The connection of dropout and Bayesian statistics

Interpretation of dropout as approximate Bayesian modelling of NN

http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf

Uncertainty in Deep Learning

Yarin Gal

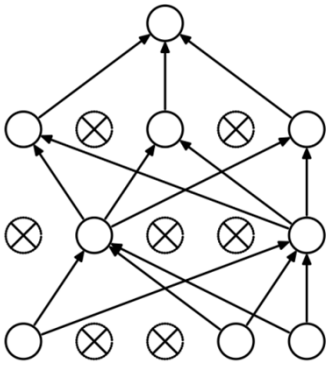
Department of Engineering
University of Cambridge

October 2016

This dissertation is submitted for the degree of
Doctor of Philosophy

Geoffrey Hinton
Google, University Toronto

Dropout



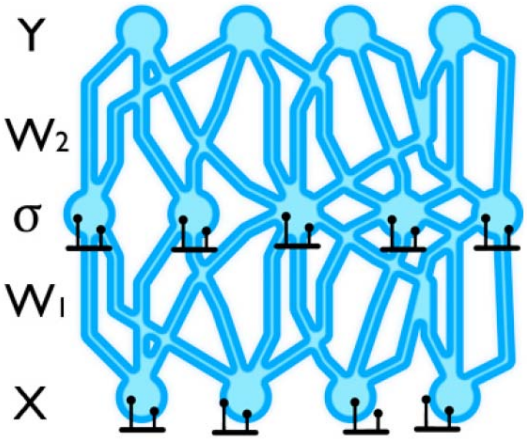
Srivastava et al., *Journal of Machine Learning Research* 15 (2014)

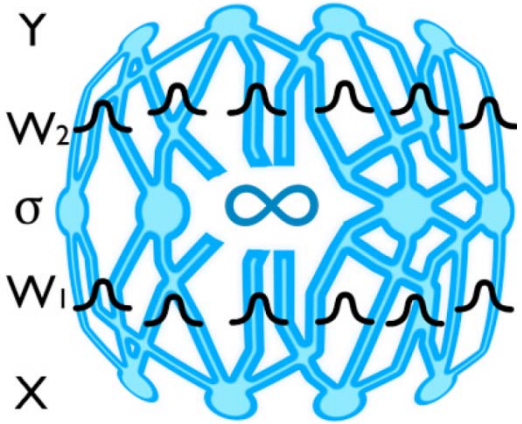
Nitish Srivastava
University Toronto

Presented by Beate Sick, IDP, ZHAW

Outline

- Main principles of frequentists and Bayesians – a reminder
- Neural networks and how they get trained – a reminder
- What is dropout and what is the connection to Bayesian inference?
- How to do approximate Bayesian inference in deep NN
- How to use dropouts to get an uncertainty measure for predictions?
- How can we use the Bayesian approach to make better models?





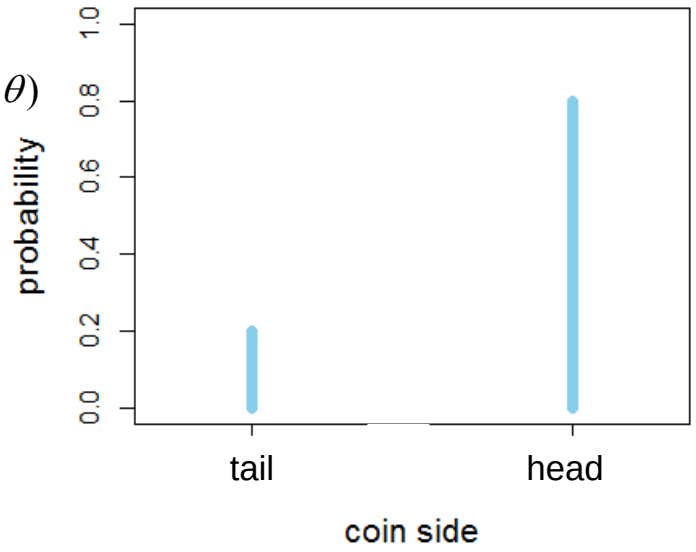
Frequentist's and Bayesian's view on data and model parameters

Frequentist:

- Data are a repeatable random sample
  -> e.g. 10 coin tosses: frequencies 2xtail, 8xheads
  Model: Bernoulli with head-probability as parameter θ
- Underlying model parameters are fixed during this repeatable data sampling
- We use Maximum Likelihood to determine the parameter value under which the observed data is most likely.
- 95% CI means that when repeating the experiment many times 95% of all 95% CI will cover the true fixed parameter.

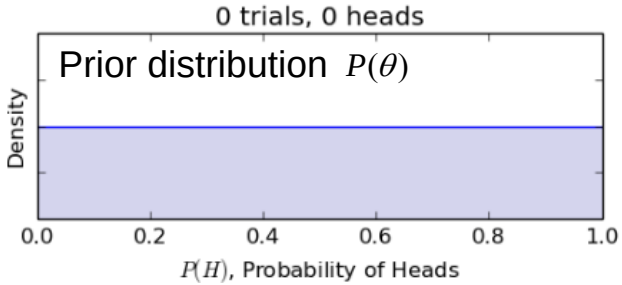
Distribution of data under fixed parameter

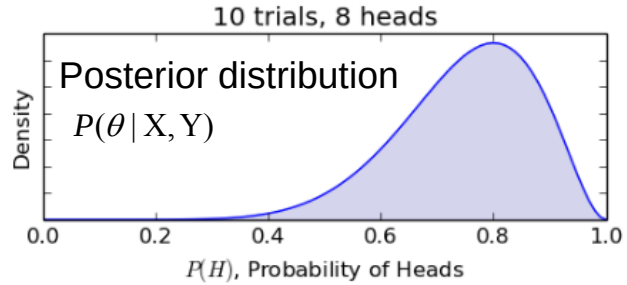
$P(X \mid \theta)$



Bayesian:

- Data are observed once and seen as fixed
- Parameters are described probabilistically
- Before data observation we formulate a prior belief in parameter distribution
- After data observation we updated posterior distribution of model parameters.
- 95% credibility interval means an interval of parameter values that cover 95% of the posterior distribution.





Bayesian modeling has less problems with complex models

Frequentist's strategy:
You can only use a complex model if you have enough data!

Bayesian's strategy:

Use the model complexity you believe in.

Do not just use the best fitting model.

Do use the full posterior distribution over parameter settings leading to vague predictions since many different parameter settings have significant posterior probability.

$$p(\mathrm{y}^*|\mathrm{x}^*,\mathrm{X},\mathrm{Y}) = \int p(\mathrm{y}^*|\mathrm{x}^*,\omega) \cdot p(\omega\,|\mathrm{X},\mathrm{Y})\, d\omega$$

prediction via marginalization over ω:

