Bayesian approaches to machine learning

Pierre Geurts

Institut Montefiore, University of Liège, Belgium

INFO8004
Advanced Machine Learning
Feb 14, 2019

Outline

Part 1: Introduction to bayesian approaches in ML
Part 2: Dropout as a bayesian approximation

Part I

Introduction to bayesian approaches in ML

Outline

Outline

Bayesian methods: general principles

The main idea of Bayesian methods is to use probability theory to model all kind of uncertainties, ie.,

- Uncertainties due to **physical randomness** associated with the data, e.g., noises, errors in labeling.
- But also uncertainties about **models** and their **parameters**.

Then, all prediction problems are reduced to making probabilistic inference using this model (by inverting conditional probabilities through Bayes' rule).

A non-Bayesian approach would only model the first kind of uncertainties using probability theory and will not explicitly model uncertainties in model and parameters.

Bayesian methods: modelling

Let assume that we have observed some data $\mathcal{D}$.

This data can be for example a set of points from IR^p
$\mathcal{D} = \{x_1, x_2, \ldots, x_N\}$ (but it does not need to be a set).

1) We construct a model m that can generate/predict the data. This model has a number of free parameters, denoted collectively by θ.

Using this model, one can compute the probability of the data for any set of parameters, ie.:

$$P(\mathcal{D}|\theta, m).$$

This is called the **likelihood** of the parameters.

Example:
In the context of supervised learning, m_1 could be a linear model $y = w_0 + w_1 x + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and its parameters $\theta_1 = (w_0, w_1, \sigma^2)$ and m_2 could be a degree-2 polynomial model $y = w_0 + w_1 x + w_2 x^2 + \epsilon$ with $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and its parameters $\theta_2 = (w_0, w_1, w_2, \sigma^2)$.

Bayesian methods: modelling

2) We specify a probability distribution for these unknown parameters that expresses our beliefs about which values are more or less likely, **before seeing the data**:

$$P(\theta|m).$$

This distribution is the called the **prior**.

Example: the coefficient of the quadratic term in m_2 should be small: $w_2 \sim \mathcal{N}(0, s^2)$, with s^2 small.

NB: Probabilities can not be interpreted here as limit of relative frequencies of event in a large number of trials but instead are used to express our belief or our uncertainty about what the value of a given parameter should be.

Bayesian methods: inference

3) From the likelihood, the prior, and the data $\mathcal{D}$, we compute the **posterior distribution** $P(\theta|\mathcal{D}, m)$ using Bayes' theorem:

$$P(\theta|\mathcal{D}, m) = \frac{P(\mathcal{D}|\theta, m)P(\theta|m)}{P(\mathcal{D}|m)}$$

Bayesian learning transforms prior knowledge $P(\theta|m)$ about the parameters into posterior knowledge $P(\theta|\mathcal{D}, m)$, by exploiting the data.

The posterior expresses our belief/uncertainty in the parameters **once we have seen the data**. One of the most distinctive features of Bayesian methods is to keep the full posterior distribution instead of extracting a point estimate from it.

$P(\mathcal{D}|m)$ acts as a normalizing constant and is called the **marginal likelihood**. It can be computed as:

$$P(\mathcal{D}|m) = \int P(\mathcal{D}|\theta, m)P(\theta|m)d\theta.$$

Bayesian methods: inference

4) We use this posterior distribution to:

- Make predictions by averaging over the posterior
- Examine/Account for uncertainty in the parameter values
- Make decisions by minimizing expected posterior loss

Predictions of new data points x_{new} are obtained from the **predictive distribution**

$$p(x_{new}|\mathcal{D}, m) = \int p(x_{new}|\theta, \mathcal{D}, m)p(\theta|\mathcal{D}, m)d\theta.$$

computed by integrating over the posterior distribution.

The predictive distribution translates uncertainties in the model parameters into **uncertainties in the predictions**.

Note that this is very different from frequentists' uncertainty, which is uncertainty due to randomness in the data. For a Bayesian, the data is fixed.

Two main challenges

There are two main challenges associated with the application of Bayesian approaches:

Modelling challenge: how to specify suitable models and prior distributions:

- A suitable model should admit all the possibilities that are thought to be at all likely.
- A suitable prior should avoid giving zero or very small probabilities to possible events, but should also avoid spreading out the probability over all possibilities.
- If the model/prior are chosen without regard for the actual situation, there is no justification for believing the results of Bayesian inference.

Computational challenge: how to compute the posterior distribution?

- Computing the marginal likelihood $P(\mathcal{D}|m)$ is indeed often intractable.
- Fortunately, several approaches exist to solve this issue.

Maximum Likelihood and Maximum a Posteriori

Two other related popular **non-bayesian** kinds of learning based on point estimates:

- **Maximum likelihood** (ML) learning: find the parameters $\hat{\theta}_{ML}$ that maximize the likelihood:

$$\hat{\theta}_{ML} = \arg\max_{\theta} P(\mathcal{D}|\theta, m)$$

- **Maximum a posteriori** (MAP) learning: find the parameters $\hat{\theta}_{MAP}$ that maximizes the posterior:

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(\theta|\mathcal{D}, m)$$

Both coincide when the prior $p(\theta|m)$ is uniform.

NB: Although it exploits the priors, computing the MAP estimate is not considered as a full Bayesian approach.

Illustration

We would like to learn about the average height of Belgian male adult.
Our data could be a sample of measured heights of Belgian men
$\mathcal{D} = \{y_1, y_2, \ldots, y_n\}$.

The likelihood:
Model m: Each measurement is drawn from a Gaussian distribution of mean μ and variance σ^2, where σ is supposed to be known (to simplify things).
Parameters θ: μ.

$$p(\mathcal{D}|\theta, m) = p(\{y_1, y_2, \ldots, y_N\}|\mu, m) = \prod_{i=1}^{N} \mathcal{N}(y_i|\mu, \sigma^2).$$

The prior:
The average height should be very close to $\mu_o = 175cm$. This can be expressed by a Gaussian prior on μ of mean μ_0 and variance s^2:

$$p(\theta|m) = p(\mu|m) = \mathcal{N}(\mu|\mu_0, s^2),$$

Illustration

The posterior:

$$\begin{aligned} p(\mu|\mathcal{D}, m) &\propto p(\mathcal{D}|\mu, m)p(\mu|m) \\ &\propto \prod_{i=1}^{N} \mathcal{N}(y_i|\mu, \sigma^2)\mathcal{N}(\mu|\mu_0, s^2) \end{aligned}$$

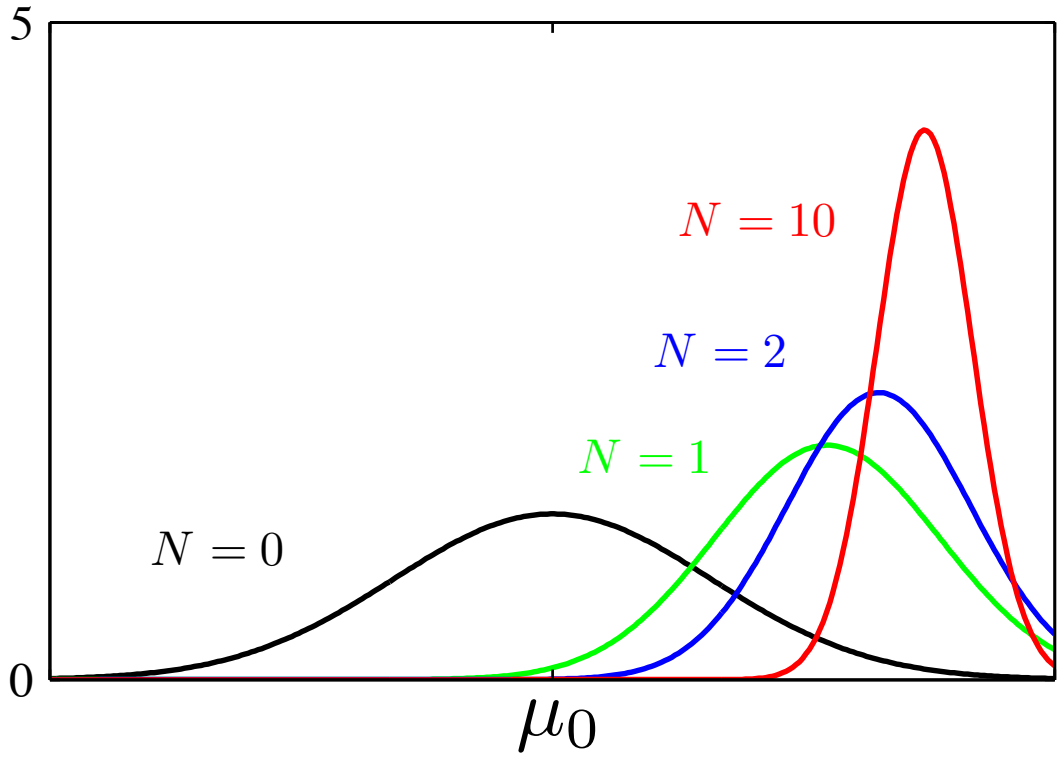
Since the product of two Gaussian densities (on μ here) is a Gaussian, we can show that:

$$p(\mu|\mathcal{D}, m) = \mathcal{N}(\mu|\mu_N, \sigma_N^2),$$

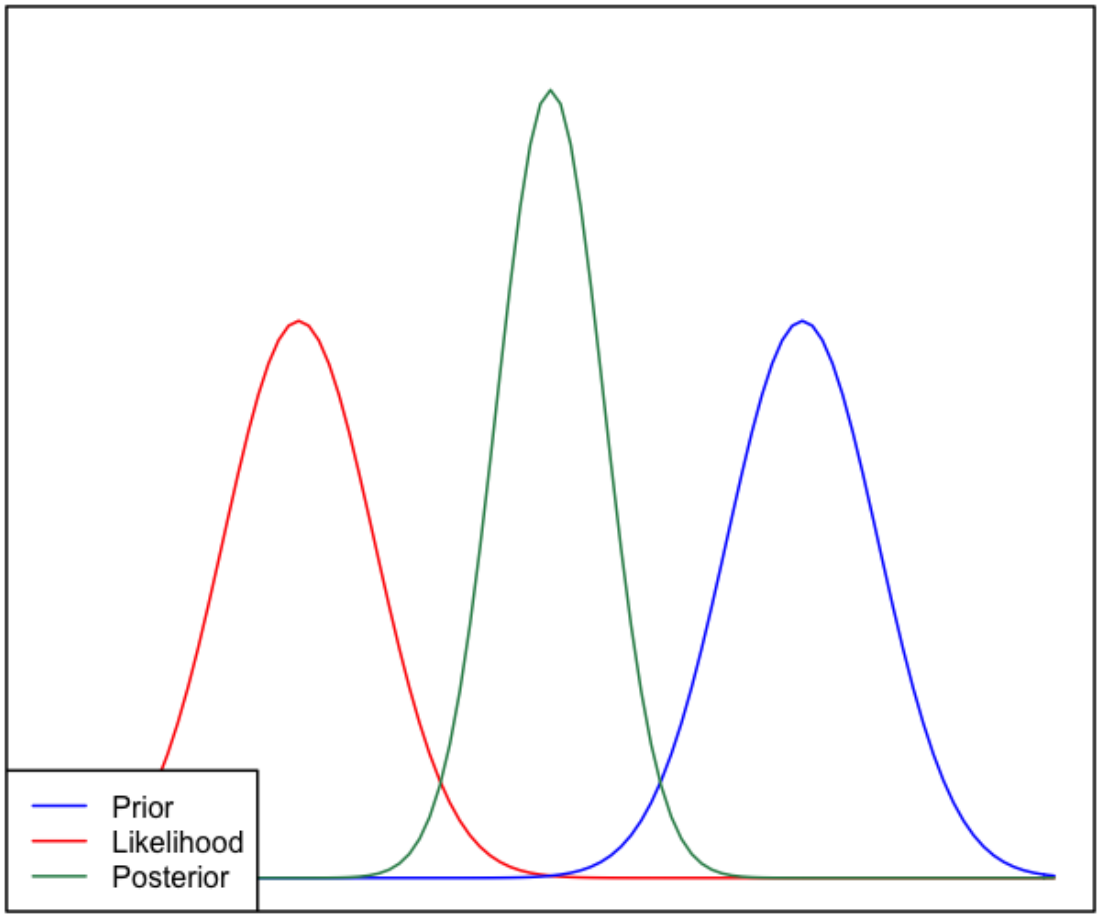
with $\sigma_N^2 = (\frac{1}{s^2} + \frac{N}{\sigma^2})^{-1}$, $\mu_N = \sigma_N^2(\frac{\mu_0}{s^2} + \frac{N\bar{y}}{\sigma^2})$, and $\bar{y} = \frac{1}{N}\sum_i y_i$.

