t,$$

$$\eta_t = \mathcal{N}(0, \epsilon).$$

- Just as in HMC, introduced to solve stochastic differential equations.
- Converges to the posterior, but discretization error should be resolved by Metropolis correction.
- Unlike HMC, Gaussian noise injected to the gradient to enhance stochasticity.
- The noise variance ϵ is proportional to the learning rate.
- Decreasing ϵ also decreases the discretization error. Will be useful soon!

Stochastic Gradient Langevin Dynamics (SGLD)[3]

Adapt Langevin dynamics to SGD

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{|\tilde{\mathcal{D}}|}{|\mathcal{D}|}\sum_{x\in\tilde{\mathcal{D}}} \nabla \log p(x|\theta_t)\right) + \eta_t,$$

$$\eta_t = \mathcal{N}(0, \epsilon_t).$$

- To be eligible for Robbins-Monro, learning rate η_t has to decrease in time.
- The discretization error will also decrease $\Rightarrow$ Metropolis correction will no longer be required!

[3] M. Welling, Y.W. Teh, Stochastic Gradient Langevin Dynamics, ICML, 2011

SGD $\Rightarrow$ SGLD

- The algorithm has two phases, starts with first, switches to the second after some iterations:
  1. **SGD:** Performs only SGD with extended stochasticity (η_t) to overpass local minima.
  2. **Langevin Dynamics:** Samples from the true posterior.
- The key question is when this switching takes place.
- The system transitions into mode two when

$$\epsilon_t \approx \frac{4\alpha|\tilde{\mathcal{D}}|}{|\mathcal{D}|}\lambda_{min}(I_F^{-1}),$$

where I_F^{-1} is the empirical Fisher information of the stochastic gradient error, $\lambda_{min}(\cdot)$ is the smallest eigenvalue of the argument, and α is a sample threshold.

Calculating posterior expectations

- During training, we collect a set of samples $\{\theta_1, \theta_2, \cdots, \theta_T\}$.
- We will use them to approximate an expectation $E[f(\theta)]$, for instance the posterior predictive of a future observation x^*

$$p(x^*|\mathcal{D}) = \int p(x^*|\theta)p(\theta|\mathcal{D})d\theta = E[p(x^*|\theta)].$$

- Simple sample averaging

$$E[p(x^*|\theta)] \approx \frac{1}{T}\sum_{t=1}^{T} p(x^*|\theta_t)$$

will over-emphasize non-minimal regions where ϵ_t (hence discretization error) was high, the system was not yet sufficiently in the Langevin dynamics phase.

- Instead, do

$$E[p(x^*|\theta)] \approx \frac{\sum_{t=1}^{T} \epsilon_t p(x^*|\theta_t)}{\sum_{t=1}^{T} \epsilon_t}.$$

Bayesian neural net benchmarking on some UCI data sets

A Bayesian neural net with a single hidden layer of 50 units used. Root Mean Square Error on the test split reported.

	HMC	**Dropout**	**PBP**	**Varout**	**SGLD**
boston	2.76	2.97	3.01	2.70	**2.21**
concrete	**4.12**	5.23	5.67	4.89	4.19
kin8nm	0.06	0.10	0.10	0.08	**0.02**
power	3.73	4.02	4.12	4.04	**2.42**
protein	3.91	4.36	4.73	4.13	**1.07**
red wine	0.63	0.62	0.64	0.63	**0.21**
yacht	**0.56**	1.11	1.02	0.71	1.32