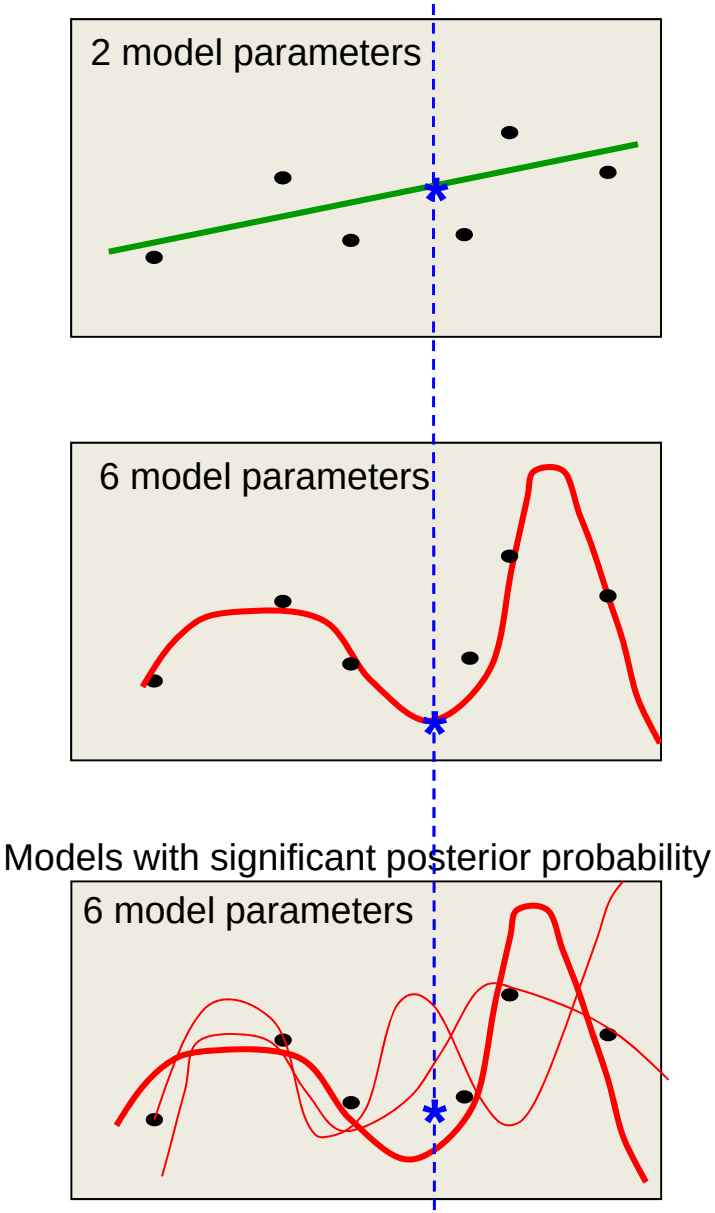
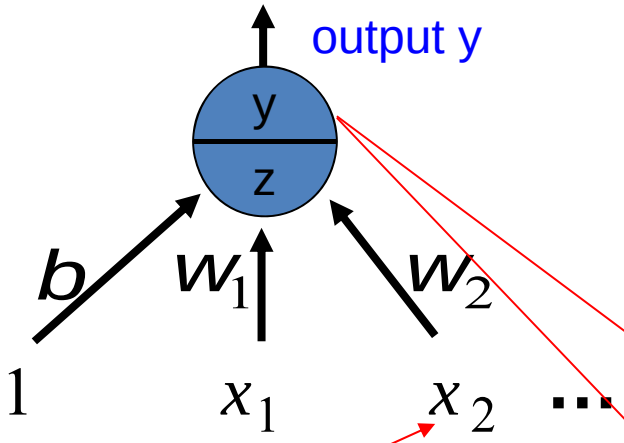


Image credits: Hinton coursera course

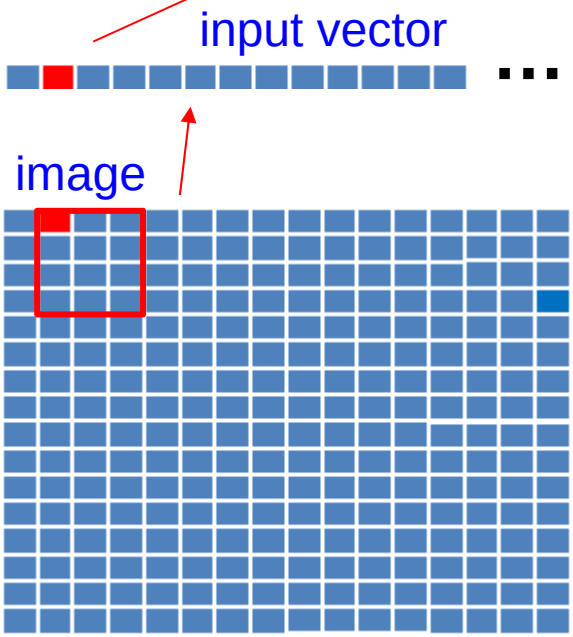
Logistic regression or a neural net with 1 neuron

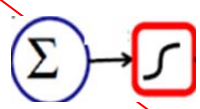


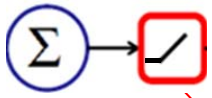
bias i^{th} input

$$z = b + \sum_i x_i w_i$$

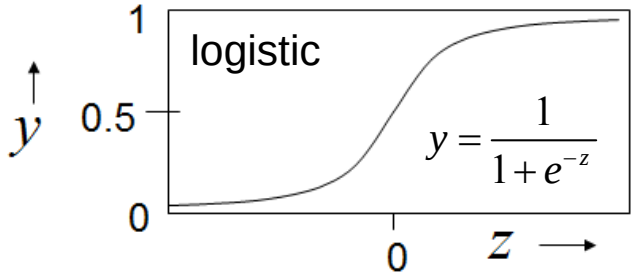
index over input weight on i^{th} input

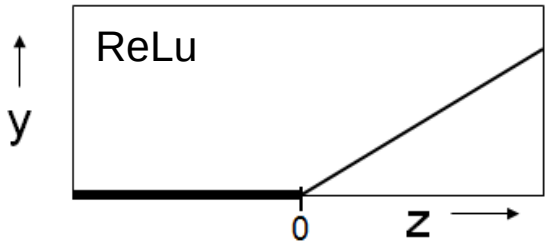






Different non-linear transformations are used to get from z to output y





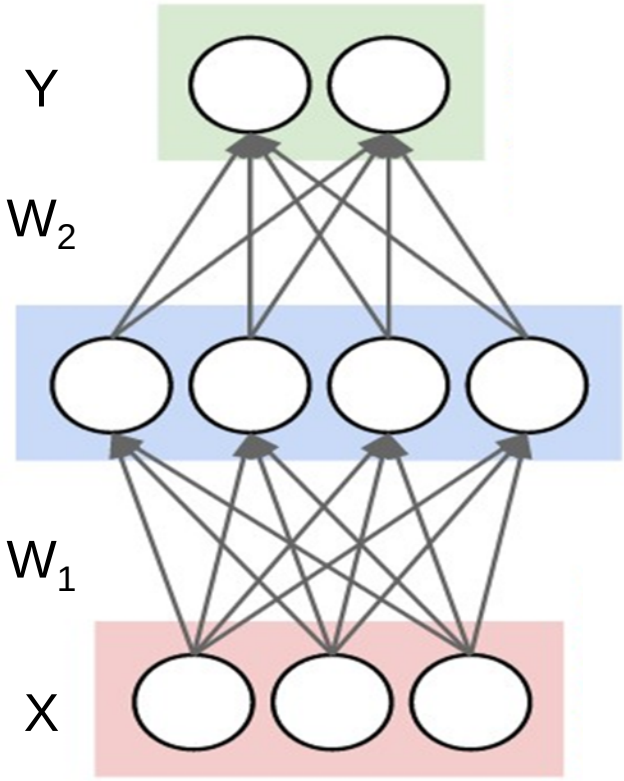
A neural network is a parametric model

We observe

- inputs $\mathbf{X} = \{\mathbf{x}_i\}_{i=1,\ldots,N}$
- outputs $\mathbf{Y} = \{y_i\}_{i=1,\ldots,N}$

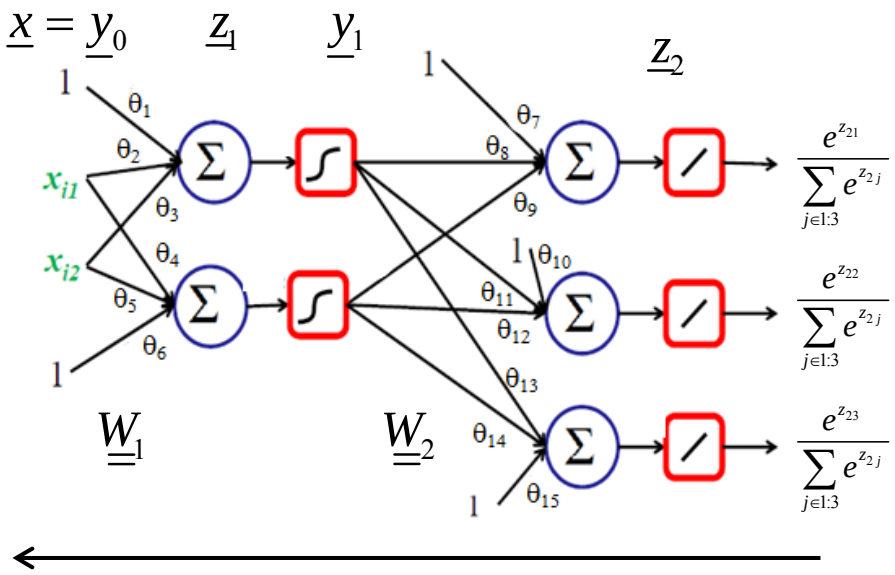
Using a NN as output generating model we can do parametric inference for the connection weights W in the NN that are the parameter defining the model.

