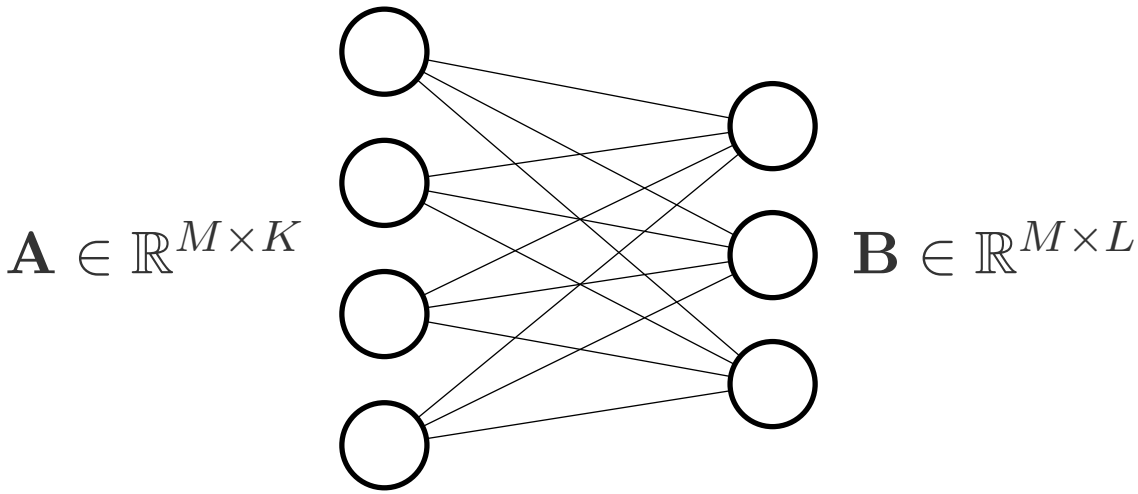
Variational Dropout for Deep Neural Networks and Linear Models

Dmitry Molchanov[1] **Arseniy Ashuha**[2]