- As $N \to +\infty$, μ_N converges to the MLE estimate $\bar{y}$, independently on the prior. Our uncertainty about μ_N also decreases to 0 with N.
- If variance of the prior s^2 increases to infinity, then μ_N also converges to MLE.

Illustration



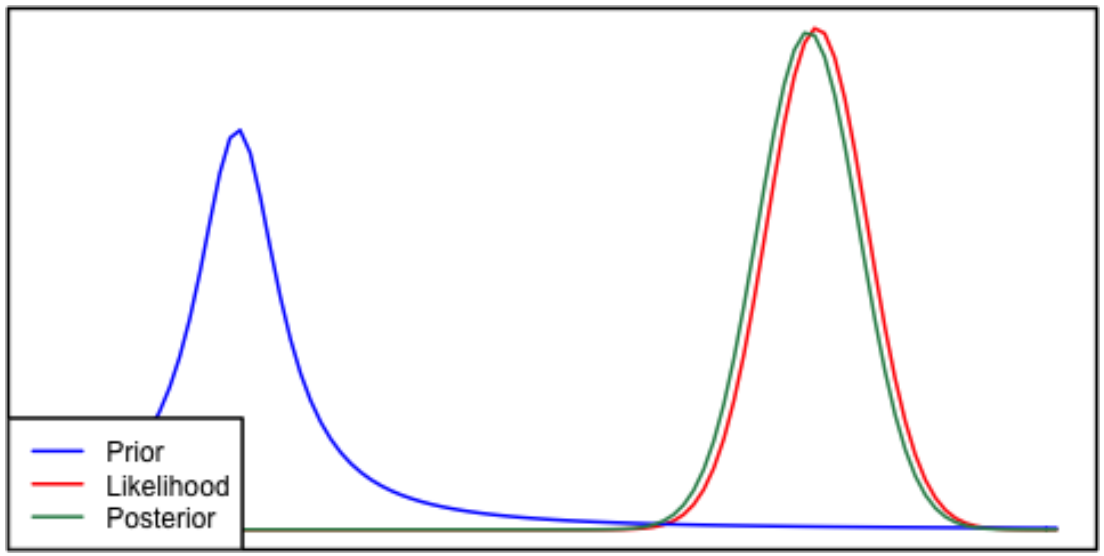
Illustration



A Bayesian is one who, vaguely expecting a horse, and catching a glimpse of a donkey, strongly believes he has seen a mule.

Karl Pearson?

Illustration



Using a prior with a longer tail (here, a Cauchy distribution).

A note on conjugate prior

If the posterior distribution $p(\theta|\mathcal{D}, m)$ is in the same family as the prior distribution $p(\theta|m)$, the prior and posterior are called **conjugate distributions**, and the prior is called a **conjugate prior** for the likelihood.

This is very convenient computationally as in such case, the posterior can be computed analytically (without worrying about computing the marginal likelihood).

Examples of conjugate (prior-likelihood) pairs:

Prior	Likelihood
Gaussian	Gaussian
Beta	Binomial
Gamma	Gaussian
Dirichlet	Multinomial

Outline

Bayesian approaches for supervised learning

In supervised learning (SL), we observe a set of pairs $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, N\}$, with $x_i \in \mathcal{R}^p$ and $y_i \in \mathcal{R}$ and we are interested in predicting y for a new x.

In what follows, we will denote the input matrix $X = [x_1, \ldots, x_N]^T \in \mathcal{R}^{N \times p}$ and the output vector $y = [y_1, \ldots, y_N]^T \in \mathcal{R}^N$, such that $\mathcal{D} = \{X, y\}$.

In SL, we are only interested in modelling and predicting the outputs. The X part of the data will always belong to the conditionings, assuming inputs are fixed.

Bayes' theorem applied to compute the **posterior** will thus take the following form:

$$p(\theta|X, y, m) = \frac{p(y|X, \theta, m)P(\theta|m)}{P(y|X, m)}.$$

(X does not appear in the conditioning of the prior $P(\theta|m)$ as it is not supposed to be used to define this prior)

Bayesian linear regression: likelihood

Linear regression assumes that there is a linear relationship between inputs and output, up to some residuals $\epsilon \in \mathcal{R}^N$:

$$y = Xw + \epsilon,$$

where $w \in \mathcal{R}^p$ are free parameters. the residuals are assumed to be independent, unbiased, and small. This can be modeled with:

$$p(\epsilon|\sigma^2) = \mathcal{N}(\epsilon|0, \sigma^2 I_N),$$

with I_N the identity matrix of size $N \times N$. We will assume that we know the true value of σ^2. The parameters of the model, θ, thus reduce in this case to the vector w.

The **likelihood** is thus written

$$p(y|X, w, \sigma^2) = \mathcal{N}(y|Xw, \sigma^2 I_N)$$

Bayesian linear regression: prior and posterior

A priori, we will assume that entries of w are small and that all directions for w are equally likely. This can be encoded by the following **prior** distribution:

$$p(w|s^2) = \mathcal{N}(w|0, s^2 I_p).$$

We want to compute the **posterior** distribution:

$$p(w|X, y, \sigma^2, s^2) \propto p(y|X, w, \sigma^2) p(w|s^2)$$

This can be done analytically in this specific case. Indeed, since we use a conjugate prior, the posterior will be Gaussian:

$$p(w|X, y, \sigma^2, s^2) = \mathcal{N}(w|\mu_w, \Sigma_w),$$

and μ_w and Σ_w can be computed by identification.

Bayesian linear regression: posterior

From conjugacy:

$$\begin{aligned} p(w|X, y, \sigma^2, s^2) &\propto \exp\left(-\frac{1}{2}(w-\mu_w)^T\Sigma_w^{-1}(w-\mu_w)\right) \\ &\propto \exp\left(-\frac{1}{2}(w^T\Sigma_w^{-1}w - 2\mu_w^T\Sigma_w^{-1}w)\right) \end{aligned}$$

From prior and likelihood definitions:

$$\begin{aligned} p(w|X, y, \sigma^2, s^2) &\propto \mathcal{N}(y|Xw, \sigma^2 I_N)\mathcal{N}(w|0, s^2 I_p) \\ &\propto \exp\left(-\frac{1}{2}(\frac{1}{\sigma^2}(y-Xw)^T(y-Xw) + \frac{1}{s^2}w^Tw)\right) \\ &\propto \exp\left(-\frac{1}{2}(w^T(\frac{1}{\sigma^2}X^TX + \frac{1}{s^2}I_p)w - \frac{2}{\sigma^2}y^TXw)\right) \end{aligned}$$

By identification:

$$\Sigma_w = \left(\frac{1}{\sigma^2}X^TX + \frac{1}{s^2}I_p\right)^{-1},\ \mu_w = \left(\frac{1}{\sigma^2}X^TX + \frac{1}{s^2}I_p\right)^{-1}\frac{1}{\sigma^2}X^Ty$$

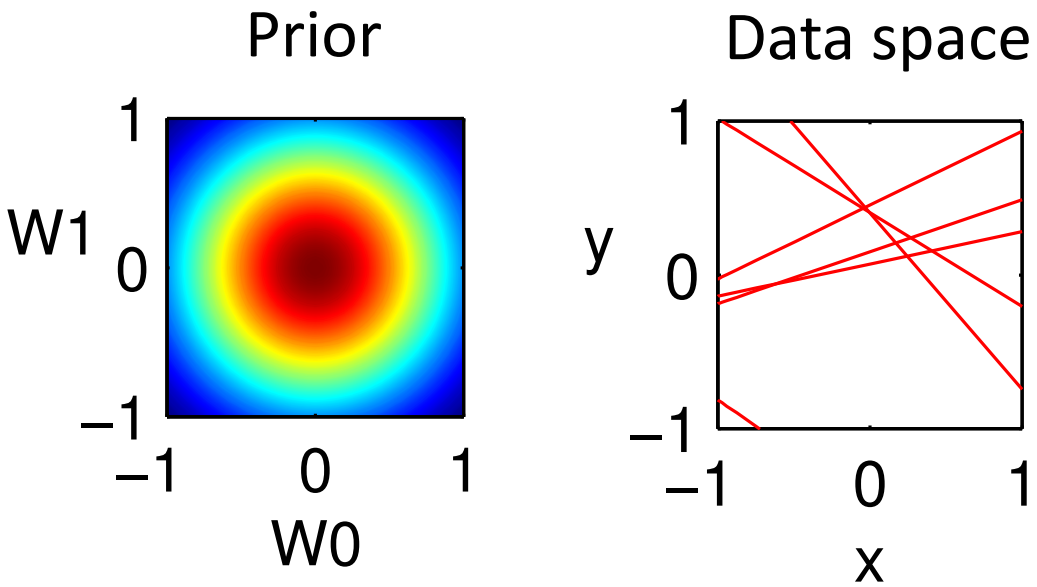
What happens when $N \to +\infty$? When $s^2 \to +\infty$?

Bayesian linear regression: posterior illustration

Let us consider BLR with a model of the form: $y(x, w) = w_0 + w_1 x$.

