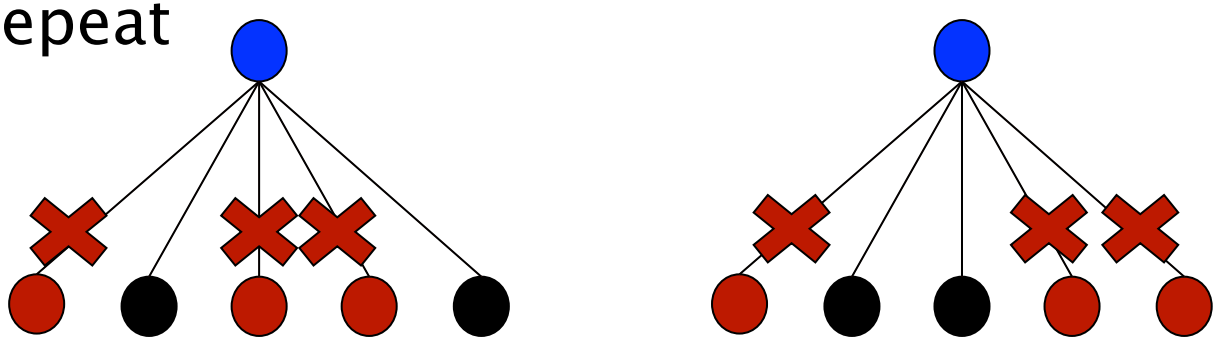
Fast deterministic dropout training

NIPS workshop
Sida Wang, Chris Manning

What is dropout training

- Recently introduced by Hinton et al. in "Improving neural networks by preventing co-adaptation of feature detectors"
- Randomly select some inputs for each unit
  - zero them
  - compute the gradient, make an update
  - repeat

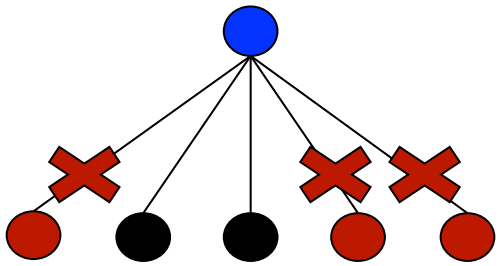


Dropout training is promising

- Won the ImageNet challenge by a margin
- George Dahl et al. won the Merck challenge
- Dropout seems to be an important ingredient

Preventing the danger of co-adaptation

- Observed in NLP and machine learning
- Model averaging
  - "Feature bagging": train models on different subset of features [Sutton et al, 2006] to prevent "weight undertraining".
- Deal with co-adaptation
  - Naïve Bayes can sometimes do better than discriminative methods
  - Regularize uninformative features more [Wang/ Manning, 2012]



In this work…

- Sampling is inefficient – we integrate
  - 50% random dropout, after 5 passes of the data, 1/32 = 3% of the data is still unseen
- No objective function was used in the original work – we use an implied objective function
- Understanding dropout as Bayesian model selection, and extensions

CLT and the Gaussian approximation

- Warm up with binary logistic regression

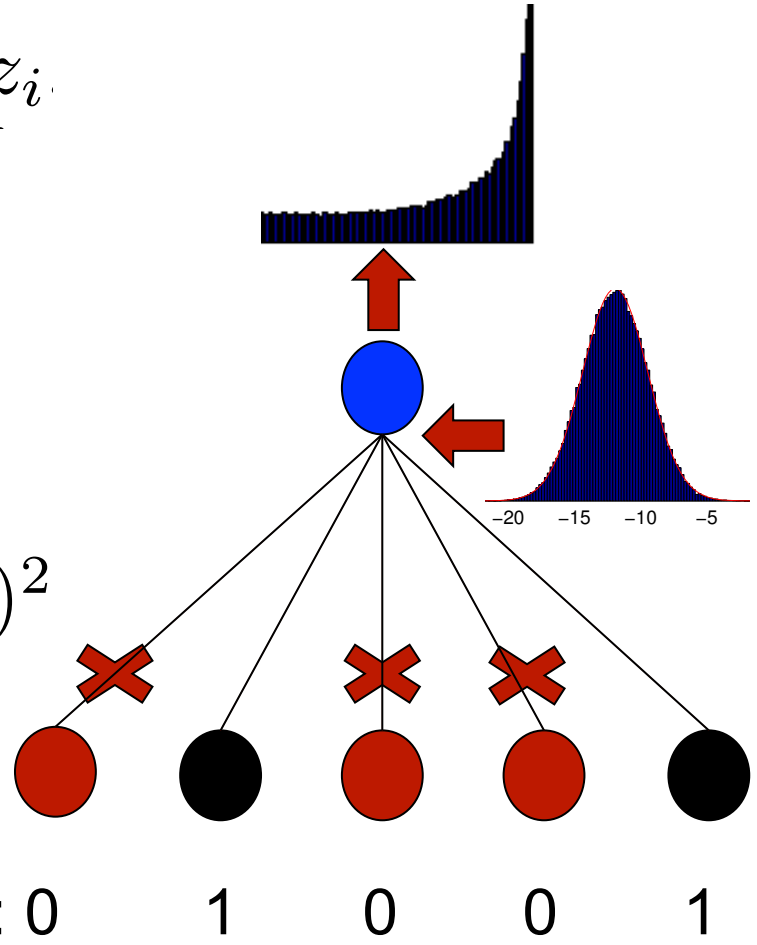
$$Y(z) = w^t D_z x = \sum_i^m w_i x_i z_i$$