The output is $y = f(x, W) = W_2(\sigma(W_1 x + b))$



Fitting a NN classifier via softmax likelihood optimization

Determine all z_s and y_s in forward-pass



Update weights in back-pass

$$P(\mathrm{y=k}) = \mathrm{p}_k = \frac{e^{z_k}}{\sum_{j \in \text{class.id}} e^{z_j}}$$

- LogLikelihood = cost-function C

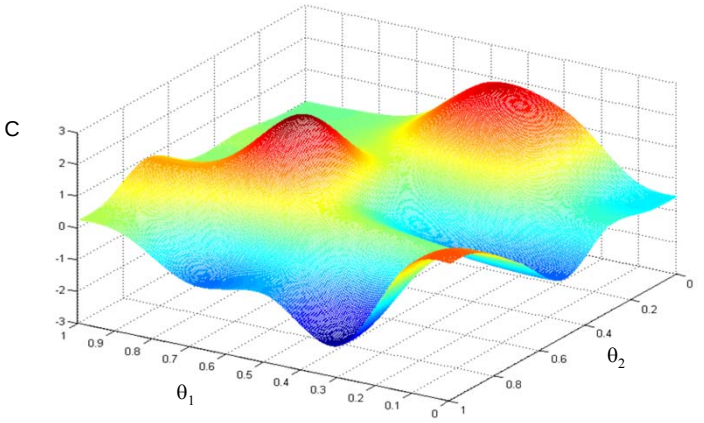
Find sets of weights θ that minimize C!

$$C = -\sum_k I_k \log p_k \quad , \mathrm{I}_k \text{ is indicator for true class k}$$

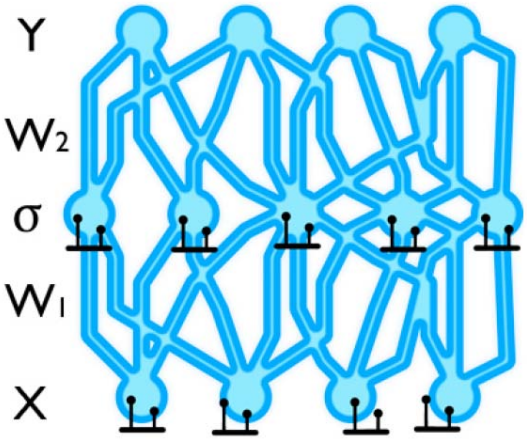
$$\theta_i^{(t)} = \theta_i^{(t-1)} - \eta^{(t)} \left. \frac{\partial C(\theta)}{\partial \theta_i} \right|_{\theta_i = \theta_i^{(t-1)}}$$

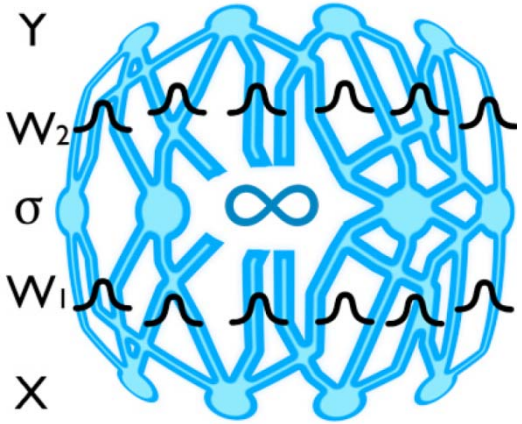
Learning rate

Taking the gradient with chain rule (via backpropagation)



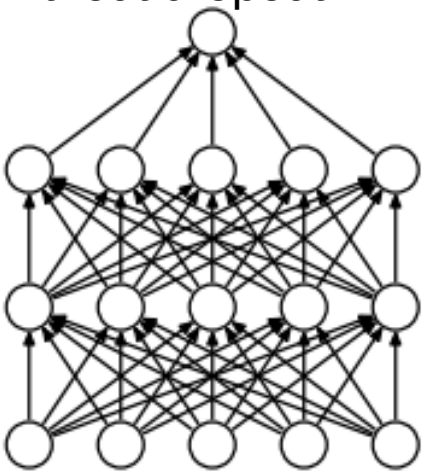
- Main principles of frequentists and Bayesians – a reminder
- Neural networks and how they get trained – a reminder
- What is dropout and what is the connection to Bayesian inference?
- How to do approximate Bayesian inference in deep NN
- How to use dropouts to get an uncertainty measure for predictions?
- How can we use the Bayesian approach to make better models?





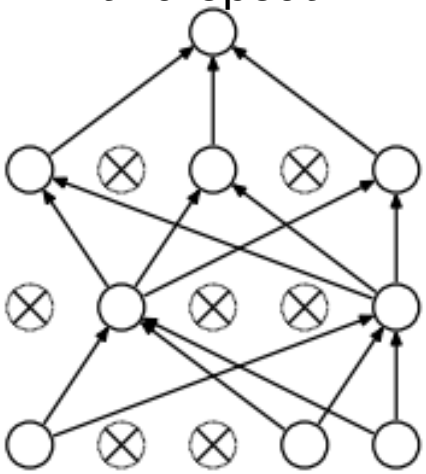
Dropout

without dropout



(a) Standard Neural Net

with dropout



(b) After applying dropout.

“dropout”

At each training step we remove random nodes with a probability of p resulting in a sparse version of the full net and we use backpropagation to update the weights.

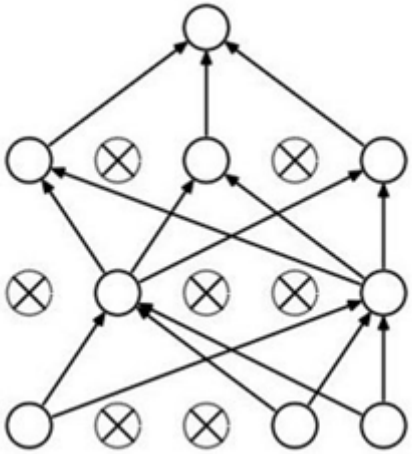
-> In each training step we train another NN model, but all models share weights!

Meaning a certain connection weight is the same in all models at the current update value.

Srivastava et al., *Journal of Machine Learning Research* 15 (2014)

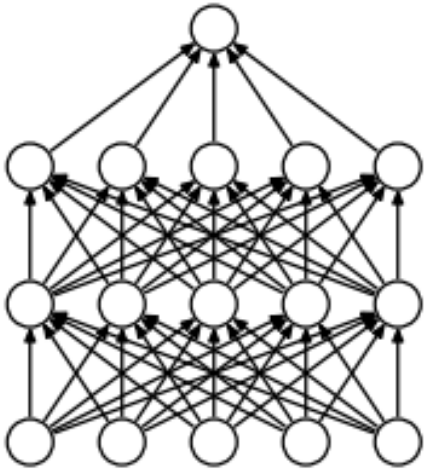
Why "dropout" can be a good idea

Dropout reduces complexity of the model and thus overfitting ?



Using dropout during training implies:

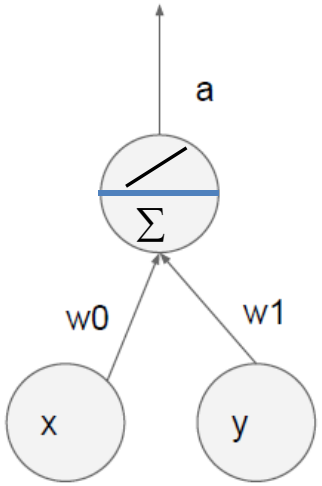
- We train in each training step another sparse model. Only weights to not-dropped units are updated
- Sharing weights may introduce "regularization"
- By averaging over these models we should be able to "reduce noise", "overfitting"



Use the trained net without dropout during test time

But we need to do a small adjustment!