[1]Skoltech
[2]MIPT

September 02, 2016

Binary Dropout

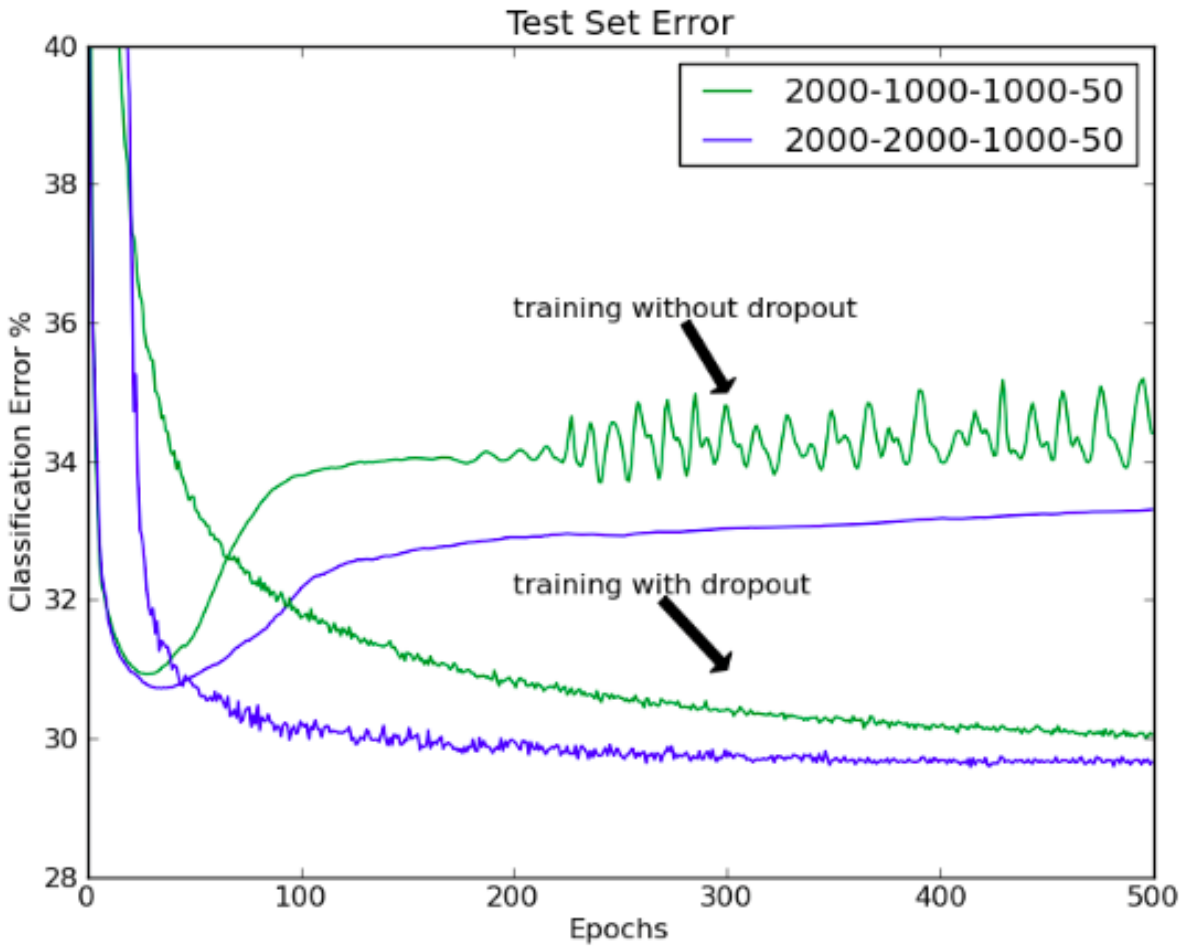


$$\mathbf{B}^t = (\mathbf{A}^t \circ \mathbf{\Xi})\mathbf{W}$$

$$\mathbf{\Xi} \in \{0, 1\}^{M \times K}$$

$$\xi_{mk} \sim \text{Bernoulli}(1 - p)$$

Binary Dropout



+ Prevents overfitting
− Slow and intractable

Gaussian Dropout

- During dropout testing we need to scale the weights:

$$\mathbf{W}_{test} = 1/(1-p)\mathbf{W}_{train}$$

- It is the same as scaling the noise during training:

$$\mathbf{B}^t = (\mathbf{A}^t \circ \mathbf{\Xi})\mathbf{W} \quad \rightarrow \quad \mathbf{B}^t = (\mathbf{A}^t \circ \mathbf{\Xi}/(1-p))\mathbf{W}$$

$$\mathbb{E}[\xi_{ij}/(1-p)] = 1, \qquad \mathbb{D}[\xi_{ij}/(1-p)] = p/(1-p)$$

- We can use a Gaussian distribution with the same mean and variance: $\xi_{ij} \sim \mathcal{N}(1, \alpha)$

$$\alpha \leftrightarrow \frac{p}{1-p}$$

\+ More tracktable

\– Still slow

Post-linear Gaussian Dropout

Gaussian dropout: $\mathbf{B}^t = (\mathbf{A}^t \circ \mathbf{\Xi})\mathbf{W}$

$$\xi_{ij} \sim \mathcal{N}(1, \alpha)$$

Activations $\mathbf{B}$ are also Gaussian:

$$p(b_{mj} \mid \mathbf{A}, \mathbf{W}, \alpha) = \mathcal{N}(\gamma_{mj}, \delta_{mj})$$

$$\gamma_{mj} = \sum_i a_{mi} w_{ij}, \quad \delta_{mj} = \alpha \sum_i a_{mi}^2 w_{ij}^2$$

+ We can sample activations $\Rightarrow$ faster
− Gradient variance is smaller, which can lead to overfitting

Variational Dropout intuition

$$p(b_{mj} \mid \mathbf{A}, \mathbf{W}, \alpha) = \mathcal{N}(\gamma_{mj}, \delta_{mj})$$

$$\gamma_{mj} = \sum_i a_{mi} w_{ij}, \quad \delta_{mj} = \alpha \sum_i a_{mi}^2 w_{ij}^2$$

It is the same as if the weights had the following Gaussian distribution:

$$q(w_{ij}) = \mathcal{N}(\theta_{ij}, \alpha\theta_{ij}^2)$$

$$b_{ij} = \sum_k a_{ik}(1 + \sqrt{\alpha} \cdot \epsilon)\theta_{kj}, \quad \epsilon \sim \mathcal{N}(0, 1)$$

Let's add a prior and search for posterior distribution approximation in this form!