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

$$\epsilon \sim \mathcal{N}(0,1),$$

$$E_z[Y(z)] = \sum_i^m p_i w_i x_i$$

$$\text{Var}[Y(z)] = \sum_i^m p_i(1-p_i)(w_i x_i)^2$$



Computing the gradient

- The most naïve E(XY) ≈ E(X)E(Y) does not work

- But we can sample from the Gaussian and linearize

$$\frac{\partial L(w)}{\partial w_i} = E_z[f(Y(z))x_i z_i]$$

$$= \sum_{z_i \in \{0,1\}} p(z_i) z_i x_i E_{z_{-i}|z_i}[f(Y(z))]$$

$$= p(z_i = 1) x_i E_{z_{-i}|z_i=1}[f(Y(z))]$$

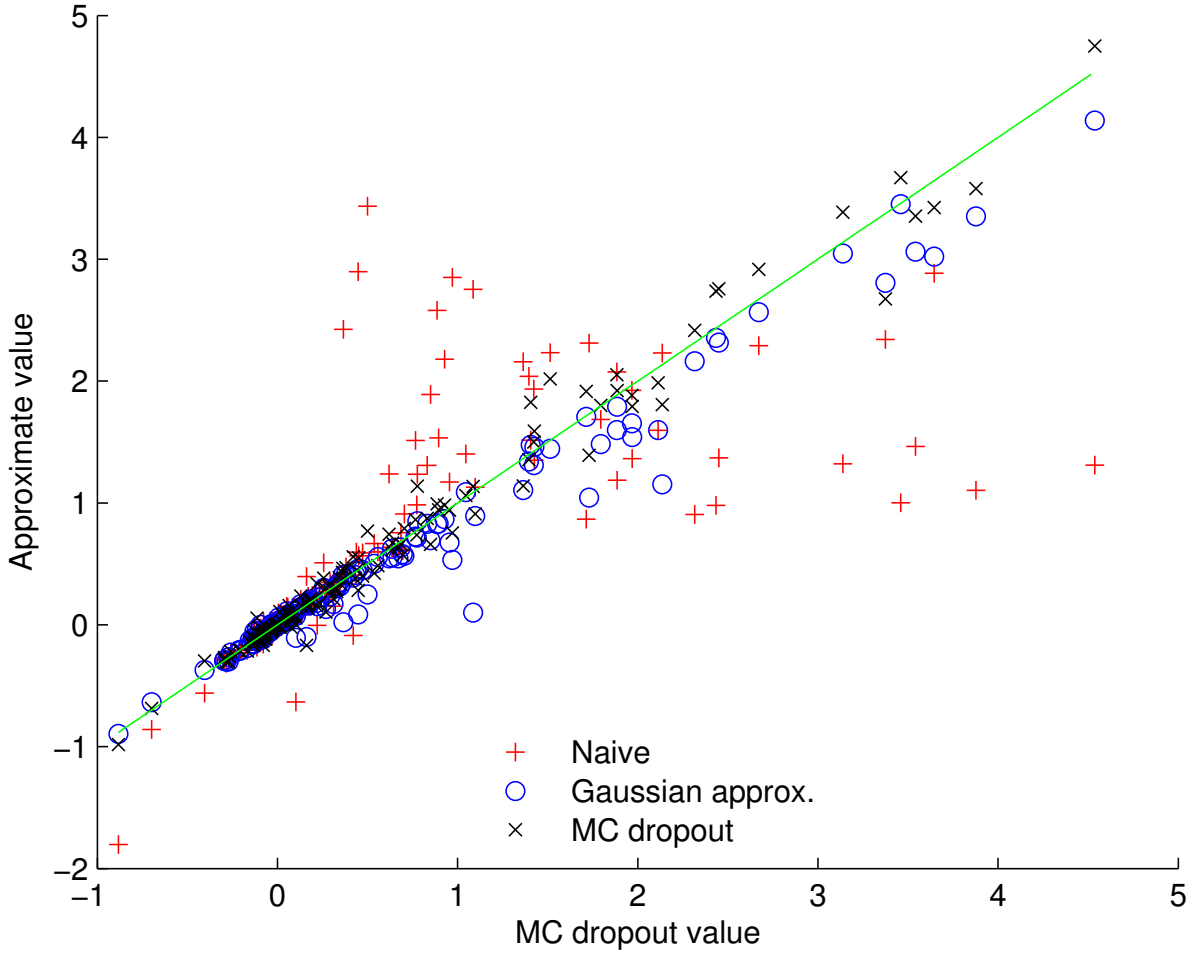
$$\approx p_i x_i \left(E_{S \sim \mathcal{N}(\mu_S, \sigma_S^2)}[f(S)] + \Delta\mu_i \frac{\partial E_{T \sim \mathcal{N}(\mu, \sigma_S^2)}[f(T)]}{\partial \mu}\bigg|_{\mu=\mu_S} + \right.$$