Why "dropout" can be a good idea



Use the trained net without dropout during test time

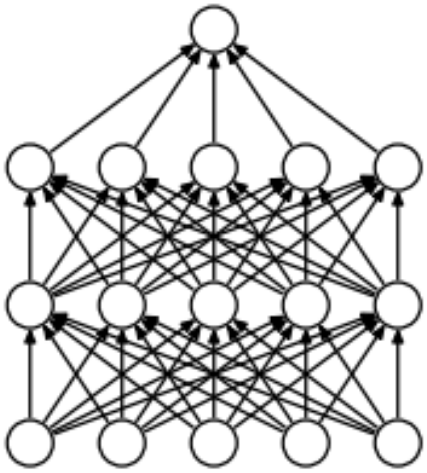
Q: Suppose that with all inputs present at test time and the output of this neuron is x.

What would its output be during training time, in expectation? (e.g. if p = 0.5)

during test: **a = w0*x + w1*y**

during training with dropout:

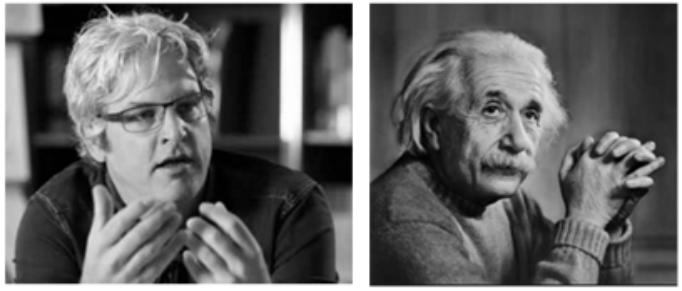
$$E[a] = \tfrac{1}{4} * (w0*0 + w1*0 + w0*0 + w1*y + w0*x + w1*0 + w0*x + w1*y)$$
$$= \tfrac{1}{4} * (2\, w0*x + 2\, w1*y)$$
$$= \tfrac{1}{2} * \mathbf{(w0*x + w1*y)}$$



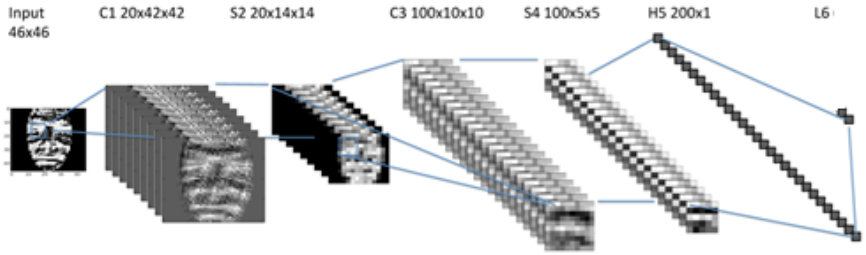
Using all units in the test time forward pass leads to an output that 2x the expected output during training
(this is true for linear units without hidden layers).
=> Have to compensate by reducing the weights during test time by multiplying them by the dropout probability p=0.5

Why "dropout" can be a good idea

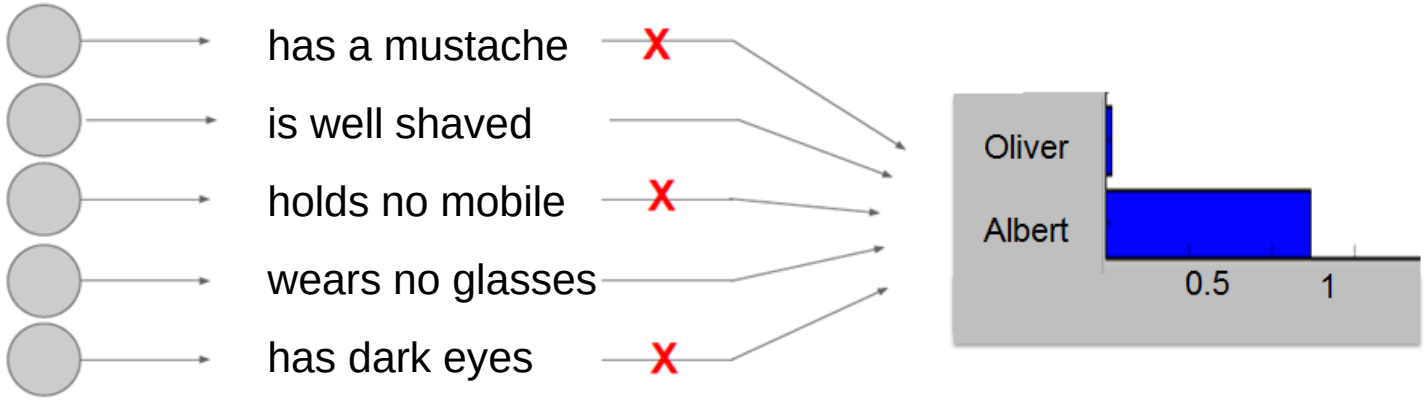
The training data consists of many different pictures of Oliver Dürr and Albert Einstein



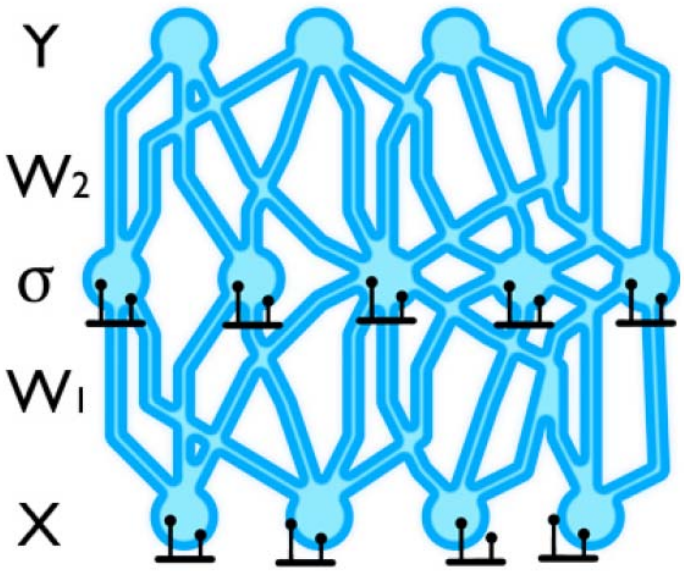
We need a huge number of neurons to extract good features which help to distinguish Oliver from Einstein



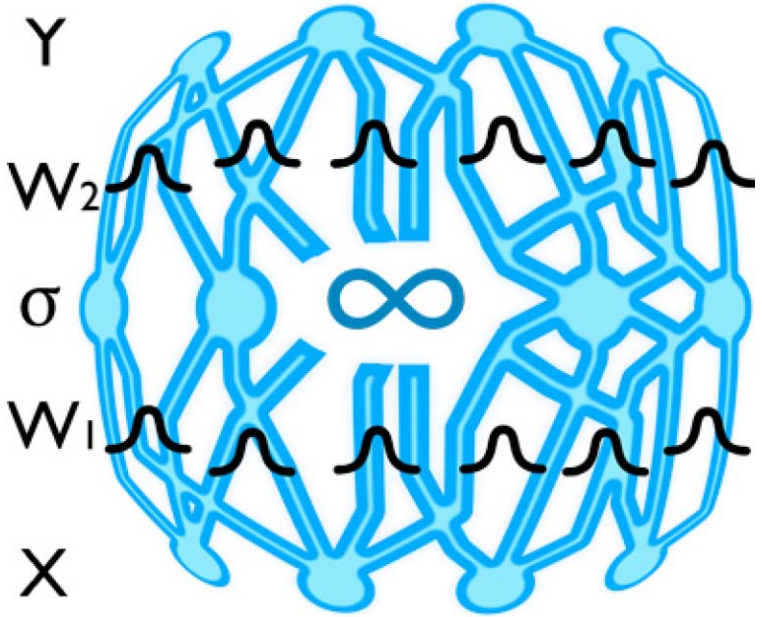
Dropout forces the network to learn redundant and independent features



We will see that dropout is an approximation of full Bayesian learning using a Bayesian Network



