Fast "dropout" training for logistic regression

Sida Wang
Department of Computer Science
Stanford University
Stanford, CA 94305
sidaw@cs.stanford.edu

Christopher D. Manning
Department of Computer Science
Stanford University
Stanford, CA 94305
manning@cs.stanford.edu

Abstract

Recently, improved classification performance has been achieved by encouraging independent contributions from input features, or equivalently, by preventing feature co-adaptation. In particular, the method proposed in [1], informally called "dropout", did this by randomly dropping out (zeroing) hidden units and input features for training neural networks. However, sampling a random subset of input features during training makes training much slower. We look at the implied objective function of the dropout training method in the context of logistic regression. Then, instead of doing a Monte Carlo optimization of this objective as in [1], we show how to optimize it more directly by using a Gaussian approximation justified by the central limit theorem and empirical evidence, resulting in a 2-30 times speedup and more stability when it is applicable. We outline potential ways of extending the Gaussian approximation to neural networks and draw some connections to other methods in the literature. Finally, we empirically compare the performance of this method to previously published results, and to baselines. Code to replicate results in this paper will be made available.

1 Introduction

Recent work [1] has shown that encouraging independent contributions from each input feature, or equivalently, preventing feature co-adaptation is a promising method for regularization. This can be considered an alternative approach to regularization in addition to the widely used parameter shrinkage methods and Bayesian averaging. In [1], neural networks are trained while randomly dropping out (zeroing) hidden units and input dimensions. This process discourages the use of a feature that is only helpful while other specific features are present.

Similar observations existed in the literature. Naive Bayes, by completely ignoring co-adaptation, performs better than discriminative methods when there is little data [2], and continues to perform better on certain relatively large datasets [3]. Furthermore, there are several works of a similar spirit. 1) In [4], it is observed that training involves trade-offs among weights, where the presence of highly indicative features can cause other useful but weaker features to undertrain. They proposed feature bagging: training different models on subset of features that are later combined. 2) In [3], each input feature has an L2 regularization strength that depends on how informative the particular feature is by itself, so individually uninformative features are strongly regularized, preventing co-adaptation.

While the effectiveness of these methods is demonstrated, actually sampling, or training multiple models, make training much slower. For example, with a dropout rate of $p = 0.5$, the proportion of data still not seen after n passes is p^n (i.e. 5 passes of the data is required to see 95% of it). If the data is not highly redundant, and if the relevant data is only partially observable at random, then the task also becomes harder, and the training efficiency may reduce further. In this paper, we look at how to achieve the benefit of "dropout" training without actually sampling, thereby using all the data efficiently. We do this by a Gaussian approximation that is justified by the central limit theorem

and empirical evidence. We focus on logistic regression for simplicity, but the idea is also applicable to other probabilistic discriminative models, and to neural networks.

2 The implied objective function, and its faster approximations

To avoid unnecessarily cumbersome notations, we illustrate the main idea with binary logistic regression (LR) with a single training vector x, and label $y \in \{0, 1\}$.

2.1 The implied objective function for dropout training

To train LR with dropout on data with dimension m, first sample $z_i \sim \text{Bernoulli}(p_i)$ for $i = 1...m$. Here p_i is the probability of not dropping out input x_i. For neural networks, x is activation of the previous layer. For LR, x is the data. Typically, $p_{\texttt{bias}} = 1$ and $p = 0.1 \sim 0.9$ for everything else. After sampling $z = \{z_i\}_{i=1...m}$ we can compute the stochastic gradient descent (sgd) update as follows:

$$\Delta w = (y - \sigma(w^t D_z x))D_z x \tag{1}$$

where $D_z = \text{diag}(z) \in \mathcal{R}^{m \times m}$, and $\sigma(x) = 1/(1 + e^{-x})$ is the logistic function.

This update rule, applied over the training data over multiple passes, can be seen as a Monte Carlo approximation to the following gradient:

$$\Delta \bar{w} = E_{z;z_i \sim \text{Bernoulli}(p_i)}[(y - \sigma(w^t D_z x))D_z x] \tag{2}$$

The objective function with the above gradient is the expected conditional log-likelihood of the label given the data with dropped out dimensions indicated by z, for $y \sim \text{Bernoulli}(\sigma(w^t D_z x)))$. This is the implied objective function for dropout training:

$$L(w) = E_z[\log(p(y|D_z x; w)] = E_z[y \log(\sigma(w^t D_z x)) + (1 - y) \log(\sigma(w^t D_z x))] \tag{3}$$

Since we are just taking an expectation, the objective is still concave provided that the original log-likelihood is concave, which it is for logistic regression.