Variational Lower Bound

$$q(w_{ij}) = \mathcal{N}(\theta_{ij}, \alpha\theta_{ij}^2)$$

$$KL(q(\boldsymbol{w})\|p(\boldsymbol{w}\,|\,\mathbf{X}, \boldsymbol{t})) \to \min_{q} \Leftrightarrow ELBO(q) \to \max_{\alpha,\theta}$$

$$ELBO = \mathbb{E}_q \log p(\boldsymbol{t}\,|\,\mathbf{X}, \boldsymbol{w}) - KL(q(\boldsymbol{w})\|p_{prior}(\boldsymbol{w}))$$

- Both α and θ are variational parameters
- We can now fit α
- We can use the same (shared) α for all layers, or a unique α_i for each layer/neuron/weight
- Gradients can be computed using reparametrization trick:

$$\mathbb{E}_q \log p(\boldsymbol{t}\,|\,\mathbf{X}, \boldsymbol{w}) = \mathbb{E}_{\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})} \log p(\boldsymbol{t}\,|\,\mathbf{X}, (1 + \sqrt{\alpha}\cdot\boldsymbol{\epsilon}) \circ \boldsymbol{\theta})$$

Prior

For ELBO optimization to be consistent with dropout training, the KL should not depend on θ:

$$ELBO \to \max_{\theta} \quad \Leftrightarrow \quad \mathbb{E}_q \log p(\boldsymbol{t} \,|\, \mathbf{X}, \boldsymbol{w}) \to \max_{\theta}$$

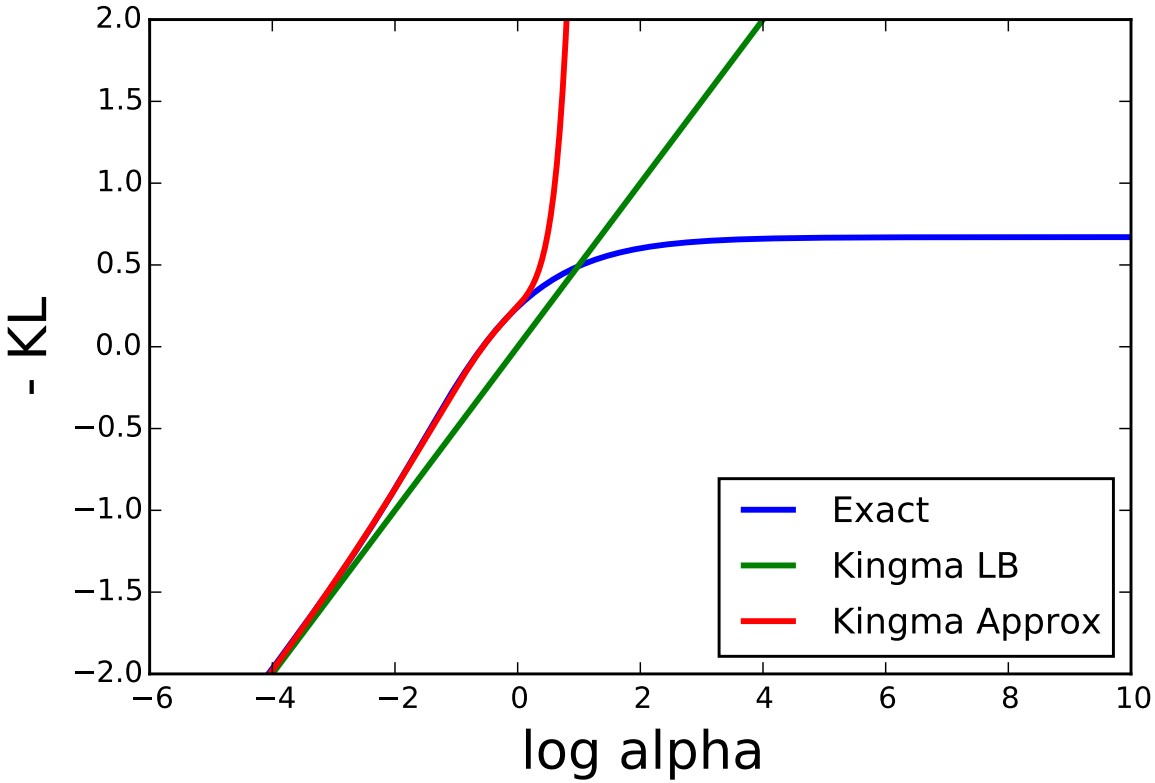
The only prior with such property is a log-scale uniform prior:

$$p(\log |w|) \propto const$$

$$KL(q(w)\|p_{prior}(w)) = const + \frac{1}{2}\log \alpha - \mathbb{E}_{\epsilon \sim \mathcal{N}(1,\alpha)} \log |\epsilon|$$