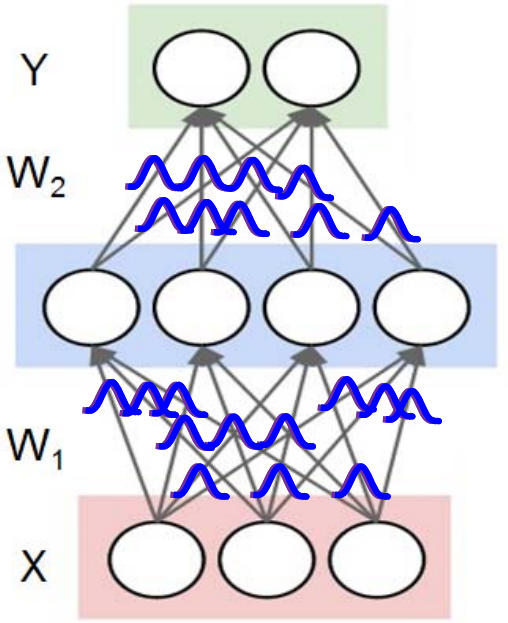
“dropout”
At each training step we remove random nodes with a probability p



“Bayesian NN”
At each training step we update the posterior distribution of the weights
In test time we use the posterior to determine the predictive distribution

Bayesian Modeling in the framework of NN

- In Bayesian Modeling we define a prior distribution over the parameter W: $W_i \sim N(0, I)$ defining p(w_i)



- For classification NN tasks we assume a softmax likelihood

$$P(\mathrm{y}=\mathrm{k} \mid \omega, \mathrm{x}) = \mathrm{p}_k = \frac{e^{\mathrm{f}_{\mathrm{k},\omega}(\mathrm{x})}}{\sum_{k'} e^{\mathrm{f}_{\mathrm{k},\omega}(\mathrm{x})}}$$

- Given a dataset X, Y we then look for the posteriori distribution capturing the most probable model parameter given the observed data

likelihood prior

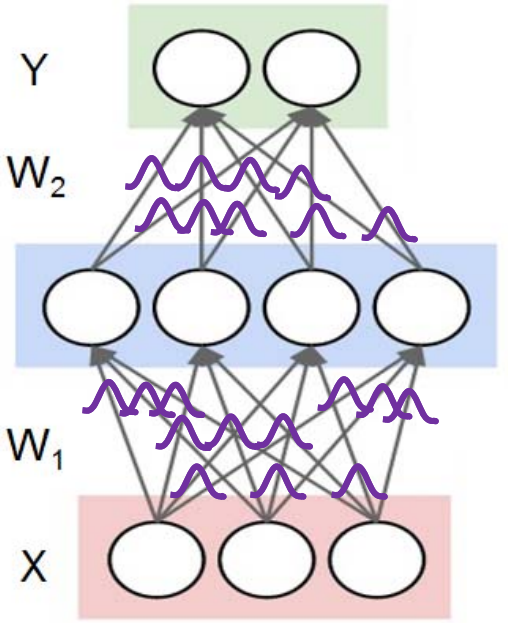
$$p(\omega \mid \mathrm{X}, \mathrm{Y}) = \frac{p(\mathrm{Y} \mid \omega, \mathrm{X}) \cdot p(\omega)}{p(\mathrm{Y} \mid \mathrm{X})}$$

normalizer=marginal likelihood

Marginalization steps in Bayesian statistics

- We can use the posterior distribution

$$p(\mathrm{w}|\mathrm{X},\mathrm{Y}) = \frac{p(\mathrm{Y}|\mathrm{w},\mathrm{X}) \cdot p(\mathrm{w})}{p(\mathrm{Y}|\mathrm{X})}$$



-> to determine the most probable weight vector
MAP: Maximum a Posteriori

-> to determine credible intervals

-> sample weight vectors from posterior distribution

- For the posteriori we need to determine the normalizer, also called model evidence via integrating over all possible parameters weighted by their probabilities. This process is also called marginalization over ω leading to the name marginal likelihood

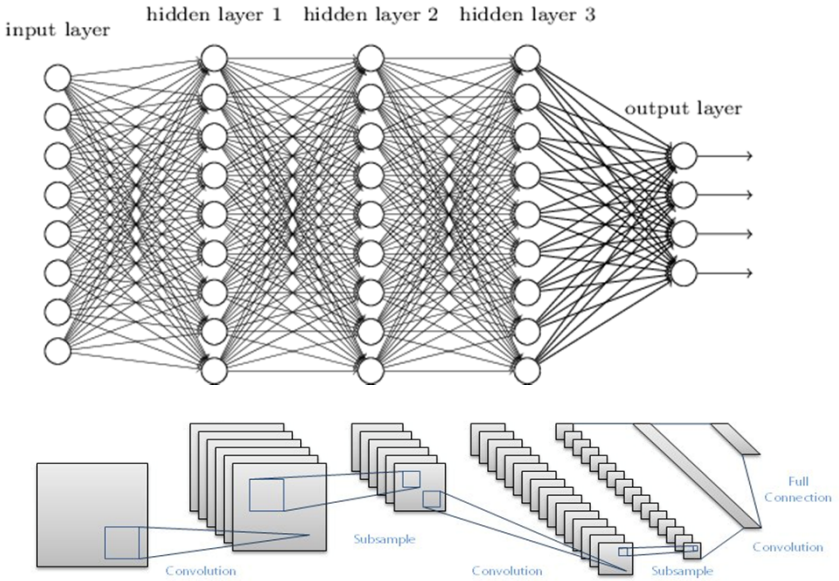
$$p(\mathrm{Y}|\mathrm{X}) = \int p(\mathrm{Y}|\mathrm{X},\omega) \cdot p(\omega) d\omega$$

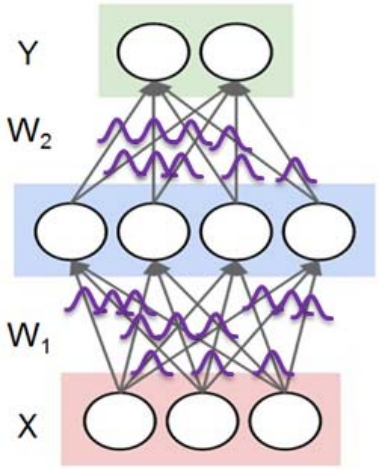
Using a neural network for prediction

A deep NN or deep CNN

- Performs a hierarchical non-linear transformation of its input $y = f(x, W)$
- These models give us only point estimates with no uncertainty information.

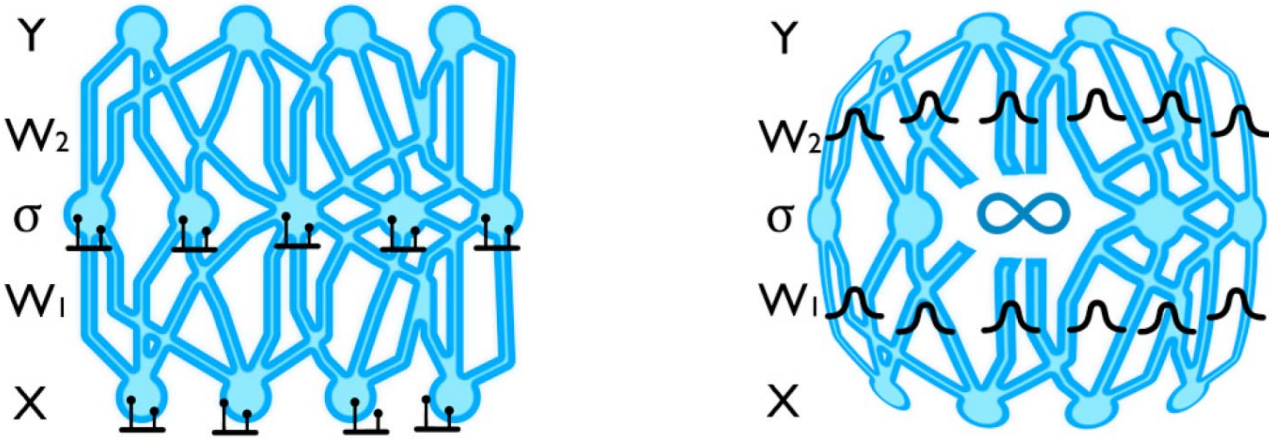
Remark: for shallow NN CIs for the weights can be determined if the network is identified (White, 1998)





With Bayesian modeling we can get a measure of uncertainty by evaluating the posterior distribution of the NN weights.