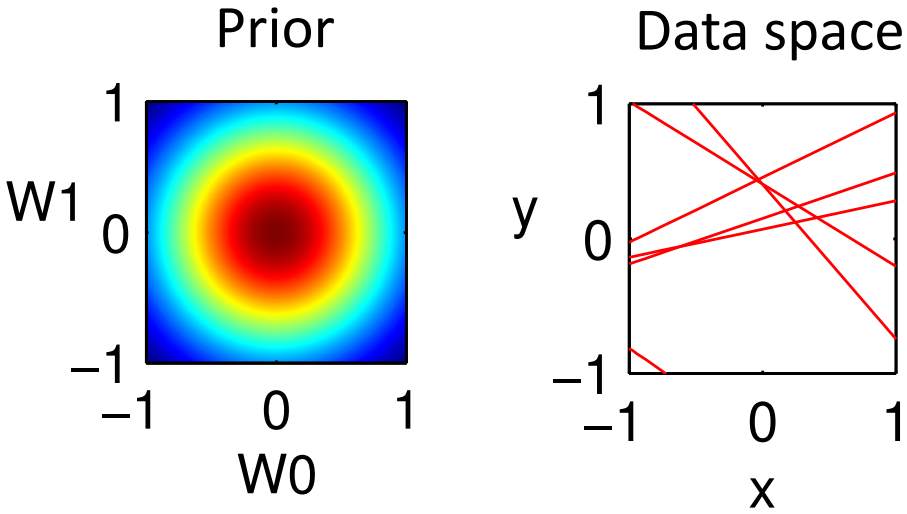
The training data is generated from the function $y(x) = -0.3 + 0.5x$ by first choosing x uniformly from $[-1, 1]$, evaluating $y(x)$, and adding a small Gaussian noise ($\sigma = 0.2$).

When zero data points have been observed:

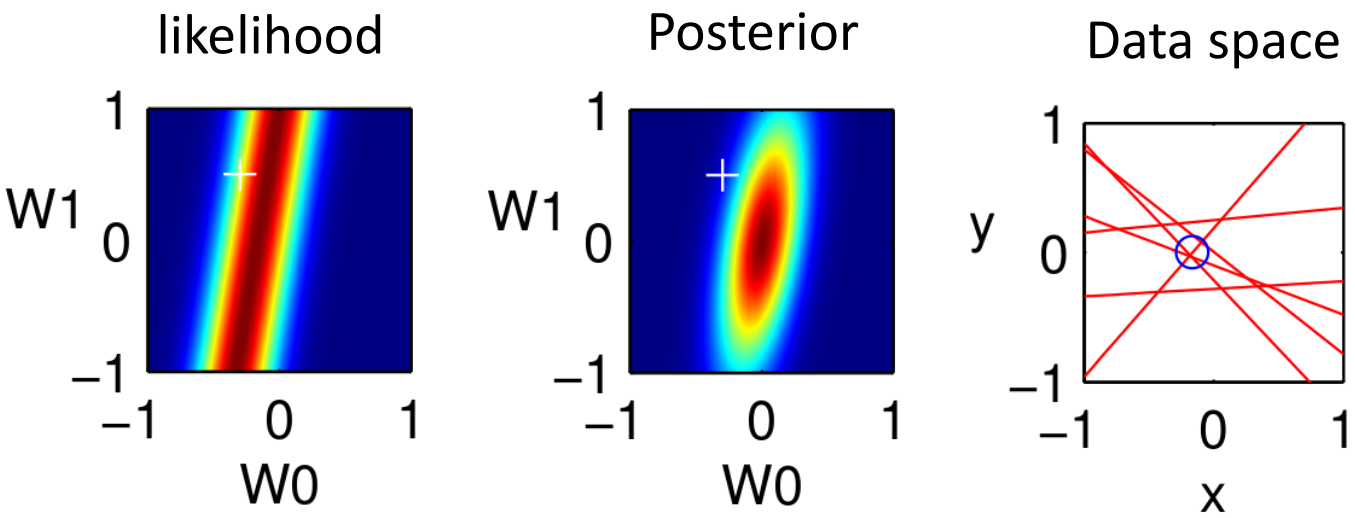


Bayesian linear regression: posterior illustration

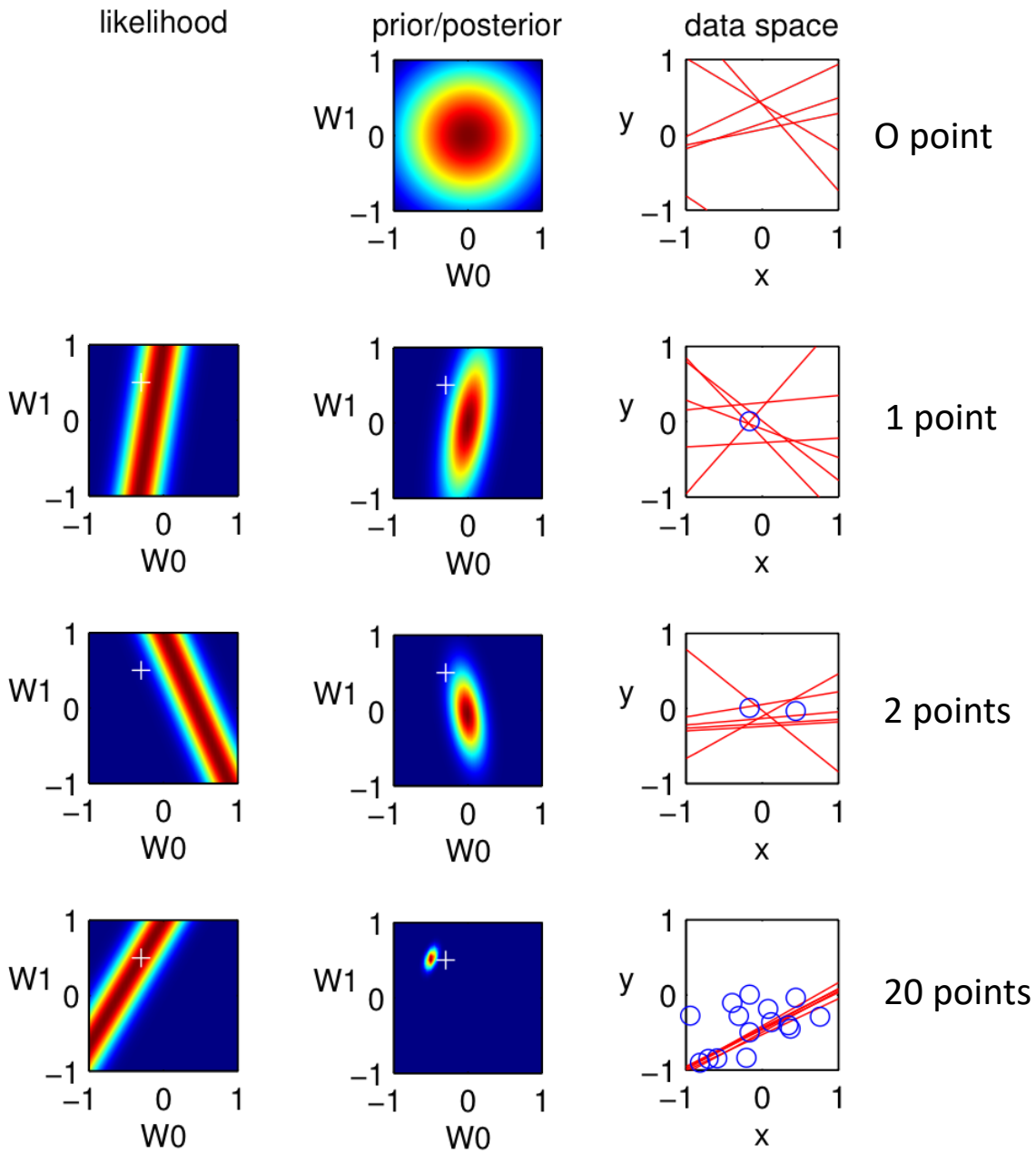
0 data points are observed:



1 data point is observed:



Bayesian linear regression: posterior illustration



(Murphy, 2012)

Bayesian linear regression: predictive distribution

Given a new observation x_{new}, we are interested in the density:

$$p(y_{new}|x_{new}, X, y, \sigma^2, s^2) = \int p(y_{new}|x_{new}, w, \sigma^2)p(w|X, y, \sigma^2, s^2)dw$$

Given the definition of our model, we have $y_{new} = x_{new}^T w + \epsilon$. Given that $w \sim \mathcal{N}(\mu_w, \Sigma_w)$, we have $x_{new}^T w \sim \mathcal{N}(x_{new}^T \mu_w, x_{new}^T \Sigma_w x_{new})$. Since in addition, $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is independent of w, we have finally:

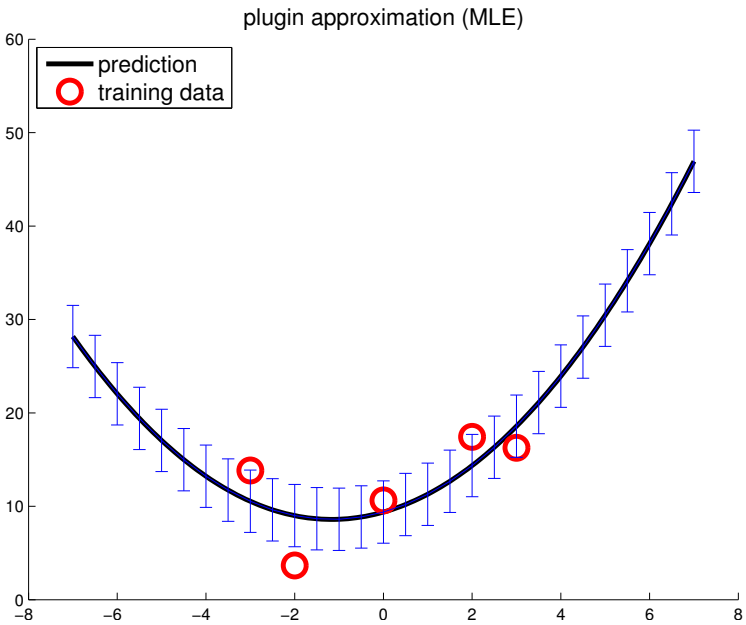
$$p(y_{new}|x_{new}, X, y, \sigma^2, s^2) = \mathcal{N}(y_{new}|x_{new}^T \mu_w, \sigma^2 + x_{new}^T \Sigma_w x_{new})$$

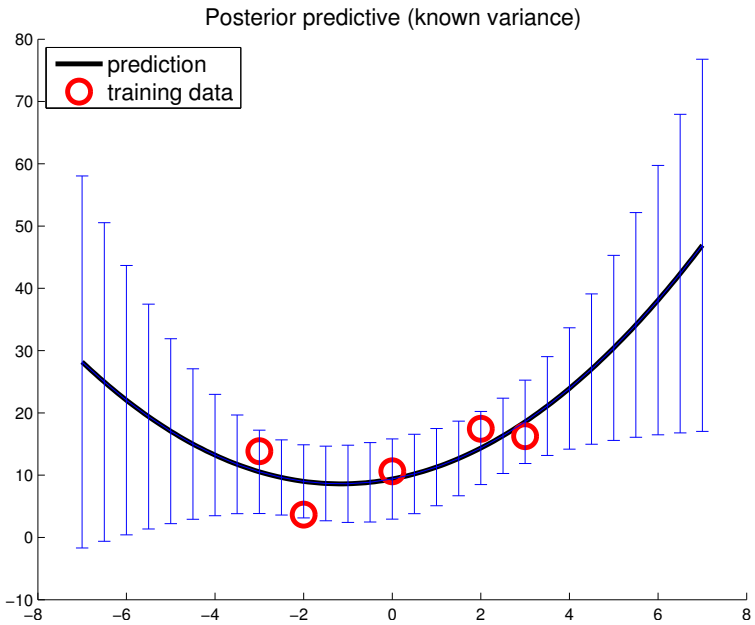
A point estimate can be obtained by minimizing a given loss function L:

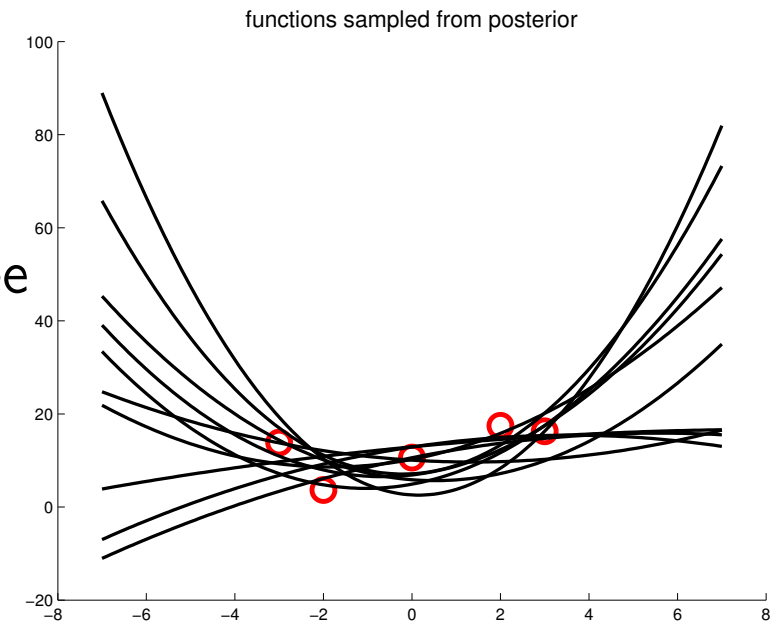
$$\arg\min_{y'} \int L(y', y_{new})p(y_{new}|x_{new}, X, y, \sigma^2, s^2)dy_{new}$$

If $L(y', y_{new}) = (y' - y_{new})^2$, the optimal prediction is the mean of the distribution $x_{new}^T \mu_w$.

Bayesian linear regression: predictive distribution







Using MLE and BLR to fit a degree 2 polynomial in one dimension

(Rogers and Girolami, 2012)

Link with ridge regression

We have found that the posterior mean (which is also the MAP estimate) is given by:

$$\mu_w = \left(\frac{1}{\sigma^2}X^T X + \frac{1}{s^2}I_p\right)^{-1}\frac{1}{\sigma^2}X^T y = \left(X^T X + \frac{\sigma^2}{s^2}I_p\right)^{-1}X^T y,$$

which is exactly the ridge regression solution when $\lambda = \frac{\sigma^2}{s^2}$.

This is not surprising as the MAP estimate minimizes exactly the ridge regression objective function.
Indeed, we have

$$\hat{w}_{MAP} = \arg\max_w p(w|X, y, \sigma^2, s^2) = \arg\max_w p(y|X, w, \sigma^2)p(w|s^2)$$

Since

$$p(y|X, w, \sigma^2) = \mathcal{N}(y|Xw, \sigma^2 I_n) = \prod_{i=1}^{N}\mathcal{N}(y_i|x_i^T w, \sigma^2),$$

one can show that (by maximising the log of the posterior):

$$\hat{w}_{MAP} \quad = \quad \arg\max_w -\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i^T w - y_i)^2 - \frac{1}{2s^2}\sum_{i=1}^{p} w_i^2$$

Link with ridge regression

...and thus:

$$\hat{w}_{MAP} = \arg\min_{w} \sum_{i=1}^{N} (x_i^T w - y_i)^2 + \frac{\sigma^2}{s^2} ||w||_2^2.$$

Squared error in the objective function comes from the hypothesis of an iid **Gaussian noise** assumption and ℓ^2 **penalisation** comes from a **Gaussian prior** on w.

One can obtain different objective functions by changing the noise model and/or the prior.

For example, one can retrieve LASSO by using Laplace priors:

$$P(w|\lambda) = \prod_{j=1}^{p} \text{Lap}(w_j|0, 1/\lambda), \text{ with } \text{Lap}(x|\mu, b) = \frac{1}{2b} \exp(-\frac{|x-\mu|}{b})$$

(Posteriors can not be computed analytically anymore however)

Discussion

When using conjugate prior, it is possible to compute everything in closed-form, which makes the approach very efficient.

To a specific choice of model and prior corresponds a specific objective function for the MAP. Bayesian modeling is a way to design principled loss function and regularization terms.

With respect to the MAP, full Bayesian predictions are more uncertain for x values that are far away from training examples (as expected).

Outline

Bayesian logistic regression: likelihood

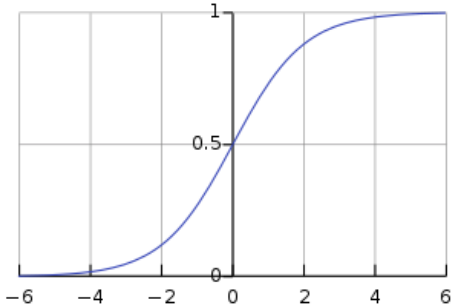
We observe a set of pairs $\mathcal{D} = \{(x_i, y_i)|i = 1, \ldots, N\}$, with $x_i \in \mathcal{R}^p$ and $y_i \in \{0, 1\}$ a binary classification response.

We assume that the class-conditional probability of belonging to the "1" class is given by a nonlinear transformation of a linear function of x:

$$P(y = 1|x, w) = \sigma(x^T w).$$

The most-commonly used function σ is the logistic function:

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$



If we assume that predictions for different x are independent given w, then the **likelihood** can be written:

$$p(y|X, w) = \prod_{i=1}^{N} \left(\frac{1}{1 + \exp(-x_i^t w)} \right)^{y_i} \left(\frac{\exp(-x_i^t w)}{1 + \exp(-x_i^t w)} \right)^{1-y_i}$$

Bayesian logistic regression: prior and posterior

It seems reasonable to use the same **prior** as for linear regression, ie.:

$$p(w|s^2) = \mathcal{N}(w|0, s^2 I_p).$$

We want to compute the **posterior** distribution from the prior and the likelihood:

$$p(w|X, y, s^2) = \frac{p(y|X, w)p(w|s^2)}{\int p(y|X, w)p(w|s^2)dw} = \frac{p(y|X, w)p(w|s^2)}{p(y|X, s^2)},$$

so as to predict the response for new inputs x_{new}:

$$p(y_{new} = 1|x_{new}, X, y, s^2) = E_{p(w|X,y,s^2)}\left\{\frac{1}{1 + \exp(-x_{new}^T w)}\right\}$$

Unfortunately, the posterior distribution does not belong to a nice parametric family and the integral $p(y|X, s^2)$ is intractable as well.

How to solve intractability of posterior computation?

Three main approaches:

1. Get a **point estimate** of w such as the MAP or the MLE
2. **Approximate** $p(w|X, y, s^2)$ by some (simpler) density
3. Obtain **samples** from the posterior $p(w|X, y, s^2)$

A point estimate: the MAP solution

The MAP estimate is defined as:

$$\begin{aligned}\hat{w}_{MAP} &= \arg\max_w p(w|X, y, s^2) = \arg\max_w p(y|X, w)p(w|s^2) \\ &= \arg\min_w \sum_{i=1}^{N} -y_n \log(\sigma(x_i^T w)) - (1 - y_n) \log(1 - \sigma(x_i^T w)) + \frac{1}{2s^2}||w||_2^2\end{aligned}$$

NB: our noise model leads to cross-entropy as the loss function.

Unlike in linear regression, $\hat{w}_{MAP}$ can not be computed analytically.

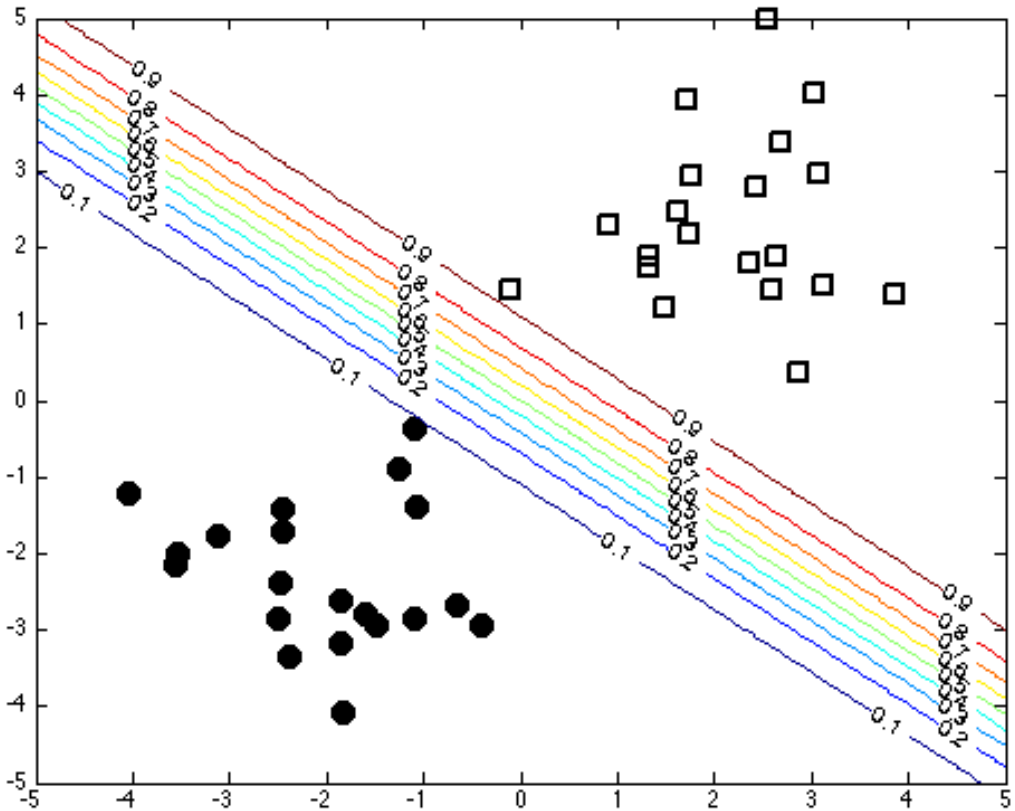
Optimization methods can however be used to compute it, e.g. stochastic gradient descent.

It can be shown that the objective function is **convex** and there is thus a unique global minimum.

Main disadvantage: we do not maintain a density over w, which is the main interest of Bayesian methods.

A point estimate: the MAP solution

$$p(y_{new} = 1 | x_{new}, \hat{w}_{MAP}) = \sigma(x_{new}^T \hat{w}_{MAP})$$



(Rogers and Girolami, 2012)

Laplace approximation

The idea of Laplace approximation is to approximate the posterior density with a Gaussian:

$$p(\theta|\mathcal{D}, m) \approx \mathcal{N}(\theta|\mu, \Sigma)$$

The Laplace approximation is based on a Taylor expansion of the negative log of the unnormalized posterior:

$$\Psi(\theta) = -\log p(\mathcal{D}|\theta, m) - \log p(\theta|m),$$

around its maximum $\hat{\theta}_{MAP} = \arg\max_\theta p(\theta|\mathcal{D}, m)$.