$$\left. \Delta\sigma_i^2 \frac{\partial E_{T \sim \mathcal{N}(\mu_S, \sigma^2)}[f(T)]}{\partial \sigma^2}\bigg|_{\sigma^2=\sigma_S^2} \right)$$

$$= p_i x_i (\alpha(\mu_S, \sigma_S^2) + \Delta\mu_i \beta(\mu_S, \sigma_S^2) + \Delta\sigma_i^2 \gamma(\mu_S, \sigma_S^2))$$

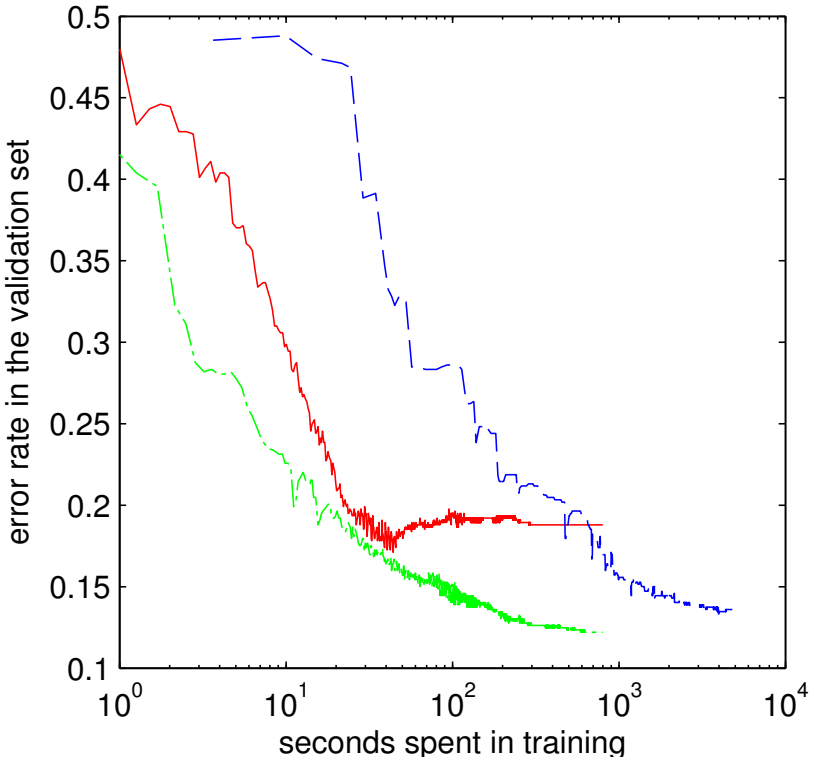
Quality of the gradients

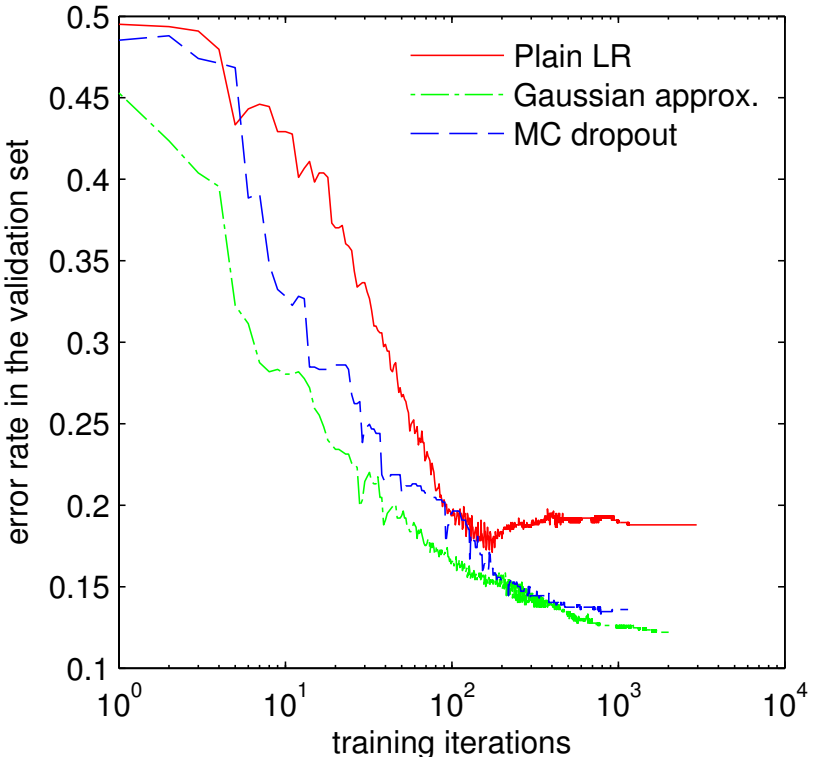
- The approximate gradient vs. true gradient



This method works well

- Reduces complexity from O(Md) to O(M+d) for M samples and data dimension d.





Document classification results

- Improvements over plain LR and most previous methods.

- Average accuracy and time on 9 datasets:

Methods	Plain LR	Dropout	Fast Drop.
Accuracy	84.97	86.88	86.99
Time	92	2661	325

Document classification results

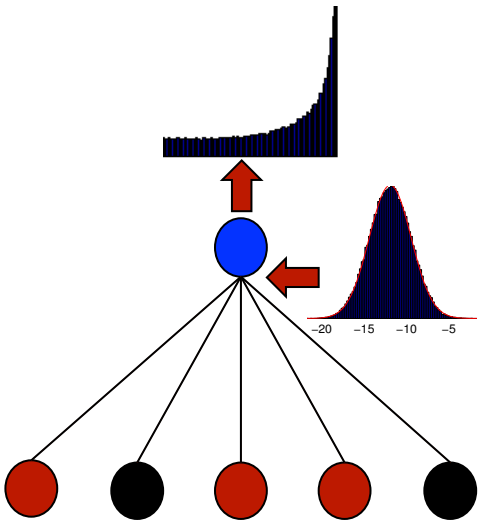
- At or close to state of the art!