- Main principles of frequentists and Bayesians – a reminder
- Neural networks and how they get trained – a reminder
- What is dropout and what is the connection to Bayesian inference?
- How to do approximate Bayesian inference in deep NN
- How to use dropouts to get an uncertainty measure for predictions?
- How can we use the Bayesian approach to make better models?



Full Bayesian learning and ideas for approximations

Full Bayesian learning means computing the full posterior distribution over all possible parameter settings.

- This is extremely computationally intensive for all but the simplest models
- But it allows us to use complicated models with not so much data

Approximation of full Bayesian learning is the only way if we have models with many (>100) parameters or complex models.

- We can use Markov Chain Monte Carlo (MCMC) methods to sample parameter vectors ω according to their posterior probabilities at the observed data - $p(\omega|X,Y)$ – and use them to estimate the distribution $q(\omega)$.

- We can approximate the posterior distribution for the model parameters via

  a) Integrated Nested Laplace Approximations (INLA)

  b) Variational inference

  - replacing the posterior distribution at the observed data $p(\omega|X,Y)$ with a member $q(\omega)$ of a simpler distribution family Q that minimizes the Kullback–Leibler divergence to the posterior

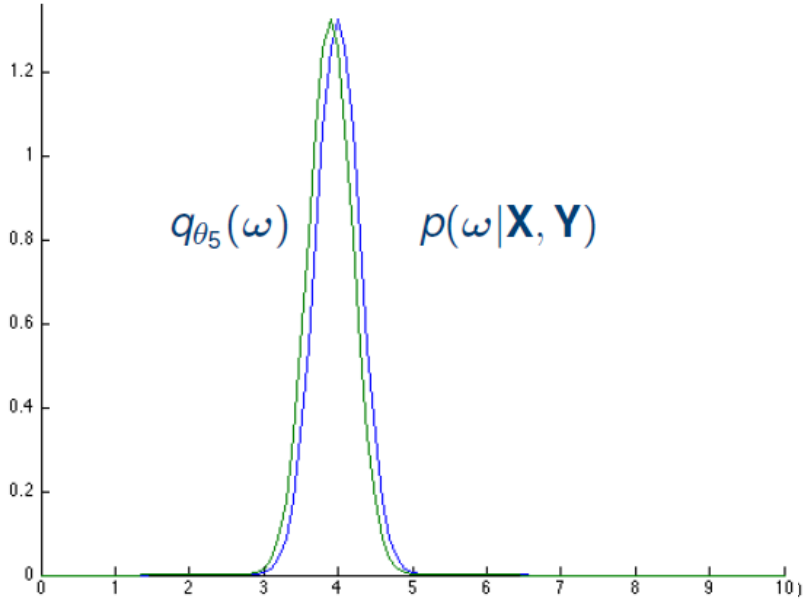
Variational inference approximates full Bayesian learning

- Approximate p(ω|**X**;**Y**) with simple distribution $q_\theta(\omega)$
- Minimize Kullback Leibler divergence of q from the posterior w.r.t.

  to the variational parameters θ:

$$\mathrm{KL}\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega \,|\, \mathbf{X}, \mathbf{Y})\right) = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{\mathrm{p}(\omega \,|\, \mathbf{X}, \mathbf{Y})} d\omega = E_q\left[\log\left(q_\theta(\omega)\right) - \log\left(\mathrm{p}(\omega \,|\, \mathbf{X}, \mathbf{Y})\right)\right]$$

Minimizing KL means to wiggle the parameter θ of q to find the value of θ for which q resembles the posterior distribution as good as possible.



The term variational is used because you pick the **best** q in Q -- the term derives from the "calculus of variations," which deals with optimization problems. A particular q in Q is specified by setting some variational parameters -- the parameter which is adjusted during optimization.

Variational inference to approximate Bayesian learning ctd.

Minimizing the KL divergence of q from the posterior dist. p w.r.t. θ

$$\mathrm{KL}\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega \,|\, \mathbf{X}, \mathbf{Y})\right) = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{\mathrm{p}(\omega \,|\, \mathbf{X}, \mathbf{Y})} d\omega$$

is equivalent to maximizing a lower bound of the log marginal likelihood w.r.t. θ
(also called *evidence lower bound* or ELBO)

$$\mathrm{L}_{VI}(\theta) := \int q_\theta(\omega) \cdot \log\left(p\left(\mathrm{Y}|\mathrm{X},\omega\right)\right) d\omega - KL\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\right) \leq \mathrm{L} = \log\left(p(\mathrm{Y} \,|\, \mathrm{X})\right)$$

The key trick is here like in most variational methods, to write the true log-likelihood L as the log of an expectation under some q. We can then get a lower bound via Jensen's inequality, which tells us that log expectation >= expectation log (since log is concave and q is a distribution).

$$\mathrm{L} = \log\left(p(\mathrm{Y} \,|\, \mathrm{X})\right) = \log \int p\left(\mathrm{Y}|\mathrm{X},\omega\right) \cdot p\left(\omega\right) d\omega = \log \int p\left(\mathrm{Y}|\mathrm{X},\omega\right) \cdot p\left(\omega\right) \frac{q_\theta(\omega)}{q_\theta(\omega)} d\omega = \log\left(E_{q_\theta}\left[\frac{p\left(\mathrm{Y}|\mathrm{X},\omega\right) \cdot p\left(\omega\right)}{q_\theta(\omega)}\right]\right)$$

$$\geq E_{q_\theta}\left[\log\left(\frac{p\left(\mathrm{Y}|\mathrm{X},\omega\right) \cdot p\left(\omega\right)}{q_\theta(\omega)}\right)\right] = E_{q_\theta}\left[\log\left(p\left(\mathrm{Y}|\mathrm{X},\omega\right)\right) + \log\left(\frac{p\left(\omega\right)}{q_\theta(\omega)}\right)\right] = E_{q_\theta}\left[\log\left(p\left(\mathrm{Y}|\mathrm{X},\omega\right)\right)\right] - E_{q_\theta}\left[\log\left(\frac{q_\theta(\omega)}{p\left(\omega\right)}\right)\right]$$

$$= \int q_\theta(\omega) \cdot \log\left(p\left(\mathrm{Y}|\mathrm{X},\omega\right)\right) d\omega - KL\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\right)$$

Variational inference to approximate Bayesian learning ctd.

The best θ that makes q_θ a good approximation for the posterior is given by the θ that maximizes the following objective as lower bound of L:

$$\mathrm{L}_{VI}(\theta) := \int q_\theta(\omega) \cdot \log\Big(p\big(\mathrm{Y}|\mathrm{X},\omega\big)\Big) d\omega - KL\big(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\big)$$

Since this integral is not tractable for almost all q therefore we will MC integration to approximate this quantity.

We will sample $\hat{\omega}$ from q and look at the following term where for each sampling step the integral is replaced by $\log(p(Y|X,\hat{\omega}))$ leading to a new objective:

$$\hat{\mathrm{L}}(\theta) := \log\Big(p\big(\mathrm{Y}|\mathrm{X},\hat{\omega}\big)\Big) - KL\big(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\big)$$

In approximate Bayesian statistics it is known, that $\hat{L}$ is an unbiased estimator of L.

Variational inference to approximate Bayesian learning ctd.

$$\mathrm{L}_{VI}(\theta) := \int q_\theta(\omega) \cdot \log\left(p\left(\mathrm{Y}|\mathrm{X},\omega\right)\right) d\omega - KL\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\right)$$

$$\hat{\mathrm{L}}(\theta) := \log\left(p\left(\mathrm{Y}|\mathrm{X},\hat{\omega}\right)\right) - KL\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\right)$$

Results from stochastic inference guarantee that optimizing $\hat{L}$ w.r.t. θ will converge to the same optimal θ than optimizing L_{VI} w.r.t. θ.

To stepwise optimize $\hat{L}$ we sample in each step one $\hat{\omega}$ from $q_\theta(\omega)$ and do one step of optimization of θ and then we update $q_\theta(\omega)$ before sampling the next $\hat{\omega}$.