The second-order Taylor expansion of $\Psi(\theta)$ at $\hat{\theta}$ is given by:

$$\Psi(\theta) \approx \Psi(\hat{\theta}) + (\theta - \hat{\theta})^T g + \frac{1}{2}(\theta - \hat{\theta})^T H(\theta - \hat{\theta}),$$

where g and H are resp. the gradient and the Hessian of $\Psi(\theta)$ evaluated at $\hat{\theta}$:

$$g = \nabla\Psi(\theta)|_{\theta=\hat{\theta}}\,,\; H = \nabla\nabla\Psi(\theta)|_{\theta=\hat{\theta}}\,.$$

Laplace approximation

Since the gradient is zero at $\hat{\theta}_{MAP}$, we have:

$$\Psi(\theta) \approx \Psi(\hat{\theta}_{MAP}) + \frac{1}{2}(\theta - \hat{\theta}_{MAP})^T H(\theta - \hat{\theta}_{MAP}),$$

Negating and exponentiating both sides, one gets:

$$p(\theta|\mathcal{D}, m) \propto \exp(-\Psi(\hat{\theta}_{MAP})) \exp\left(-\frac{1}{2}(\theta - \hat{\theta}_{MAP})^T H(\theta - \hat{\theta}_{MAP})\right),$$

which is proportional to a Gaussian distribution:

$$p(\theta|\mathcal{D}, m) \approx \mathcal{N}(\theta|\hat{\theta}_{MAP}, H^{-1}).$$

Note that this approximation makes the computation of the marginal likelihood $P(\mathcal{D}|m)$ possible using the known normalization of the Gaussian.

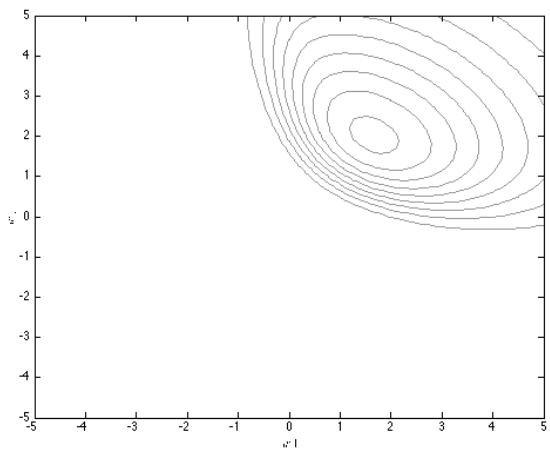
Laplace approximation for logistic regression: posterior

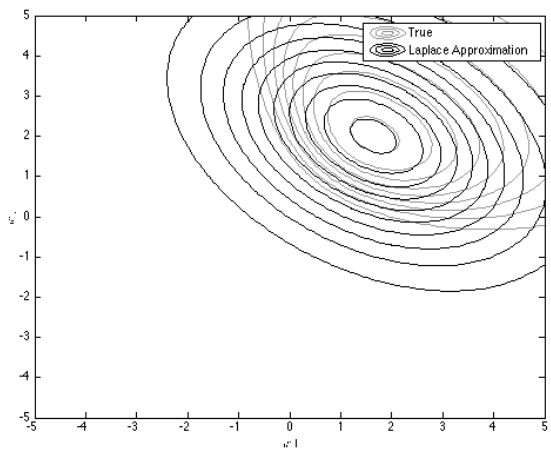
In the case of logistic regression, one can thus approximate the posterior as follows:

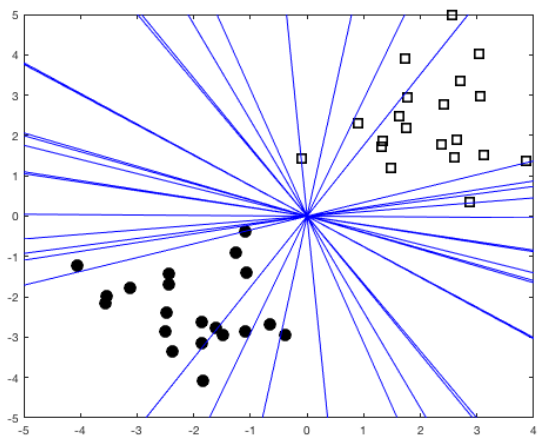
$$p(w|X, y, s^2) \approx \mathcal{N}(w|\hat{w}_{MAP}, H^{-1}),$$

where $\hat{w}_{MAP}$ can be obtained by optimization and H is given by:

$$H = \frac{1}{s^2} I_p + \sum_{i=1}^{N} x_i x_i^T P_i(1 - P_i), \text{ with } P_i = \sigma(x_i^T \hat{w}_{MAP})$$







(Rogers and Girolami, 2012)

Laplace approximation for logistic regression: prediction

To make a prediction, one needs the predictive distribution:

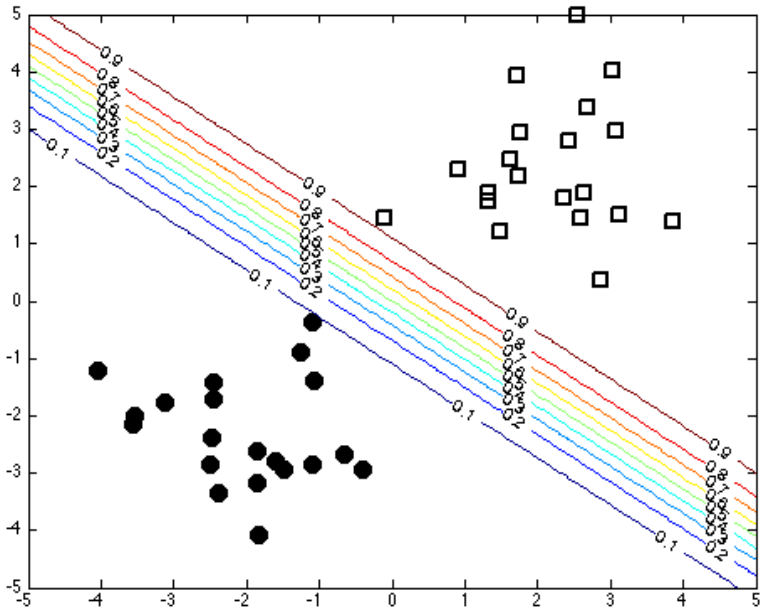
$$\begin{aligned} P(y_{new} = 1|x_{new}, X, y, s^2) &= \int P(y_{new} = 1|x_{new}, w)p(w|X, y, s^2)dw \\ &= E_{p(w|X,y,s^2)}\{P(y_{new} = 1|x_{new}, w)\}. \end{aligned}$$

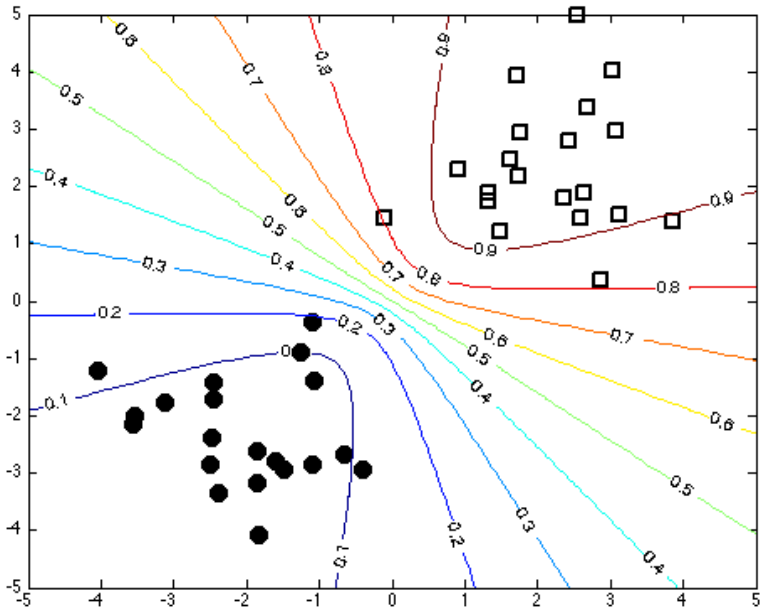
Unfortunately, this integral can not be computed analytically even with the Gaussian approximation of the posterior. A numerical approximation can however be easily obtained by Monte-Carlo sampling:

$$P(y_{new} = 1|x_{new}, X, y, s^2) \approx \frac{1}{S}\sum_{i=1}^{S} \sigma(x_{new}^T w^s),$$

where w^s are independently sampled from $\mathcal{N}(w|\hat{w}_{MAP}, H^{-1})$.

Laplace approximation for logistic regression: prediction





(Rogers and Girolami, 2012)

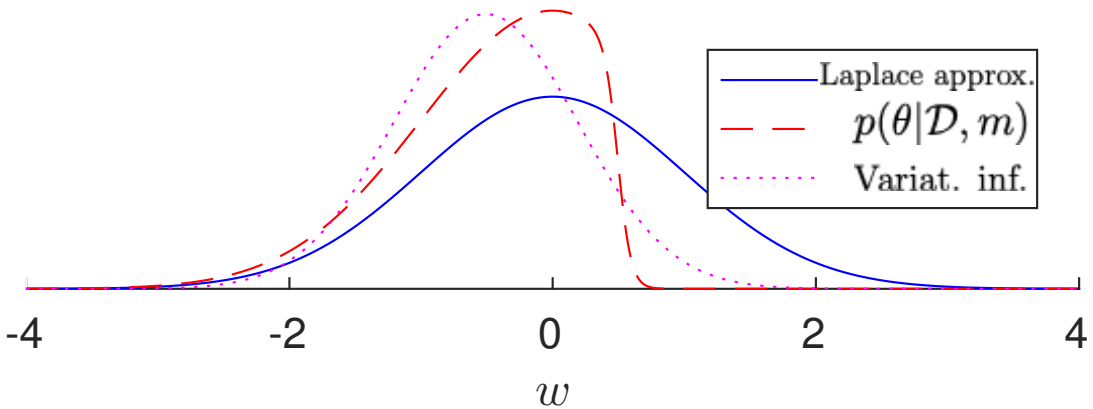
Predictive probability contours are now curved and better reflect expected uncertainty.

Laplace approximation: limitations

Laplace approximation will (obviously) fail when the posterior is not well approximated by a Gaussian.

It also does not necessarily produce the optimal Gaussian approximation of the posterior.

Example:



(Murray)

There are ways to obtain better approximation of the posterior, e.g., **variational inference**.

Variational inference

Main idea: approximate the posterior on model parameters $p(\theta|\mathcal{D}, m)$ with a simpler distribution $q(\theta)$.

One needs to define:

- a **family** $\mathcal{Q}$ of candidate distributions q
- a **cost function**, which measures the discrepancy between a candidate distribution $q \in \mathcal{Q}$ and the true posterior
- an **optimisation strategy** to find $q^* \in \mathcal{Q}$ that minimises the cost function.

Candidate distributions in $\mathcal{Q}$ are typically defined through a parametric form $q(\theta; \alpha)$, with **variational parameters** α that will be tuned to minimise the cost.