Evaluating the expectation in (2) naively by summing over all possible z has complexity $O(2^m m)$. Rather than directly computing the expectation with respect to z, we propose a variable transformation that allows us to compute the expectation with respect to a simple random variable $Y \in \mathbb{R}$, instead of $z \in \mathbb{R}^m$.

2.2 Faster approximations to the dropout objective

We make the observation that evaluating the objective function $L(w)$ involves taking the expectation with respect to the variable $Y(z) = w^t D_z x = \sum_i^m w_i x_i z_i$, a weighted sum of Bernoulli random variables. For most machine learning problems, $\{w_i\}$ typically form a unimodal distribution centered at 0, $\{x_i\}$ is either unimodal or in a fixed interval. In this case, Y can be well approximated by a normal distribution even for relatively low dimensional data with $m = 10$. More technically, the Lyapunov condition is generally satisfied for a weighted sum of Bernoulli random variables of the form Y that are weighted by real data [5]. Then, Lyapunov's central limit theorem states that $Y(z)$ tends to a normal distribution as $m \to \infty$. We empirically verify that the approximation is good for typical datasets of moderate dimensions, except when a couple of dimensions dominate all others. Finally, let

$$S = E_z[Y(z)] + \sqrt{\text{Var}[Y(z)]}\epsilon = \mu_S + \sigma_S \epsilon \tag{4}$$

be the approximating Gaussian, where $\epsilon \sim \mathcal{N}(0, 1)$, $E_z[Y(z)] = \sum_i^m p_i w_i x_i$, and $\text{Var}\,[Y(z)] = \sum_i^m p_i(1 - p_i)(w_i x_i)^2$.

2.2.1 Gaussian approximation

Given good convergence, we note that drawing samples of the approximating Gaussian S of $Y(z)$, a constant time operation, is much cheaper than drawing samples directly of $Y(z)$, which takes $O(m)$. This effect is very significant for high dimensional datasets. So without doing much, we can already approximate the objective function (3) m times faster by sampling from S instead of

$Y(z)$. Empirically, this approximation is within the variance of the direct MC approximation of (3) by taking 200 samples of z.

Approximating the gradient introduces a complication while using samples from the Gaussian. The gradient (2) involves not only $Y(z) \to S$, but also $D_z x$ directly:

$$\nabla L(w) = E_z[(y - \sigma(Y(z)))D_z x] \tag{5}$$

Let $f(Y) = f(Y(z)) = y - \sigma(Y(z))$ and let $g(z) = D_z x \in \mathbb{R}^m$. Naively approximating $E_z[f(Y(z))g(z)]$ by either $E_S[f(S)]E_z[g(z)]$, or worse, by $f(E_s[S])E_z[g(z)]$ works poorly in terms of both approximation error and final performance. Note $g(z)$ is a linear function and therefore $E_z[g(z)] = g(E_z[z]) = \text{diag}(p)x$. A good way to approximate (5) is by analytically taking the expectation with respect to z_i and then using a linear approximation to the conditional expectation. More precisely, consider dimension i of the gradient:

$$\begin{aligned}
\frac{\partial L(w)}{\partial w_i} &= E_z[f(Y(z))x_i z_i] \\
&= \sum_{z_i \in \{0,1\}} p(z_i) z_i x_i E_{z_{-i}|z_i}[f(Y(z))] \\
&= p(z_i = 1) x_i E_{z_{-i}|z_i=1}[f(Y(z))] \\
&\approx p_i x_i \left(E_{S \sim \mathcal{N}(\mu_S, \sigma_S^2)}[f(S)] + \Delta\mu_i \frac{\partial E_{T \sim \mathcal{N}(\mu, \sigma_S^2)}[f(T)]}{\partial \mu}\bigg|_{\mu=\mu_S} + \right. \\
&\left. \Delta\sigma_i^2 \frac{\partial E_{T \sim \mathcal{N}(\mu_S, \sigma^2)}[f(T)]}{\partial \sigma^2}\bigg|_{\sigma^2=\sigma_S^2} \right) \\
&= p_i x_i (\alpha(\mu_S, \sigma_S^2) + \Delta\mu_i \beta(\mu_S, \sigma_S^2) + \Delta\sigma_i^2 \gamma(\mu_S, \sigma_S^2))
\end{aligned} \tag{6}$$

where z_{-i} is the collection of all other zs except z_i, μ_S, σ_S is defined in (4), $\Delta\mu_i = (1 - p_i)x_i w_i$, $\Delta\sigma_i^2 = -p_i(1 - p_i)x_i^2 w_i^2$ are the changes in μ_S, σ_S^2 due to conditioning on z_i. Note that the partial derivatives as well as $E_{S \sim \mathcal{N}(\mu_S, \sigma_S^2)}[f(S)]$ only need to be computed once per training case, since they are independent of i.