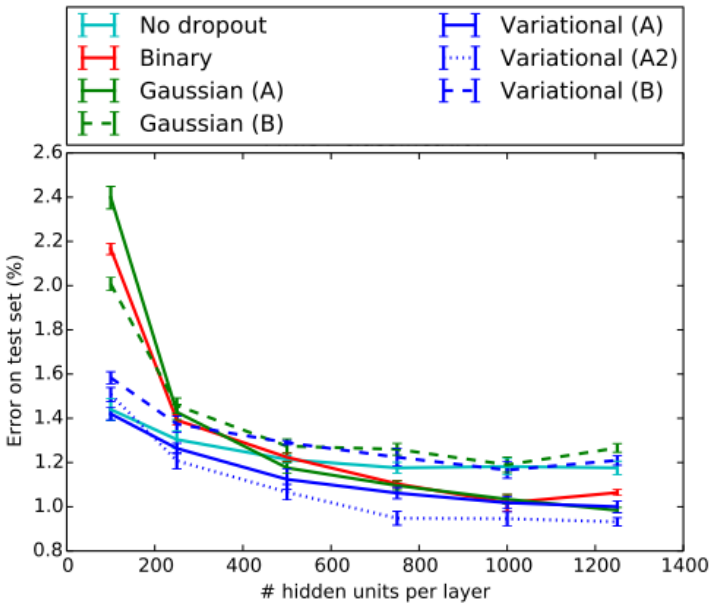
It can't be computed analytically, but can be approximated accurately.

Prior

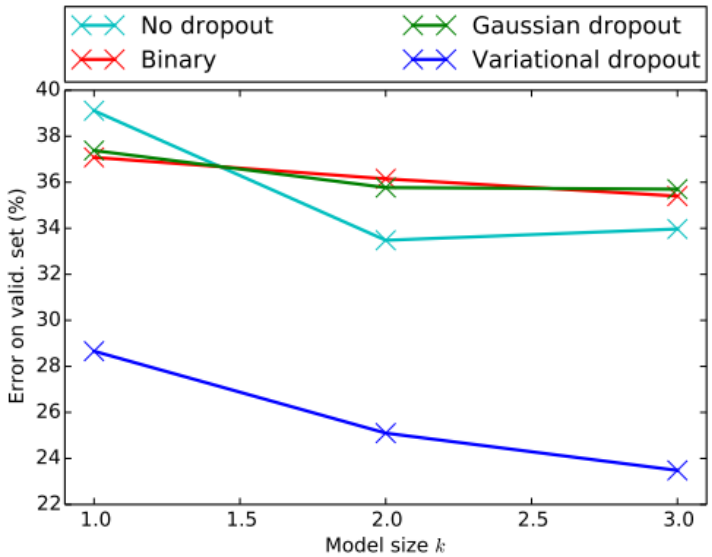


- It controls weight precision
- It favours large alphas
- It is scale-invariant

Kingma's experiments



(a) Classification error on the MNIST dataset



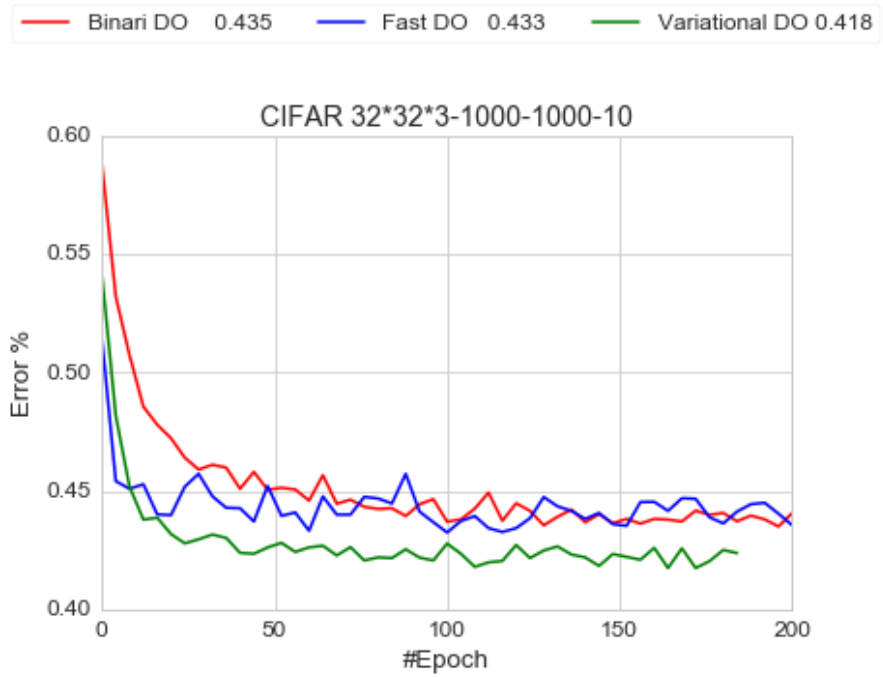
(b) Classification error on the CIFAR-10 dataset