E.g., in the case of bayesian logistic regression, $\theta = w$ and $\mathcal{Q}$ could the set of Gaussian distributions $\mathcal{N}(w|\mu_w, \Sigma_w)$ with the mean μ_w and covariance matrix Σ_w as the variational α parameters.

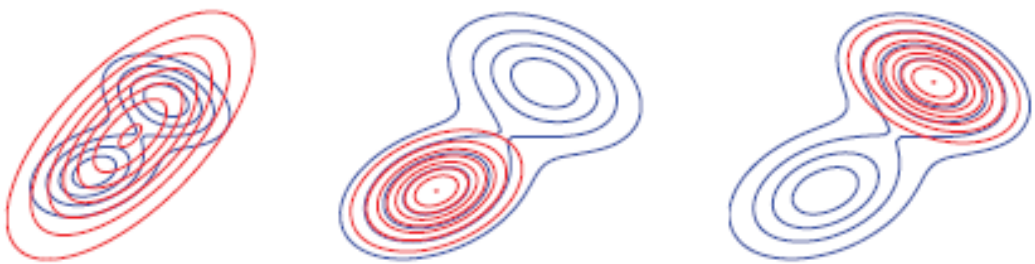
Kullback Leibler (KL-)divergence

One common measure of the discrepancy between two distributions p_1 and p_2 is the **Kullback-Leibler divergence**:

$$D_{KL}(p_1||p_2) \triangleq \int p_1(x) \log \frac{p_1(x)}{p_2(x)} dx$$

KL-divergence is **non-negative** and is only zero when the two distributions are identical (see ELEN0060).

KL-divergence is not symmetric $\Rightarrow$ in the context of variational inference, one could minimize either $D_{KL}(p(\theta|\mathcal{D}, m)||q(\theta; \alpha))$ or $D_{KL}(q(\theta; \alpha)||p(\theta|\mathcal{D}, m))$, leading to different approximations.



(Murphy)

It is usually easier to minimize $D_{KL}(q(\theta; \alpha)||p(\theta|\mathcal{D}))$ (reverse KL).

Variational methods with KL-divergence

Minimising $D_{KL}(q(\theta;\alpha)||p(\theta|\mathcal{D},m))$ directly might still not be tractable as evaluating $p(\theta|\mathcal{D},m)$ requires to compute model evidence $p(\mathcal{D}|m)$.

However, using Bayes'rule, one can rewrite D_{KL} as follows:

$$D_{KL}(q(\theta;\alpha)||p(\theta|\mathcal{D},m)) = -L(q(\theta;\alpha)) - \log p(\mathcal{D}|m)$$

with

$$L(q(\theta;\alpha)) = \int q(\theta;\alpha)\log(p(\mathcal{D}|\theta,m))d\theta - D_{KL}(q(\theta;\alpha)||p(\theta|m)).$$

Since $P(\mathcal{D}|m)$ does not depend on q, *minimising* $D_{KL}(q(\theta;\alpha)||p(\theta|\mathcal{D},m))$ amounts at *maximising* $L(q(\theta;\alpha))$.

Because $D_{KL} \geq 0$, we have that $\log p(\mathcal{D}|m) \geq L(q(\theta;\alpha))$. $L(q(\theta;\alpha))$ is therefore called the **evidence lower bound (ELBO)**.

Evidence lower bound

We want to maximise ELBO (over α):

$$L(q(\theta;\alpha)) = \int q(\theta;\alpha)\log(p(\mathcal{D}|\theta,m))d\theta - D_{KL}(q(\theta;\alpha)||p(\theta|m)).$$

Interpretation:

- The first term is the expectation (over q) of the log likelihood. It measures how well samples of parameters from $q(\theta;\alpha)$ explain the data.
- The second term is KL-divergence between q and the prior $p(\theta|m)$. It penalises complexity (and thus avoids overfitting when choosing α).
- By maximising $L(q(\theta;\alpha))$, we also maximise $\log P(\mathcal{D}|m)$. The two will be equal only when $q(\theta;\alpha) = p(\theta|\mathcal{D},m)$.

Maximising the ELBO

We want to find α that maximises ELBO:

$$L(q(\theta;\alpha)) = \int q(\theta;\alpha)\log(p(\mathcal{D}|\theta,m))d\theta - D_{KL}(q(\theta;\alpha)||p(\theta|m)).$$

The second term can be often computed analytically but the first one is harder to evaluate.

Several clever techniques exist to solve this optimisation problem. The most common ones are based on **stochastic Monte-Carlo integrations**:

Repeat:

1. Sample $\hat{\theta} \sim q(\theta;\alpha)$
2. Do one step of maximisation with respect to α of:

$$\hat{L}(q(\theta;\alpha)) = \log(p(\mathcal{D}|\theta,m)) - D_{KL}(q(\theta;\alpha)||p(\theta|m))$$

As $\hat{L}$ is an unbiased approximation of L, this algorithm will maximise L.

Sampling techniques

Laplace approximation and variational inference first approximate the posterior and then use sampling to make predictions.

There exist general techniques to sample directly from the posterior without requiring to approximate it first.

We will explain and illustrate, without proof, one of these techniques, **Metropolis-Hastings**, in the case of logistic regression.

Links to other techniques will be given later.

Metropolis-Hastings algorithm

Input: a target distribution $p(x)$ from which we would like to sample and a proposal distribution $q(x'|x)$.
Output: a sequence of random samples $\{x_0, x_1, x_2, \ldots\}$.

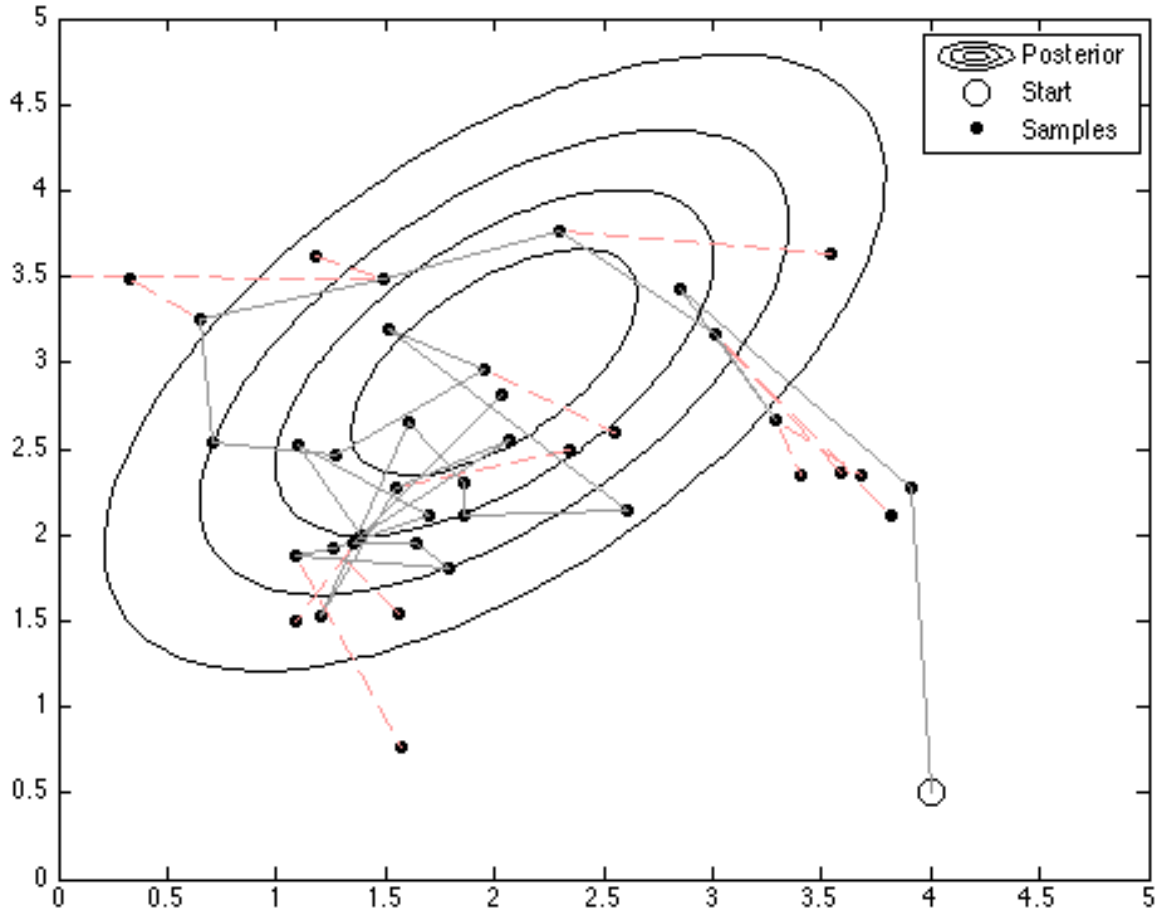
- Initialize x^0
- for $s = 0, 1, 2, \ldots$:
  1. Sample $x' \sim q(x'|x^s)$
  2. Compute acceptance probability

$$\alpha = min\left(1, \frac{p(x')q(x^s|x')}{p(x^s)q(x'|x^s)}\right)$$

  3. Sample $u \sim U(0, 1)$
  4. Set new sample to

$$x^{s+1} = \begin{cases} x' & \text{if } u < \alpha \\ x^s & \text{if } u \geq \alpha \end{cases}$$

Metropolis-Hastings algorithm: illustration



(Rogers and Girolami, 2012)

Metropolis-Hastings algorithm: some comments

- MH algorithm is an instance of Markov Chain Monte Carlo (MCMC) algorithms. It defines a Markov chain (new state x^{s+1} only depends on previous state x^s), which stationary distribution coincides with $p(x)$.
- One should wait before collecting, to let the chain reach convergence (waiting time is called burn-in period). There are tools to diagnose whether convergence has occurred or not.
- The proposal distribution $q(x'|x)$ has an influence on convergence/acceptance rate and should be chosen wisely.
- If one takes a symetric proposal distribution, ie. $q(x|x') = q(x'|x)$, then the acceptance rate only depends on target density $p(x)$.
- The algorithm only requires to be able to compute ratios $\frac{p(x')}{p(x^s)}$. This will be very convenient in our setting.

MH with logistic regression

The target density is our posterior $p(w|X, y, s^2)$. We can not compute $p(w|X, y, s^2)$ because the denominator is unknown. We can however easily compute ratios:

$$\frac{p(w_1|X, y, s^2)}{p(w_2|X, y, s^2)} = \frac{p(y|X, w_1)p(w_1|s^2)}{p(y|X, w_2)p(w_2|s^2)}$$

We can use a Gaussian as a proposal density:

$$q(w|w') = \mathcal{N}(w'|w, \gamma^2 I_p),$$

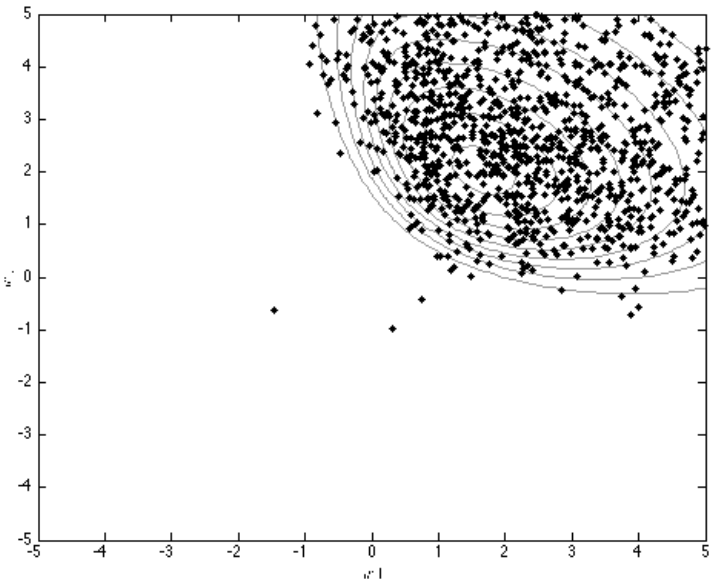
with γ a user-defined parameter. The later being symmetric, the new sample will be accepted as soon as its posterior is higher than that of the old sample.