α, β, γ can be computed by drawing K samples from S and takes time $O(K)$. See A.2 for details. In practice, using only β approximates the derivative to within the variance of successive MC computations of the objective L (see figure 2). In our experiments, this is 2-30 times faster compared to sampling (see figure 1 and table 1). While MC dropout becomes inefficient quickly with very high dropout rate (> 0.8), the Gaussian approximation is fairly insensitive because it still gets all the information from every data point (see 1).

2.2.2 Deterministic approximations

We can further improve the performance by using fast deterministic approximations. With a potentially much faster deterministic approximation, randomness harmful to line-search is removed. While not faster, we can deterministically "sample" from S. There is no analog of doing this with the MC dropout method (it would be silly to permanently corrupt predetermined data dimensions). In addition to taking the expectation as mentioned in 2.2.1, α, β, γ are functions of $\mu_s, \sigma_s^2 \in \mathbb{R}$, and can be computed by 1-d numerical integration as defined in (7), (8), and (9). For even faster speed, one can also tabulate α, β, γ instead of computing them as needed. The function is smoother if parameterized by $\frac{\mu}{\sigma}, \sigma$ instead. However, numerical integration and a lookup table fail to scale to multinomial LR. In that case, we can use the Unscented Transform (UST) instead of sampling [6]. The rest of the procedure remains unchanged from 2.2.1.

3 Experiments

The accuracy and time taken are listed in table 1 for the datasets described in section A.1. The Gaussian approximation is generally around 10 times faster than MC dropout and performs comparably to NBSVM in [3]. Further speedup is possible by using one of the deterministic approximations instead of sampling. While each iteration of the Gaussian approximation is still slower than LR, it sometimes reaches a better validation performance in less time.

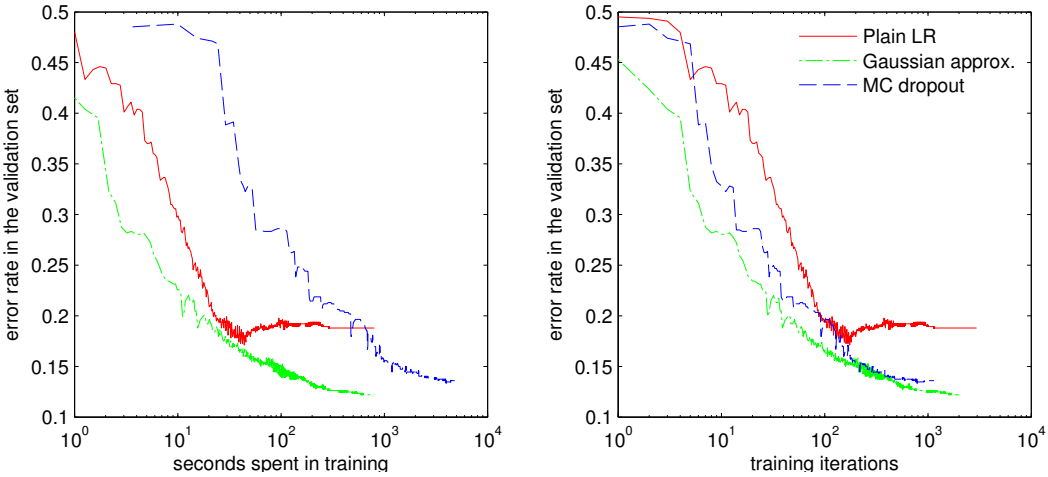

Figure 1: *Validation errors vs. time spent in training (left), and number of iterations (right)*. trained using batch gradient descent with Wolfe line search on the 20-newsgroup subtask alt.atheism vs. religion.misc. 100 samples are used. For MC dropout, z_i is sampled only for non-zero x_i, with a dropout rate of 0.5.