The experimental part of the paper was quite strange:

- α are clipped at 1: $\alpha < 1,\ p < 0.5$
- Training method for alphas and alpha sharing scheme are not specified
- The KL divergence was divided by 3 to improve perfomance

Architectures and Result: Layer case

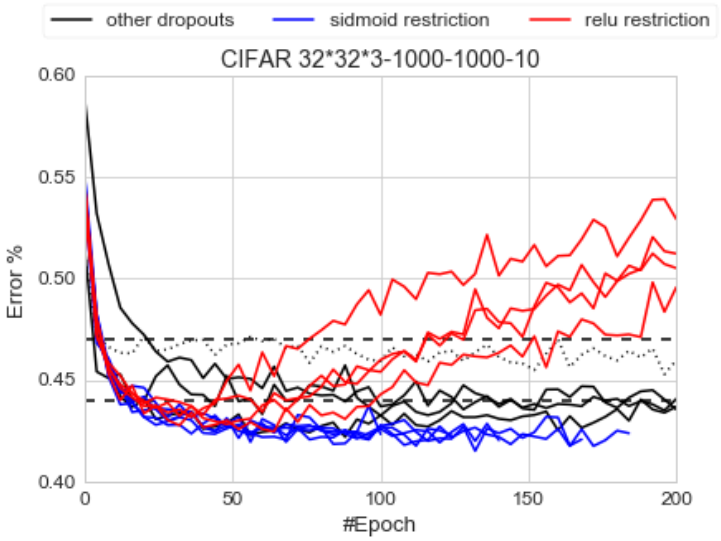
- **Datasets**: CIFAR **Arch**: 3FC(1000)



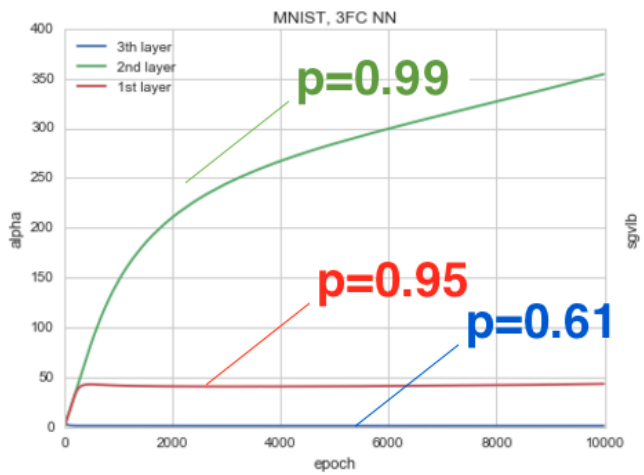
- **Problem**: Reguralizer is very large $\rightarrow \alpha = 1.0$ after few iteration

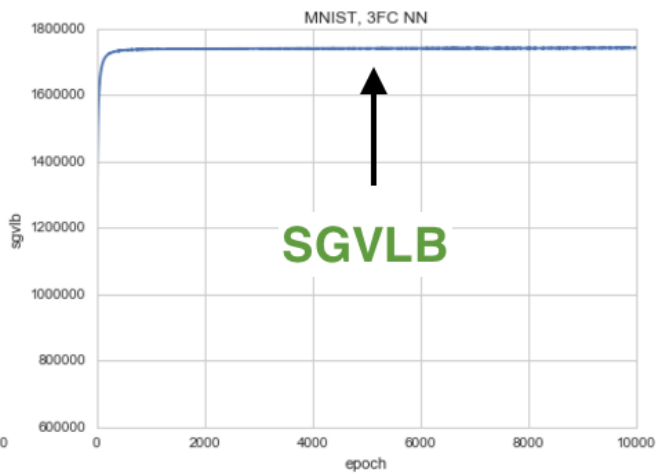
Can we remove the boundaries?

- Yes we can, but the quality becomes much worse



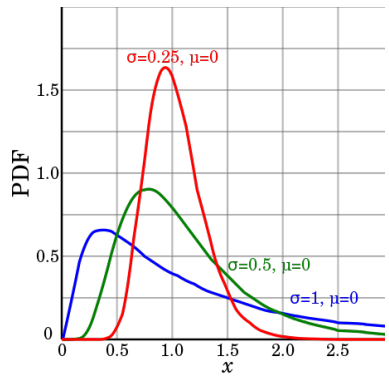
- Do alphas stabilize?