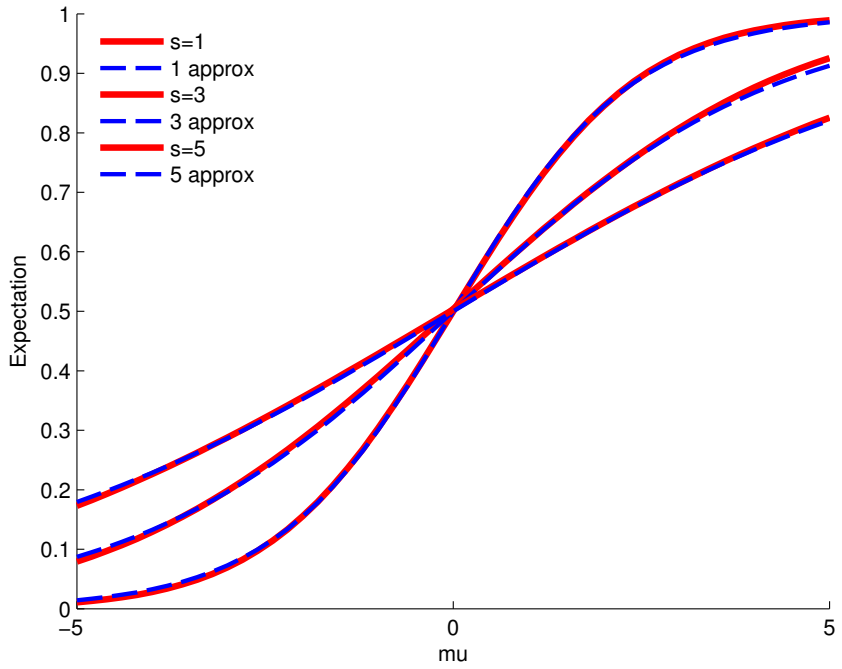
Methods\ Datasets	RT-2k	IMDB	RTs	subj	AthR	CR	MPQA	Average
MC dropout	89.8	91.2	79.2	93.3	86.7	82.0	86.0	86.88
training time	6363	6839	2264	2039	126	582	417	2661
Gaussian approx.	89.7	91.2	79.0	93.4	87.4	82.1	86.1	86.99
training time	240	1071	362	323	6	90	185	325
plain LR	88.2	89.5	77.2	91.3	83.6	80.4	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	

But it still requires sampling!

- We need to compute several expectations
- Could just build a table, and table of partial derivatives, a pain to implement. Here is a good numerical approximation:

$$\int_{-\infty}^{\infty} \sigma(x)\mathcal{N}(x|\mu, s^2)dx \approx \sigma\left(\frac{\mu}{\sqrt{1+\pi s^2/8}}\right)$$





We can even approximate the objective function directly!

- For binary LR, no linearizing and sampling is needed at all!

$$E_{Y\sim\mathcal{N}(\mu,s^2)}[\log(\sigma(Y))] = \int_{-\infty}^{\infty} \log(\sigma(x))\mathcal{N}(x|\mu,s^2)dx$$
$$\approx \sqrt{1+\pi s^2/8}\log\sigma\left(\frac{\mu}{\sqrt{1+\pi s^2/8}}\right)$$

- Works slightly worse in practice compared to sampling
- still retains over 80% of the improvement over baseline

Relation to Bayesian model selection

- We are in effect maximizing a lower bound on the Bayesian evidence:

$$M_\mu = \int p(\mathcal{D}|w)p(w|\mu)dw$$

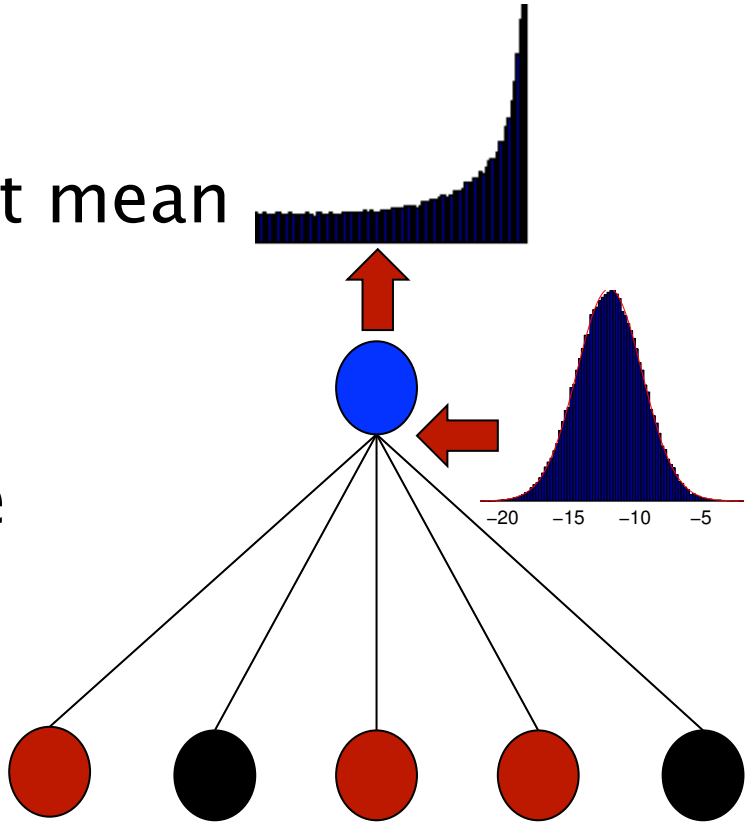
- Under the model $p(w_i|\mu_i) = \mathcal{N}(w_i|\mu_i, \alpha\mu_i^2)$