Methods\ Datasets	RT-2k	IMDB	RTs	subj	AthR	CR	MPQA	Average
MC dropout	89.8	91.2	79.2	93.3	86.7	82.0	86.0	86.88
training time	6363	6839	2264	2039	126	582	417	2661
Gaussian approx.	89.7	91.2	79.0	93.4	87.4	82.1	86.1	86.99
training time	240	1071	362	323	6	90	185	325
plain LR	88.2	89.5	77.2	91.3	83.6	80.4	84.6	84.97
training time	145	306	81	68	3	17	22	92
Previous results								
TreeCRF[7]	-	-	77.3	-	-	81.4	86.1	-
Vect. Sent.[8]	88.9	88.89	-	88.13	-	-	-	-
RNN[9]	-	-	77.7	-	-	-	86.4	-
NBSVM[3]	89.45	91.22	79.4	93.2	87.9	81.8	86.3	87.03
$\lvert\{i : x_i > 0\}\rvert$	788	232	22	25	346	21	4	

Table 1: *The main table of results.* The top section contains the accuracy, and training time (in seconds) for various datasets. 100 samples are used for both MC dropout and the Gaussian approximation. The last row shows the average number of non-sparse dimensions in the dataset.

4 Conclusions

Dropout training, as originally proposed, was intended for neural networks where hidden units are dropped out, instead of the data. Fast dropout is directly applicable to dropping out the final hidden layer of neural networks: the gradients for backpropagation to the previous layer, $\frac{dL}{dx}$, can be computed by the procedure described in 2.2.1 replacing x with w. This is a topic of ongoing research.

We presented a way of getting the benefits of dropout training for LR without actually sampling, thereby speeding up the process by a factor of 2-30 times. For high dimensional datasets (over a few hundred), each iteration of fast dropout is only about 2 times slower than plain LR. While the objective for fast dropout is the same as MC dropout in the long run, because fast dropout is not losing any information in individual training cases, it is capable of doing more work in each iteration, reaching the same validation set performance in a shorter time than LR, and generalizing better.

References

[1] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors". In: *CoRR* abs/1207.0580 (2012).

[2] Andrew Y. Ng and Michael I. Jordan. "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes". In: *Proceedings of NIPS*. Vol. 2. 2002, pp. 841–848.

[3] Sida Wang and Christopher Manning. "Baselines and Bigrams: Simple, Good Sentiment and Topic Classification". In: *Proceedings of the ACL*. 2012, pp. 90–94.

[4] Charles Sutton, Michael Sindelar, and Andrew McCallum. "Reducing Weight Undertraining in Structured Discriminative Learning". In: *Conference on Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*. 2006.

[5] Erich L. Lehmann. *Elements of Large-Sample Theory*. Springer, 1998, p. 101. ISBN: 03873985956.

[6] Simon J. Julier and Jeffrey K. Uhlmann. "A New Extension of the Kalman Filter to Nonlinear Systems". In: *Proceedings of AeroSense: Simulations and Controls*. 1997, pp. 182–193.

[7] Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. "Dependency tree-based sentiment classification using CRFs with hidden variables". In: *Proceedings of ACL:HLT*. 2010.

[8] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning Word Vectors for Sentiment Analysis". In: *Proceedings of the ACL*. 2011.

[9] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions". In: *Proceedings of EMNLP*. 2011.

[10] Bo Pang and Lillian Lee. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales". In: *Proceedings of the ACL*. 2005.

[11] Minqing Hu and Bing Liu. "Mining and summarizing customer reviews". In: *Proceedings ACM SIGKDD*. 2004, pp. 168–177.

[12] Janyce Wiebe, Theresa Wilson, and Claire Cardie. "Annotating Expressions of Opinions and Emotions in Language". In: *Language Resources and Evaluation* 39.2-3 (2005), pp. 165–210.

[13] Bo Pang and Lillian Lee. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". In: *Proceedings of the ACL*. 2004.

A Supplementary Material

A.1 Description of the datasets used

We compare with published results on the these datasets. Detailed statistics are shown in table 2.

RT-s: Short movie reviews dataset containing one sentence per review [10]. This is known as the movie review (MR) dataset in some works.

CR: Customer review dataset [11] processed like in [7].[1]

MPQA: Opinion polarity subtask of the MPQA dataset [12].[2]

Subj: The subjectivity dataset with subjective reviews and objective plot summaries [13].

IMDB-2k (RT-2k): The standard 2000 full-length movie review dataset [13]. We mistakenly refered to this as **RT-2k** in [3], but it actually comes from $IMDB$.