How can we deal with too powerful regularizer?

- Modern Neural Nets contain really a lot of parameters
- We will try to change our mind about regularizer parameters
- Let's control precision of neuron instead of precision of parameters
- We will try to describe our data by inexact neurons, and use exact ones only if it's necessary
- Activation usually is nonnegative, let's try Lognormal distribution



- Non-Symmetricity of distribution causes correct multiplicative noise, it will depend on absolute value
- The results will be announced later

Relevance Vector Machine

$\boldsymbol{x} \in \mathbb{R}^D$ is an object, t is its target.

Relevance Vector Regression:

$$t \in \mathbb{R}$$

$$p(t \,|\, \boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(t \,|\, \boldsymbol{w}^T \boldsymbol{x}, \beta^{-1})$$

$$p(\boldsymbol{w} \,|\, \boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w} \,|\, \mathbf{0}, \text{diag}(\alpha_1^{-1}, \alpha_2^{-1}, \ldots, \alpha_d^{-1})) = \mathcal{N}(\boldsymbol{w} \,|\, \mathbf{0}, \mathbf{A}^{-1})$$

$$\mathbf{A} = \text{diag}(\alpha_1, \alpha_2, \ldots, \alpha_d)$$

Relevance Vector Classification:

$$t \in \{-1, +1\}$$

$$p(t \,|\, \boldsymbol{x}, \boldsymbol{w}) = \sigma(t \boldsymbol{w}^T \boldsymbol{x})$$

$$p(\boldsymbol{w} \,|\, \boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w} \,|\, \mathbf{0}, \mathbf{A}^{-1})$$

Evidence maximization

$$E(\boldsymbol{\alpha}) = \int \left(\prod_{i=1}^{N} p(t_i \,|\, \boldsymbol{x}_i, \boldsymbol{w}) \right) p(\boldsymbol{w} \,|\, \boldsymbol{\alpha}) d\boldsymbol{w} \to \max_{\boldsymbol{\alpha}}$$

Automatic Relevance Determination:
$E(\alpha) \to \max \quad \Rightarrow \quad \alpha_j \to +\infty \quad \Rightarrow \quad w_j \to 0$ if feature j is irrelevant.

Variational Dropout: linear regression

Likelihood:

$$p(\boldsymbol{t} \,|\, \mathbf{X}, \tilde{\boldsymbol{w}}) = \mathcal{N}(\boldsymbol{t} \,|\, \mathbf{X}\tilde{\boldsymbol{w}}, \beta^{-1}\mathbf{I})$$

Objective:

$$\mathcal{L}(q) = \frac{N}{2}\log\frac{\beta}{2\pi} - \frac{\beta}{2}\Big[\|\mathbf{X}\boldsymbol{w} - \boldsymbol{t}\|^2 + \mathrm{Tr}(\mathbf{X}^T\mathbf{X}\mathrm{diag}(\boldsymbol{\alpha} \circ \boldsymbol{w}^2))\Big] - \mathcal{D}(\boldsymbol{\alpha})$$

Exact update for $\boldsymbol{w}$:

$$\boldsymbol{w}^* = (\mathbf{X}^T\mathbf{X} - \mathrm{diag}(\boldsymbol{\varkappa}))^{-1}\mathbf{X}^T\boldsymbol{t}$$

$$\varkappa_d = \alpha_d \sum_{n=1}^{N} x_{nd}^2$$

- Looks much like corresponding expression from RVR
- $\alpha_d \to +\infty \Rightarrow \theta_d \to 0$

Variational Dropout: logistic regresion

Likelihood:

$$p(\boldsymbol{t} \,|\, \mathbf{X}, \tilde{\boldsymbol{w}}) = \prod_{n=1}^{N} \sigma(t_n \tilde{\boldsymbol{w}}^T \boldsymbol{x}_n)$$