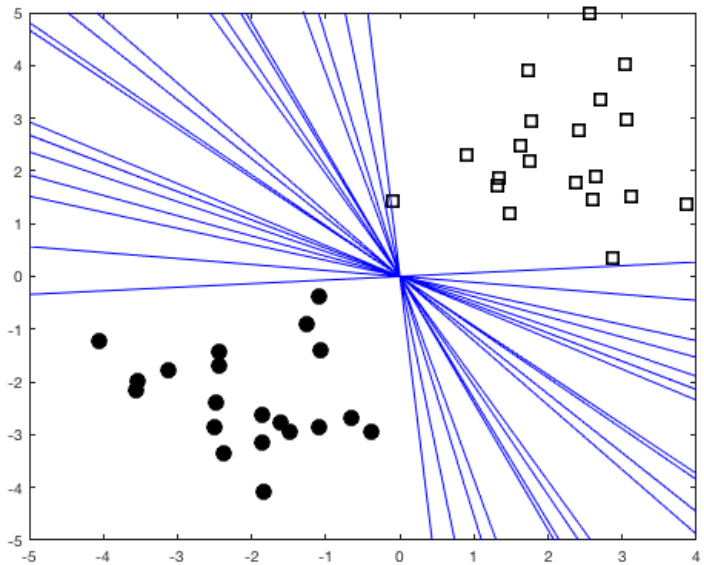
Predictions can be obtained by computing the following average:

$$P(y_{new} = 1|x_{new}, X, y, s^2) \approx \frac{1}{S} \sum_{i=1}^{S} \sigma(x_{new}^T w^s),$$

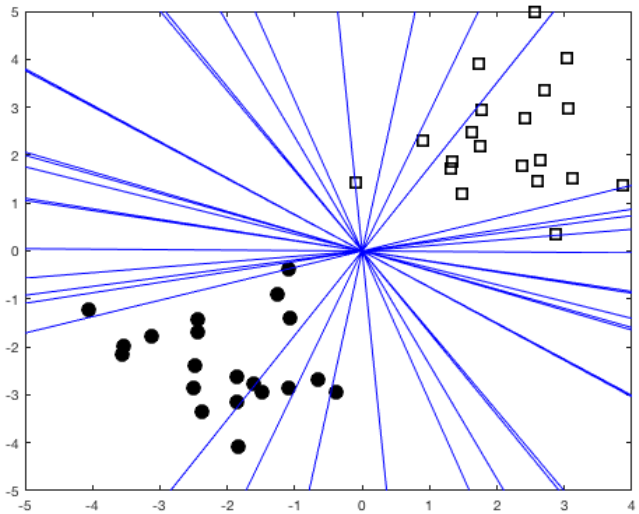
where w^s are samples generated by MH algorithm.

MH with logistic regression: sampling



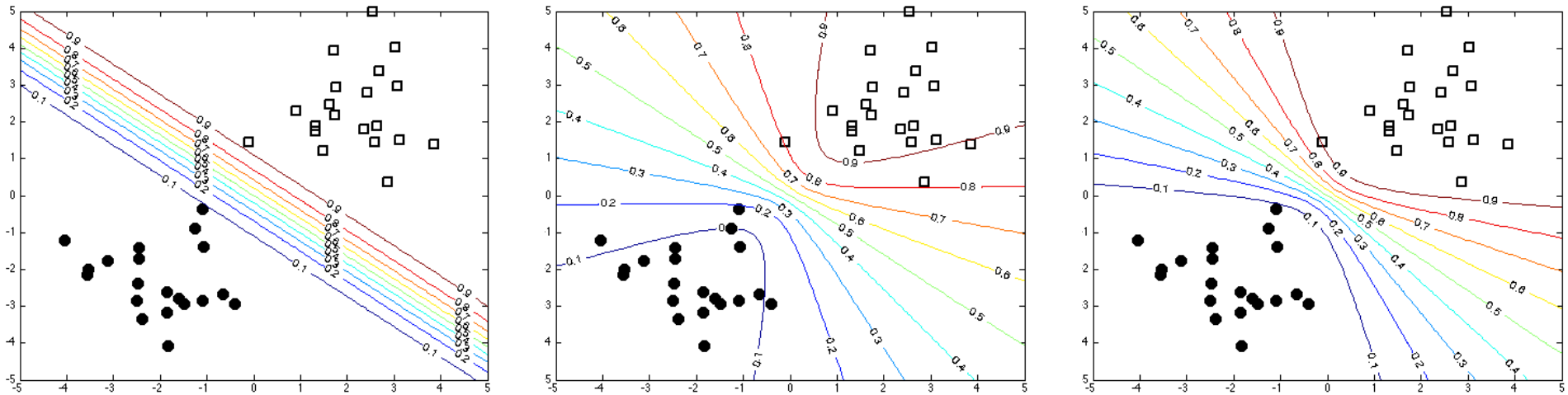


To be compared with samples from Laplace approximation:



(Rogers and Girolami, 2012)

MH with logistic regression: predictions



(Rogers and Girolami, 2012)

From left to right: MAP, Laplace approximation, Metropolis-Hastings.

Discussion

Having no analytical solution should not stop us using Bayesian approaches.

There are many solutions to approximate the posterior and the marginal likelihood when conjugate prior distributions can not be used, among which:

- Laplace approximation
- Variational approximations
- Markov Chain Monte Carlo methods
- Bayesian Information Criterion (BIC)
- Expectation Propagation (EP)
- Sequential Monte Carlo methods
- ...

See Gilles Louppe's Deep learning and Yvik Swan's Large sample analysis courses for a more detailed treatment of some of these methods.

Outline

Bayesian model selection

Let us assume that we have L models at our disposal: $\{m_1, m_2, \ldots, m_L\}$, each with its own parameters. How do we select between them using the data $\mathcal{D}$, in a bayesian way?

We can compare models on the basis of the following **model posteriors**:

$$P(m_i|\mathcal{D}) = \frac{p(\mathcal{D}|m_i)P(m_i)}{\sum_j p(\mathcal{D}|m_j)P(m_j)},$$

where $p(m_i)$ are user-defined model priors and

$$p(\mathcal{D}|m_i) = \int p(\mathcal{D}|\theta_i, m_i)p(\theta_i|m_i)d\theta_i$$

is the marginal likelihood or, in the context of model selection, the **model evidence**.

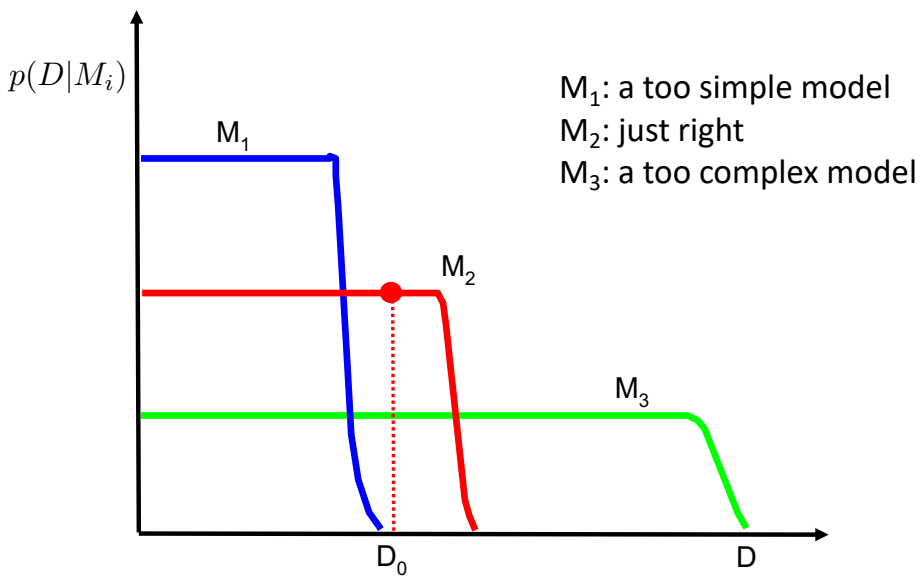
If all models are a priori equally likely, then model evidence expresses the preference shown by the data for different models. It measures the probability of generating the dataset from a model whose parameters are sampled at random from the prior.

The ratio of two model evidences is known as a **Bayes factor**: $\frac{P(\mathcal{D}|m_1)}{P(\mathcal{D}|m_2)}$.

Bayesian Occam's razor effect

If two models can both explain the data equally well (in terms of likelihood), then $P(\mathcal{D}|m)$ has an intrinsic bias towards the simplest one among them. This is called the Bayesian Occam's razor effect.