For inference repeatedly do:

a) Sample $\hat{\omega} \sim q_\theta(\omega)$
b) Do one step of minimization w.r.t. θ:

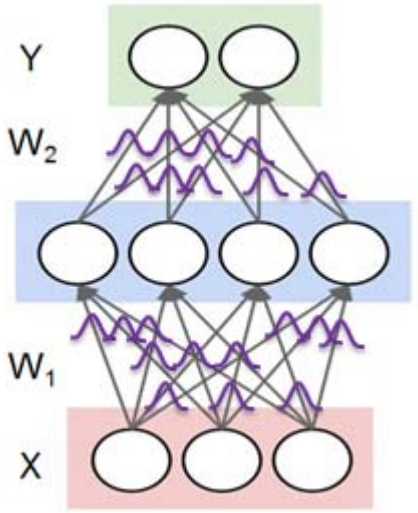
$$\hat{\mathrm{L}}(\theta) := \log\left(p\left(\mathrm{Y}|\mathrm{X},\hat{\omega}\right)\right) - KL\left(q_\theta(\omega) \,||\, \mathrm{p}(\omega)\right)$$

What kind of q-distribution should we use to resemble dropout with Bayesian learning?

Define the structure of the approximate distribution q

We define q to factorizes over weight matrices $\mathbf{W}_i$ of the layers 1 to L.

$$q_\theta(\omega) = \prod_i q_{\mathbf{M}_i}(\mathbf{W}_i)$$



For each $\mathbf{W}_i$ we define q to be the product of the mean weight matrix $\mathbf{M}_i$ and a diagonal matrix which holds Bernoulli variables on the diagonal.

$$q_{\mathbf{M}_i}(\mathbf{W}_i) = \mathbf{M}_i \cdot \mathrm{diag}\left(\left[\mathbf{z}_{i,j}\right]\right)$$

$$\mathbf{M}_i = \mathrm{mean}(\mathbf{W}_i)$$

$\mathbf{M}_i$ is as variational parameter of q

$$\mathbf{z}_{i,j} \sim \mathrm{Bernoulli}(\mathrm{p}_i)$$

$$\mathbf{W}_i \sim q_{\mathbf{M}_i}(\mathbf{W}_i)$$

Approximate posterior of model parameters

Sampling the diagonal elements **z** from a Bernoulli is identical to randomly setting columns of M to zero which is identical to randomly setting units of the network to zero -> dropout!

For derivations and proofs see: http://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf

Why this structure of the approximate distribution q

- it resembles dropout!
  We know what to do during test time: use full NN and set $\mathbf{W}_i = p_i \mathbf{W}_i$

- Bernoullis are computationally cheap to get multi-modality

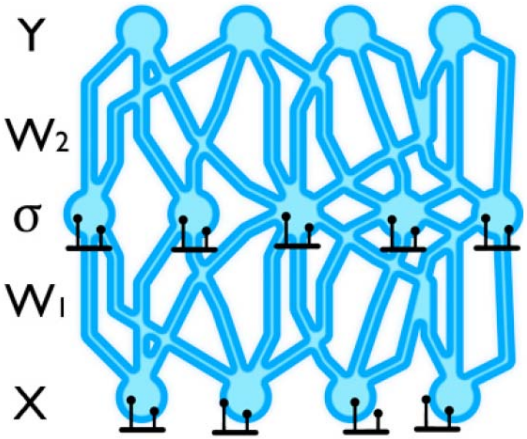
- This q constrains the weights to be near zero:
  - for this q structure the variance of the weights are:

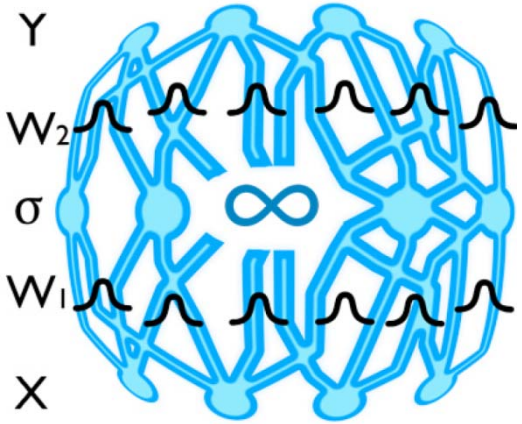
$$\mathrm{Var}(\mathbf{W}_i) = \mathbf{M}_i \cdot \mathbf{M}_i^T \cdot p_i \cdot (1 - p_i)$$

  - we know that posterior uncertainty decreases with more data implying that for fixed dropout probability p_i the average weights need to decrease with more data.

The strongest regularization is achieved with dropout probability $p_i = 0.5$.

- Main principles of frequentists and Bayesians – a reminder
- Neural networks and how they get trained – a reminder
- What is dropout and what is the connection to Bayesian inference?
- How to do approximate Bayesian inference in deep NN
- How to use dropouts to get an uncertainty measure for predictions?
- How can we use the Bayesian approach to make better models?

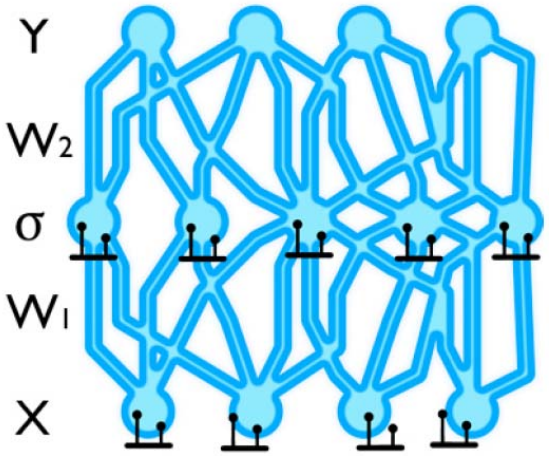




DL with dropout as approximate Bayesian inference allows to get a prediction distribution for uncertainty estimation

To get an approximation of the posterior via training

- Randomly set columns of $\mathbf{M}_i$ to zero (do dropout)
- Update the weights by doing one step



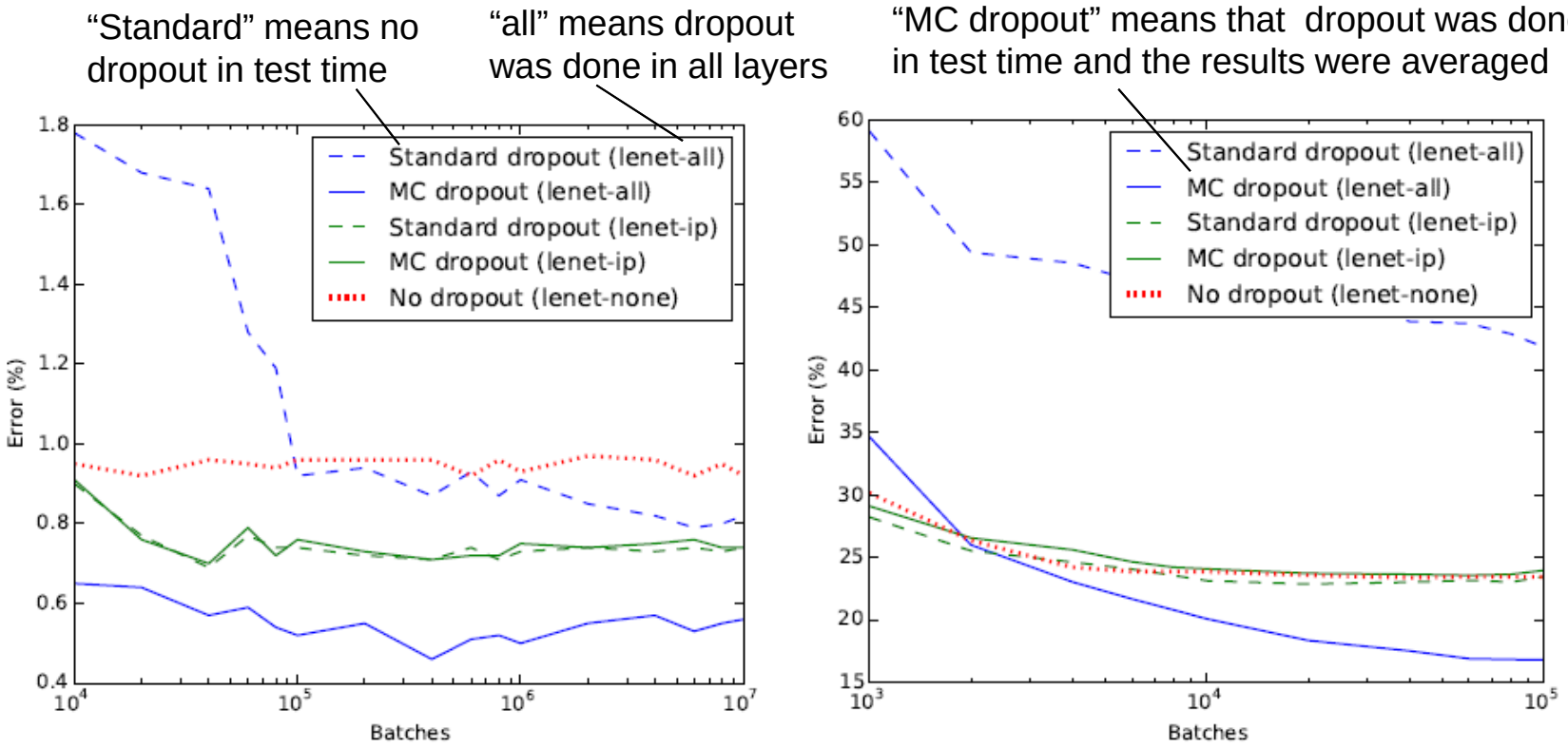
To sample from the learned approximate posterior we just can do dropout during the test time when using the trained NN for prediction.