$$p(y|x,w) = \sigma(yw^T x)$$

$$\begin{aligned}
L(w) &= E_{z;z_i\sim\text{Bernoulli}(p_i)}[\log p(y|w^T D_z x)] \\
&\approx E_{Y\sim\mathcal{N}(E[w^T D_z x],\text{Var}[w^T D_z x])}[\log p(y|Y)] \\
&= E_{v:v_i\sim\mathcal{N}(\mu_i,\alpha\mu_i^2)}[\log p(y|v^T x)] \\
&\leq \log E_{v:v_i\sim\mathcal{N}(\mu_i,\alpha\mu_i^2)}[p(y|v^T x)] \\
&= \log(M_\mu)
\end{aligned}$$

Fast dropout in neural networks

- The fast dropout idea can be applied to neural networks
- Each hidden unit has an input mean and an input variance
- Outputs a mean and variance



Applies to different types of units

- Sigmoid unit:

$$\nu = \int_{-\infty}^{\infty} \sigma(x)\mathcal{N}(x|\mu, s^2)dx \approx \sigma\left(\frac{\mu}{\sqrt{1+\pi s^2/8}}\right)$$

- Rectified linear unit: $f(x) = \max(0, x)$ $\quad r = \mu/s$

$$\nu = \int_{-\infty}^{\infty} f(x)\mathcal{N}(x|\mu, s^2)dx = \Phi(r)\mu + s\mathcal{N}(r|0, 1)$$

- Can also apply to different classifiers, and regression.

Can now train neural networks with deterministic dropout!

- We can now adjust α and add more variance freely if the one decided by dropout is suboptimal

Method name	Number of errors
NN-1200-1200 plain	182
NN-1200-1200 det. Dropout	134
NN-1200-1200 det. Dropout + Var	109
NN-1200-1200 det. Dropout + Var	110
NN-300 MSE [LeCun 1998]	360
NN-800 [Simard 2003]	160
Real dropout [Hinton 2012]	105-120 79 with pretraining

More classification results

- Classification on some more datasets

	SmallM	USPS	Isolet	hepatitis	soybean
NoDrop	87	94.6	90.5	83	**94**
F.Dropout	**90**	**96.3**	**93.2**	**87**	88

Regression

- The final layer is like L2 penalty:

$$s^2 = \sum_j \alpha w_j^2 x_j^2$$

$$E_{X\sim\mathcal{N}(\mu,s^2)}[(X-t)^2] = \int_{-\infty}^{\infty}(x-t)^2\mathcal{N}(x|\mu,s^2)dx$$
$$= s^2 + (\mu - t)^2$$

- Test error (training error) for X-200-100-y

Sq.Error	liver	cardio	housing	CPU
NoDrop	16 (3.1)	4320(90)	39 (29)	1.5 (1.2)
F.Dropout	10 (9.5)	298(217)	35 (32)	1.3 (1.4)

Conclusions

- Dropout training seems promising
- But doing real dropout is slow, sampling is expensive
- Apply the Gaussian approximation
- Cheaper samples or completely deterministic approximations!

The output variance of hidden units

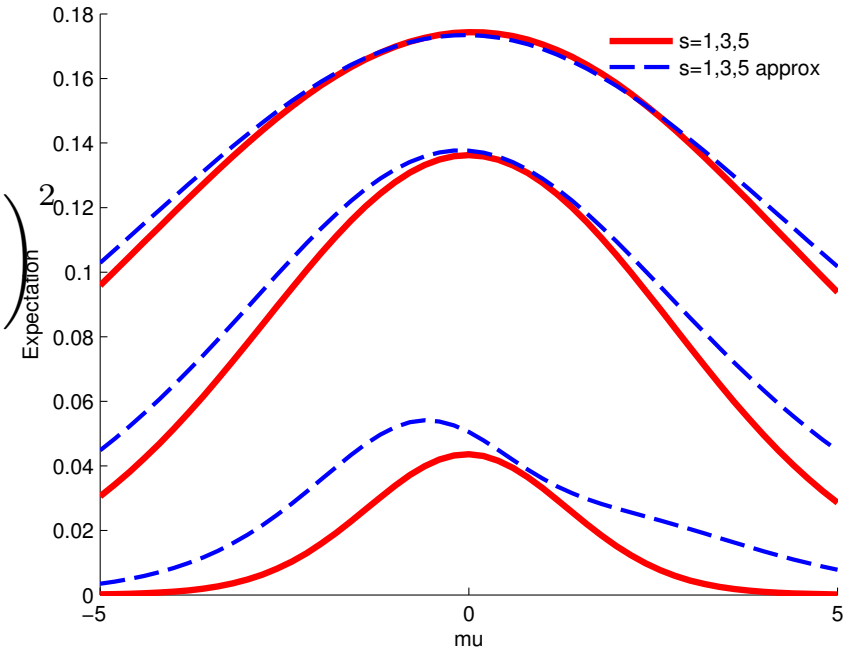
- Wanted an overestimate. Not exact, but fairly accurate