IMDB: A large movie review dataset with 50k full-length reviews [8].[3]

AthR, XGraph, BbCrypt: Classify pairs of newsgroups in the 20-newsgroups dataset with all headers stripped off (the third (18828) version[4]), namely: alt.atheism vs. religion.misc, comp.windows.x vs. comp.graphics, and rec.sport.baseball vs. sci.crypt, respectively.

Dataset	(N_+, N_-)	l	CV	$\lvert V\rvert$	Δ
RT-s	(5331,5331)	21	10	21K	0.8
CR	(2406,1366)	20	10	5713	1.3
MPQA	(3316,7308)	3	10	6299	0.8
Subj.	(5000,5000)	24	10	24K	0.8
RT-2k	(1000,1000)	787	10	51K	1.5
IMDB	(25k,25k)	231	N	392K	0.4
AthR	(799,628)	345	N	22K	2.9
XGraph	(980,973)	261	N	32K	1.8
BbCrypt	(992,995)	269	N	25K	0.5

Table 2: Dataset statistics. (N_+, N_-): number of positive and negative examples. l: average number of words per example. CV: number of cross-validation splits, or N for train/test split. $|V|$: the vocabulary size. Δ: upper-bounds of the differences required to be statistically significant at the $p < 0.05$ level.

A.2 Computing α, β, γ

α, β, γ can be computed by either numerical integration or by sampling:

$$\alpha(\mu, \sigma^2) = y - E_{T\sim\mathcal{N}(\mu,\sigma^2)}\left[\frac{1}{1+e^{-T}}\right] = y - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \frac{1}{1+e^{-x}} dx \tag{7}$$

$$\beta(\mu, \sigma^2) = \frac{\partial\alpha(\mu,\sigma^2)}{\partial\mu} = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \frac{x-\mu}{\sigma^2(1+e^{-x})} dx = -\sigma^{-2} E_{T\sim\mathcal{N}(\mu,\sigma^2)}\left[\frac{T-\mu}{1+e^{-T}}\right] \tag{8}$$

$$\gamma(\mu, \sigma^2) = \frac{\partial\alpha(\mu,\sigma^2)}{\partial\sigma^2} = -\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \left(\frac{(x-\mu)^2}{2\sigma^4(1+e^{-x})} - \frac{1}{2\sigma^2(1+e^{-x})}\right) dx \tag{9}$$

[1] http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html

[2] http://www.cs.pitt.edu/mpqa/

[3] http://ai.stanford.edu/~amaas/data/sentiment

[4] http://people.csail.mit.edu/jrennie/20Newsgroups

A.3 Regularization by individual informativeness

While the Gaussian and deterministic approximations are more efficient than the direct MC computation, it still does considerably more work than plain LR. We present another procedure that is in a very similar spirit that is as fast as plain LR, but possibly less robust than dropout. It is a reinterpretation of the feature transformation described in [3] applied to Support Vector Machines. The idea can be very easily applied to logistic regression. Let $L_0(w) = -\log(p(y|x, w))$ be the negative log-likelihood. Define the cost function:

$$L(w) = L_0(w) + \sum_{i,j} C_{ij} w_{ij}^2 \tag{10}$$

The only difference from regular logistic regression is that C_{ij} depends on how informative x_j is for label $y = i$, so that individually uninformative features are heavily regularized. For sparse binary features a good choice is $\frac{1}{C_{ij}} = |\log \frac{p(x_j|y=i)}{p(x_j|y\neq i)}| + \epsilon$. More generally, one can either bin the data to make it binary or fit a decision stump $\tau_{ij} \in \mathbb{R}, b_{ij} \in \{0, 1\}$ to the data. We can set $\frac{1}{C_{ij}} = \left| log(\frac{p(b_{ij}x_j>\tau_{ij}|y=i)}{p(b_{ij}x_j>\tau_{ij}|y\neq i)} \right| + \epsilon$.