From the received predictions we can estimate the predictive distribution and from this different uncertainty measures such as the variance.

Can this insights help us to make better models?

Yes – just do dropout also in the convolutional layer during training and test time!

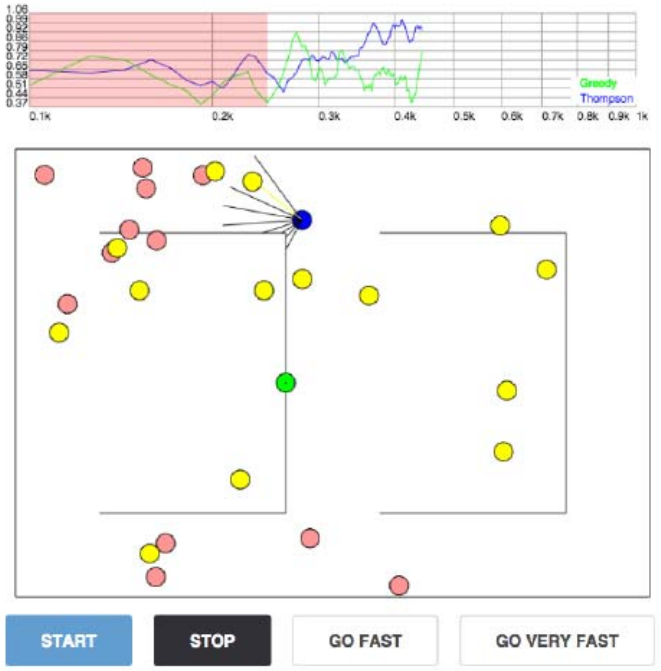
MC dropout is equivalent to performing *several* stochastic forward passes through the network and averaging the results. By doing so Yarin Gal was able to outperform state of the art error rates on MNIST and CIFAR-10 without changing the architecture of the used CNNs or anything else beside dropout.



(a) MNIST (b) CIFAR-10

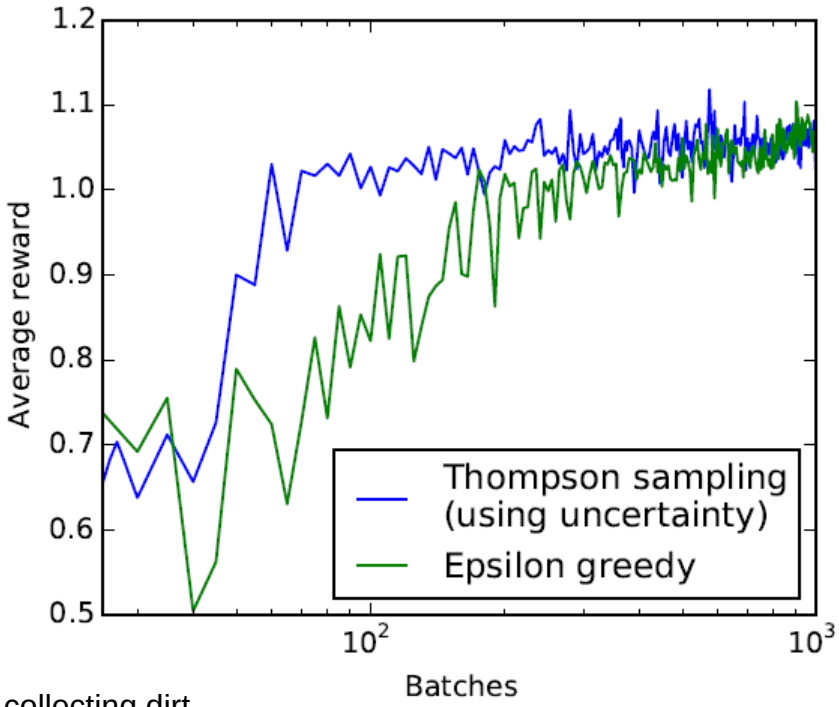
Dropout approach speeds up reinforcement learning

Example: train a Roomba vacuum cleaner via enforcement learning



[Online demo] [13]

http://mlg.eng.cam.ac.uk/yarin/blog_3d801aa532c1ce.html#graph_canvas



Reward for collecting dirt
Penalty for bumping into walls

Without dropout: take random action with probability ε and the model-predicted action otherwise (Epsilon-greedy)

With dropout: Do several stochastic forward passes through the dropout network and choose action with highest value (Thompson sampling)

Thank you for your attention

How good is the dropout approach?

Compare performance of dropout-method with well established and published variational inference (VI) and probabilistic back-propagation (PBP) results on the basis of root-mean-square-error (RMSE) and log-likelihood (LL).

The better fitting the model is the smaler is the RMSE.

The smaller the uncertainty of the model is the larger is the LL.

	Avg. Test RMSE and Std. Errors			Avg. Test LL and Std. Errors		
Dataset	**VI**	**PBP**	**Dropout**	**VI**	**PBP**	**Dropout**
Boston Housing	4.32 ±0.29	3.01 ±0.18	**2.97 ±0.85**	-2.90 ±0.07	-2.57 ±0.09	**-2.46 ±0.25**
Concrete Strength	7.19 ±0.12	5.67 ±0.09	**5.23 ±0.53**	-3.39 ±0.02	-3.16 ±0.02	**-3.04 ±0.09**
Energy Efficiency	2.65 ±0.08	1.80 ±0.05	**1.66 ±0.19**	-2.39 ±0.03	-2.04 ±0.02	**-1.99 ±0.09**
Kin8nm	**0.10 ±0.00**	**0.10 ±0.00**	**0.10 ±0.00**	0.90 ±0.01	0.90 ±0.01	**0.95 ±0.03**
Naval Propulsion	**0.01 ±0.00**	**0.01 ±0.00**	**0.01 ±0.00**	3.73 ±0.12	3.73 ±0.01	**3.80 ±0.05**
Power Plant	4.33 ±0.04	4.12 ±0.03	**4.02 ±0.18**	-2.89 ±0.01	-2.84 ±0.01	**-2.80 ±0.05**
Protein Structure	4.84 ±0.03	4.73 ±0.01	**4.36 ±0.04**	-2.99 ±0.01	-2.97 ±0.00	**-2.89 ±0.01**
Wine Quality Red	0.65 ±0.01	0.64 ±0.01	**0.62 ±0.04**	-0.98 ±0.01	-0.97 ±0.01	**-0.93 ±0.06**
Yacht Hydrodynamics	6.89 ±0.67	**1.02 ±0.05**	1.11 ±0.38	-3.43 ±0.16	-1.63 ±0.02	**-1.55 ±0.12**
Year Prediction MSD	9.034 ±NA	8.879 ±NA	**8.849 ±NA**	-3.622 ±NA	-3.603 ±NA	**-3.588 ±NA**

Table 1: **Average test performance in RMSE and predictive log likelihood** for a popular variational inference method (VI, Graves [20]), Probabilistic back-propagation (PBP, Hernández-Lobato and Adams [27]), and dropout uncertainty (Dropout).