$$\operatorname*{Var}_{Y \sim \mathcal{N}(\mu, s^2)}[\sigma(Y)]$$

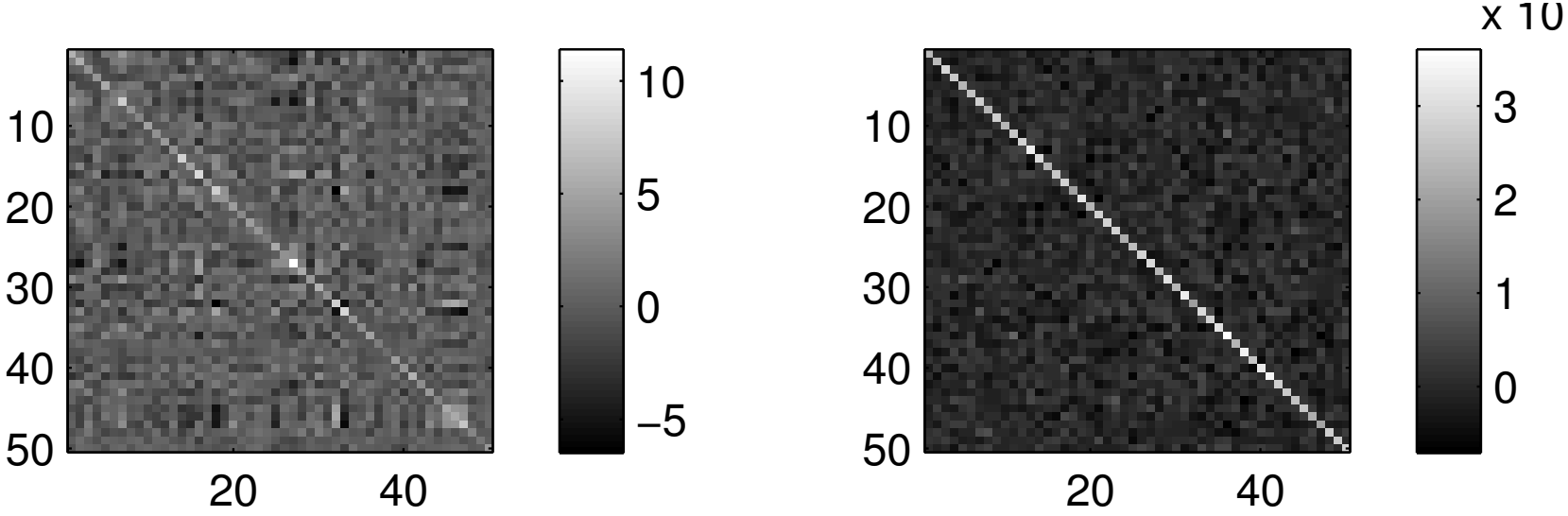
$$= E[\sigma(Y)^2] - E[\sigma(Y)]^2$$

$$\approx E[\sigma(a(Y-b))] - E[\sigma(Y)]^2$$

$$\approx \sigma\left(\frac{a(\mu-b)}{\sqrt{1+\pi/8a^2s^2}}\right) - \sigma\left(\frac{\mu}{\sqrt{1+\pi/8s^2}}\right)^2$$



Covariance matrices



Left: covariance of the inputs to hidden units after training converges. Right: in the beginning

Testing the assumptions

- The training objective functions
  - Left: training on expected cross entropy (dropout)
  - Right: training on cross entropy (plain LR)