A.4 Other Figures

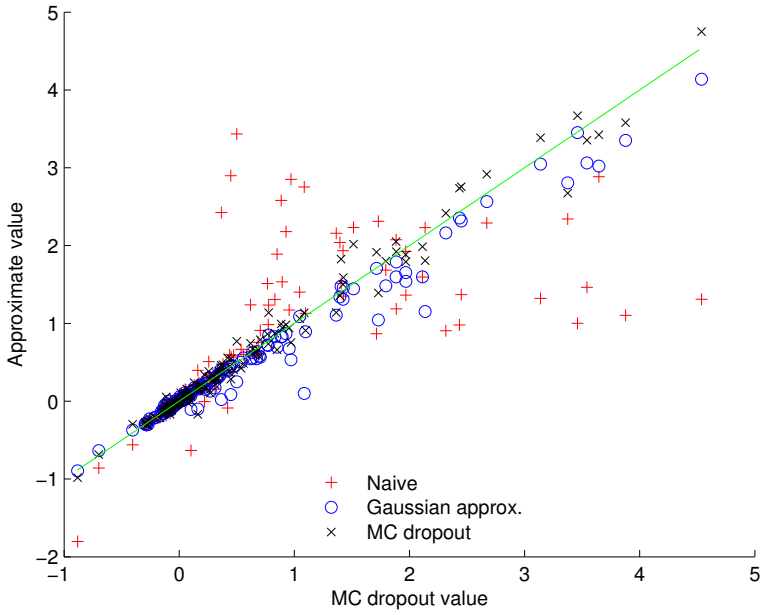

Figure 2: *Scatterplot of various approximations (y-axis) vs. direct MC dropout*: Each point is a random dimension of the gradient, with its x-value computed from MC dropout with 200 samples of z, and its y-value computed by the method in the legend. MC dropout and Gaussian approximation used 200 samples. Naive is the approximation defined after (5), by assuming that $f(z)$ and $g(z)$ are independent. The RMSE for different MC dropout runs is 0.0305, and the RMSE between MC dropout and Gaussian approximation is 0.0308, showing no difference between our approximation and MC dropout training. The green line is the reference $y = x$.

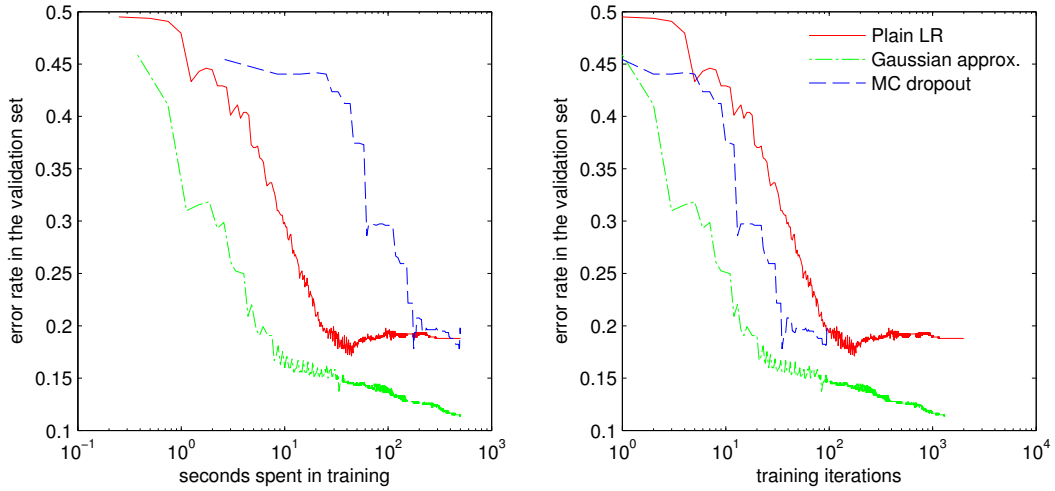

Figure 3: *Validation errors vs. time spent in training*: trained using batch gradient descent with Wolfe line search on the 20-newsgroup subtask alt.atheism vs. religion.misc. 100 samples are used. For MC dropout, z_i is sampled only for non-zero x_i. This is with a dropout rate of 0.7.

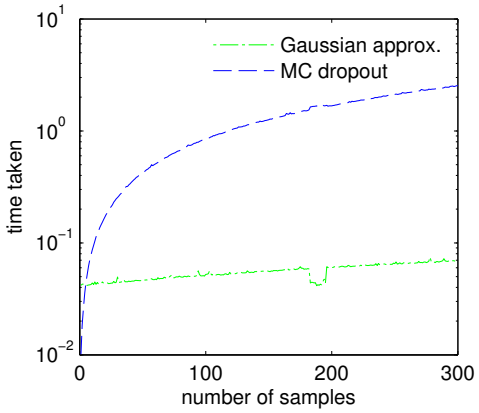

Figure 4: *Time required vs. number of samples*: while for MC dropout increases linearly with the number of samples, K, the time required to run the Gaussian approximation is only weakly dependent on K.