To calculate the objective, we need to compute

$$\int \log \sigma(t_n \tilde{\boldsymbol{w}}^T \boldsymbol{x}_n) \mathcal{N}(\tilde{\boldsymbol{w}} \,|\, \boldsymbol{w}, \operatorname{diag}(\boldsymbol{\alpha} \circ \boldsymbol{w}^2)) d\tilde{\boldsymbol{w}} =$$

$$= \int \log \sigma(s_n y + m_n) \mathcal{N}(y \,|\, 0, 1) dy$$

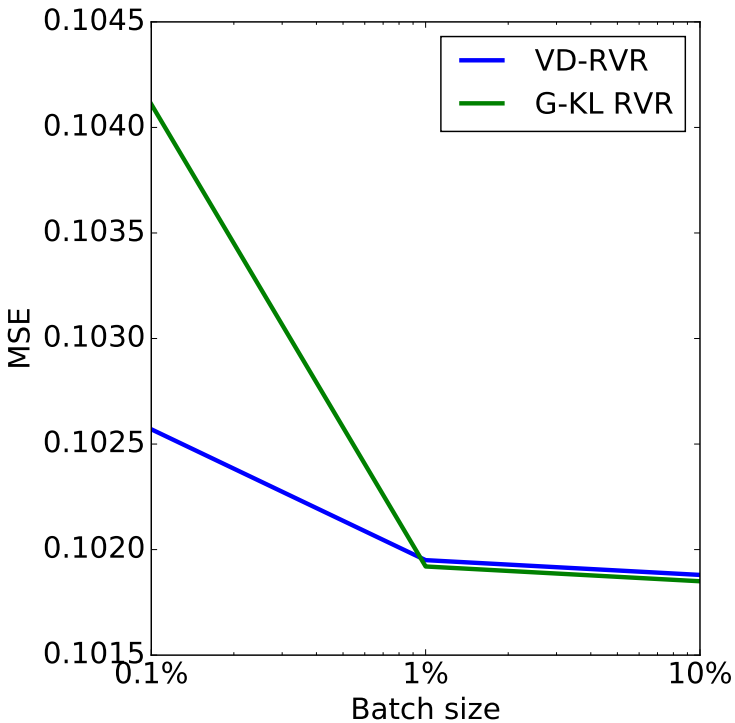
$$\boldsymbol{m} = \boldsymbol{t} \circ \mathbf{X}\boldsymbol{w}, \quad \boldsymbol{s}^2 = \mathbf{X}^2(\boldsymbol{\alpha} \circ \boldsymbol{w}^2)$$

- It can be computed with Gauss–Hermite quadratures
- Or we can use sampling lie in NNs

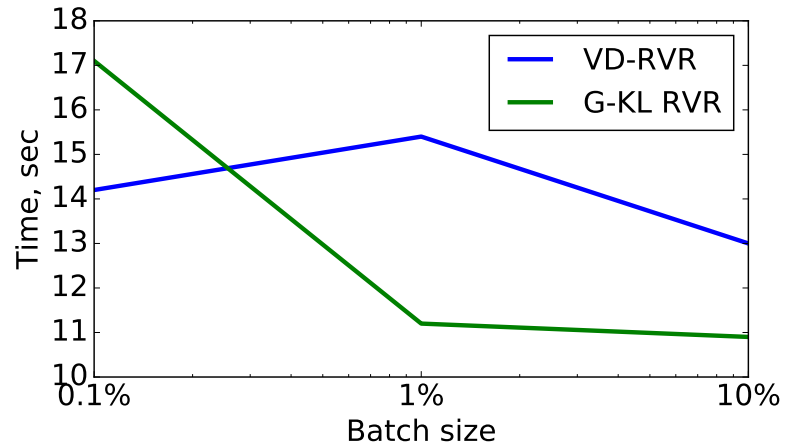
Experiments: linear regression

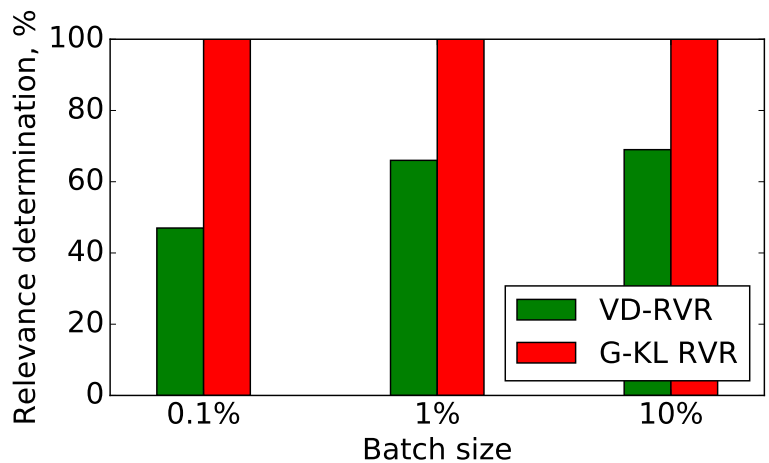
Synthetic data, 100000 objects, 100 features.

Test set mean squared error



Time and relevance determination quality

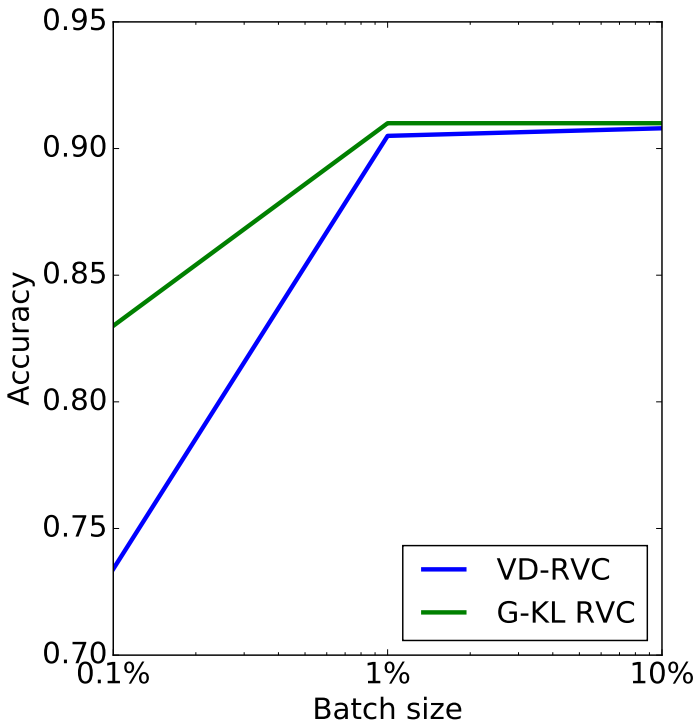




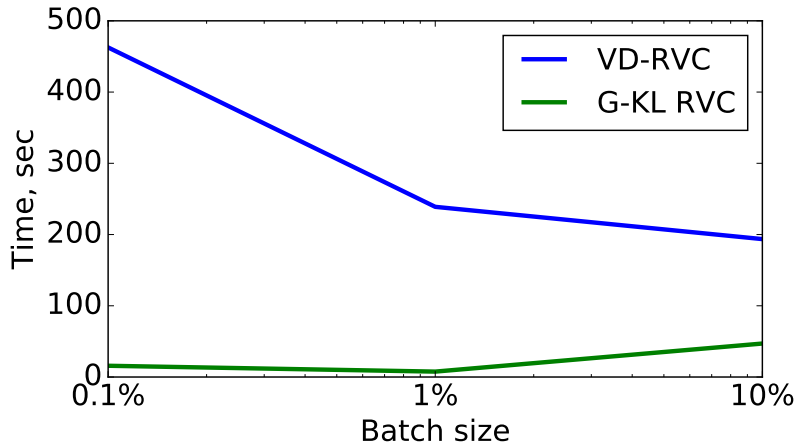
Experiments: logistic regression

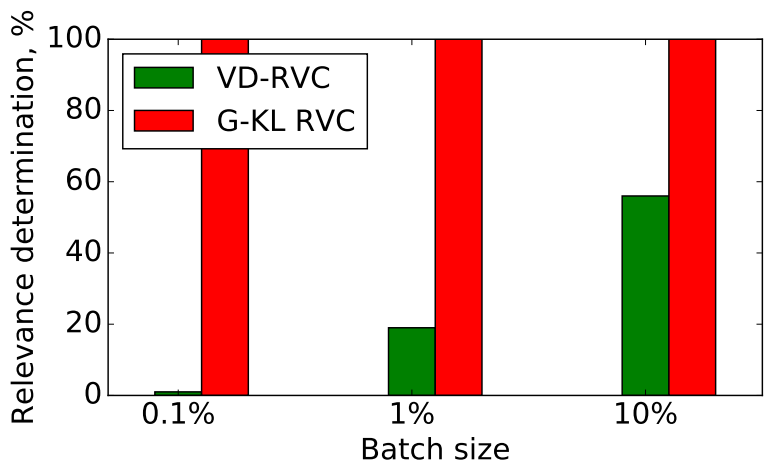
Synthetic data, 100000 objects, 100 features.

Test set accuracy



Time and relevance determination quality





Experiments: more features

1000 objects, 10 relevant features, 990 irrelevant features.

Linear regression:

Method	Test MSE	Relevance determination
VD-RVR	0.119	816/990
G-KL RVR	0.128	990/990

Logistic regression:

Method	Test accuracy	Relevance determination
VD-RVC	0.916	984/990
G-KL RVC	0.907	983/990