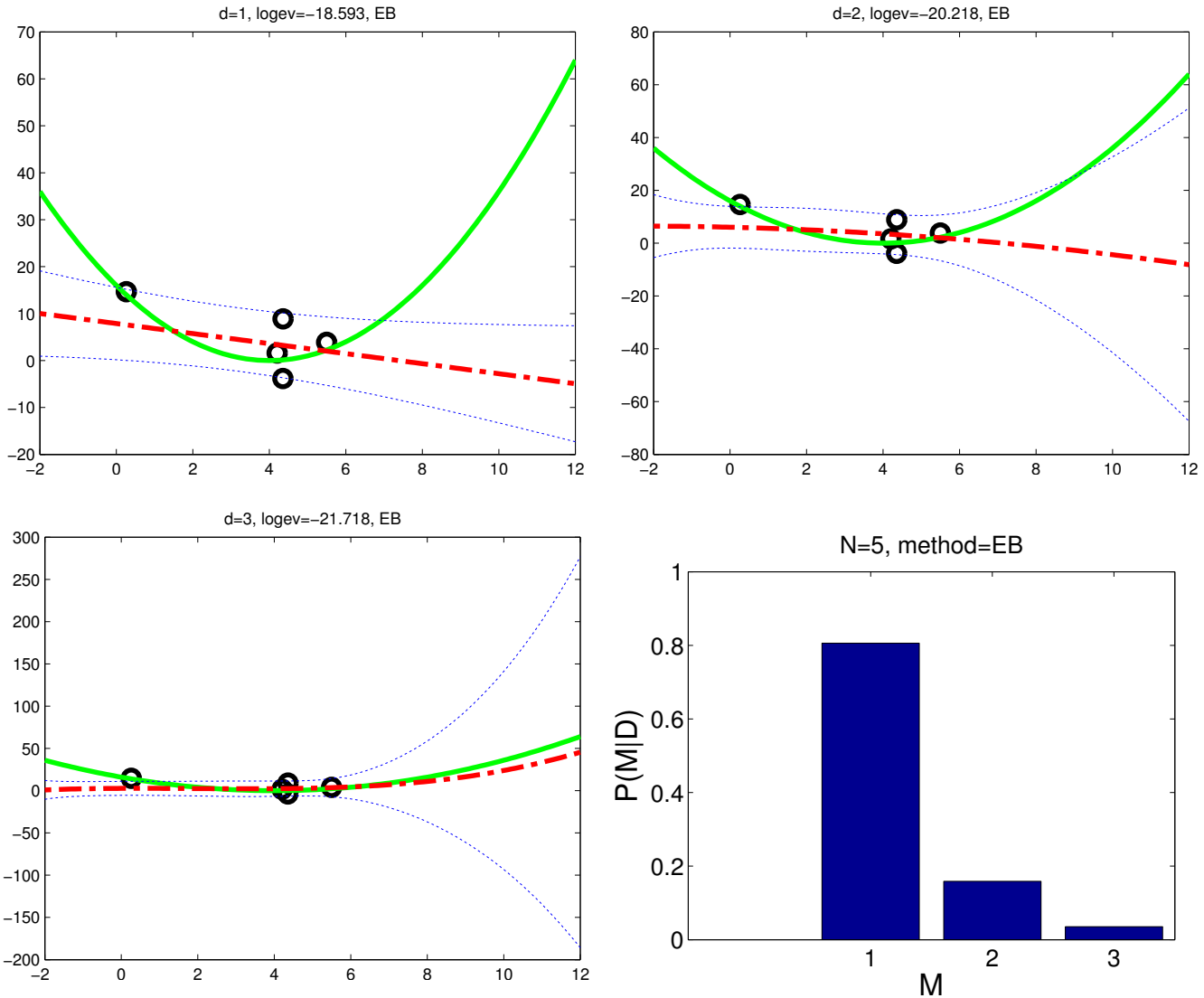
Since $\sum_{\mathcal{D}'} p(\mathcal{D}'|m) = 1$, a complex model being able to fit more datasets will attribute less probabilistic weight to each of them. A simple model, on the other hand, will be able to fit less datasets but therefore will put more weight to each of them.



(Murphy, 2012)

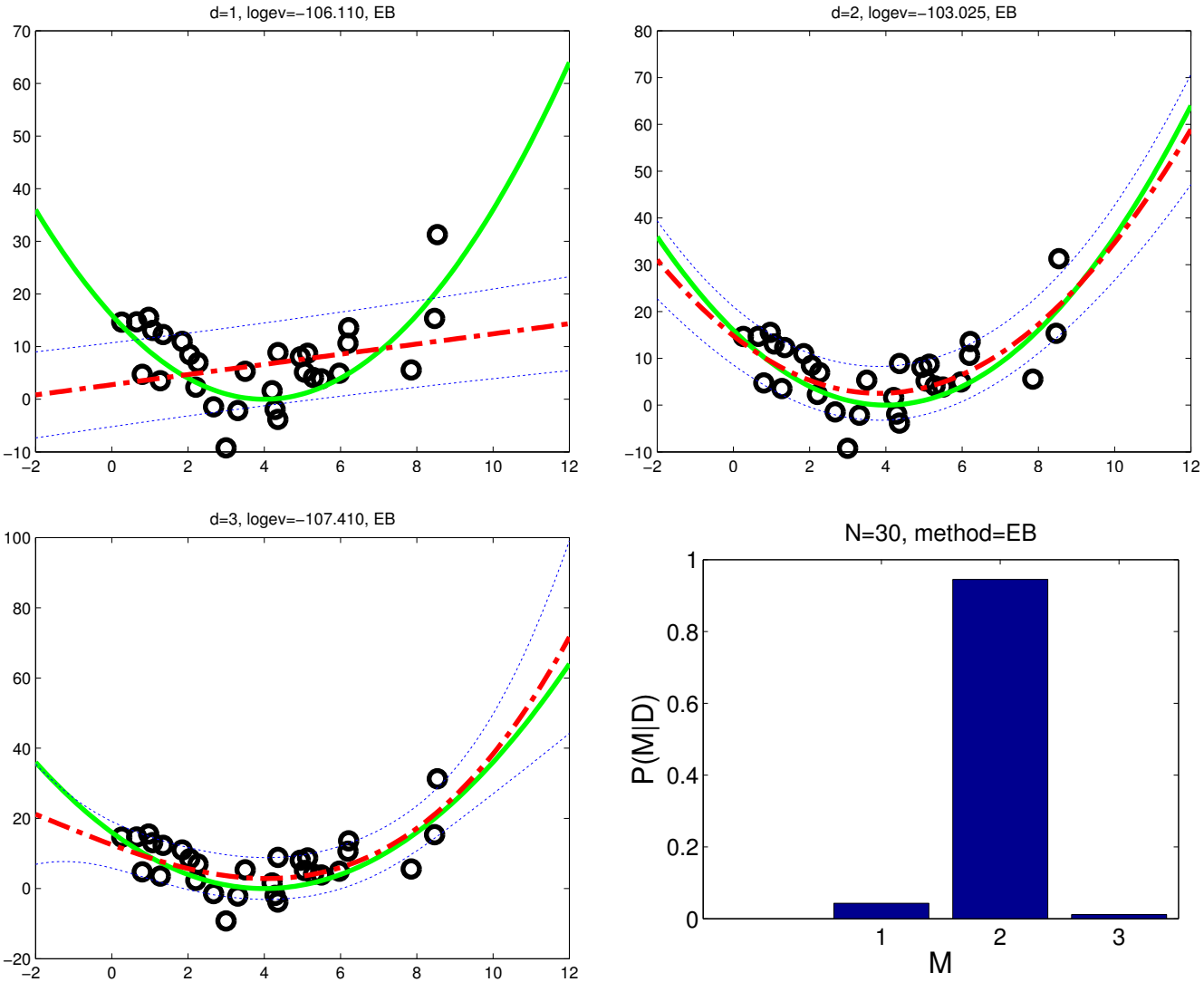
Note however that $P(\mathcal{D}|\hat{\theta}_{ML}, m_i)$ would be biased towards more complex models.

Bayesian Occam's razor effect: illustration



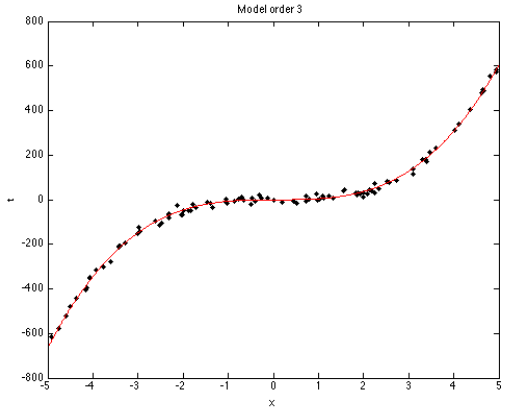
(Murphy, 2012)

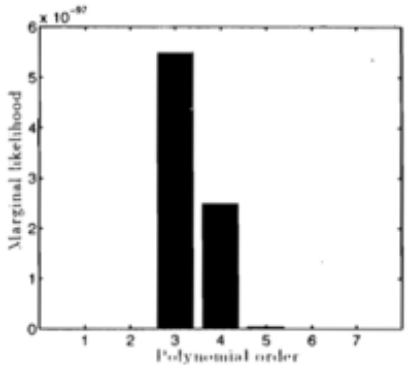
Bayesian Occam's razor effect: illustration



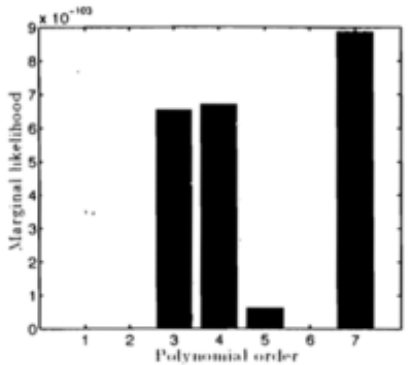
(Murphy, 2012)

Bayesian Occam's razor effect: impact of the prior

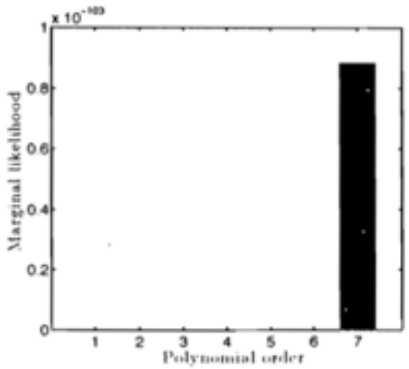




$s^2 = 0.7$



$s^2 = 0.4$



$s^2 = 0.3$

Bayesian LR with different polynomial order (from 1 to 7). The true function is $y = 5x^3 - x^2 + x$. Three prior variances are considered: $s^2 = \{0.7, 0.4, 0.3\}$. Why does the prior affect model selection?

(Rogers and Girolami, 2012)

Bayesian model averaging

Bayesian model averaging is an alternative to model selection. The predictive distribution is given by:

$$p(y_{new}|x_{new}, X, y) = \sum_{i=1}^{L} p(y_{new}|x_{new}, X, y, m_i)p(m_i|X, y)$$

The overall predictive distribution is obtained by averaging the predictive distributions of individual models, weighted by the posterior probabilities.

Note that if two models are combined with peaky distributions, the resulting predictive distribution will end up being multi-modal.

Model selection is still often preferred because it decreases complexity.

Outline

Advantages and limitations of Bayesian approaches

Advantages:

- They provide a very coherent and principled framework
- They are conceptually straightforward
- They are modular
- Often good performance (averaging effect, robust against overfitting)
- They increase model interpretability
- Adding uncertainties to predictions is crucial in many IA applications (e.g., medical applications, autonomous driving)

Limitations:

- They are subjective
- It is hard to come up with a prior. Usually, our assumptions are wrong
- The closed world assumption: need to consider all possible hypotheses before observing the data
- They can be computationally demanding
- The use of approximations weakens the coherence argument

References

Bayesian approaches in textbook:

- A first course in machine learning, Rogers and Girolami, CRC, 2012
  Chapter 3 and 4
- Machine learning: a probabilistic perspective, Murphy, MIT Press, 2012
  Chapter 5, 7.6
- Pattern recognition and machine learning, Bishop, Springer, 2006
  Sections 3.3 to 3.5

Other interesting references:

- Mackay, D. (1995) Probable Networks and Plausible Predictions - A Review of Practical Bayesian Methods for Supervised Neural Networks, David Mackay, Network: Computation in Neural Systems Vol. 6, Iss. 3.
- Ghahramani, Z. (2015) Probabilistic machine learning and artificial intelligence. Nature 521:452-459.
- Course notes of Roman Garnett:
  http://www.cse.wustl.edu/~garnett/cse515t/spring_2017/
- Course notes of Iain Murray:
  http://www.inf.ed.ac.uk/teaching/courses/mlpr/2018/notes/

Part II

Dropout as a bayesian approximation

The paper

Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning

Yarin Gal YG279@CAM.AC.UK
Zoubin Ghahramani ZG201@CAM.AC.UK
University of Cambridge

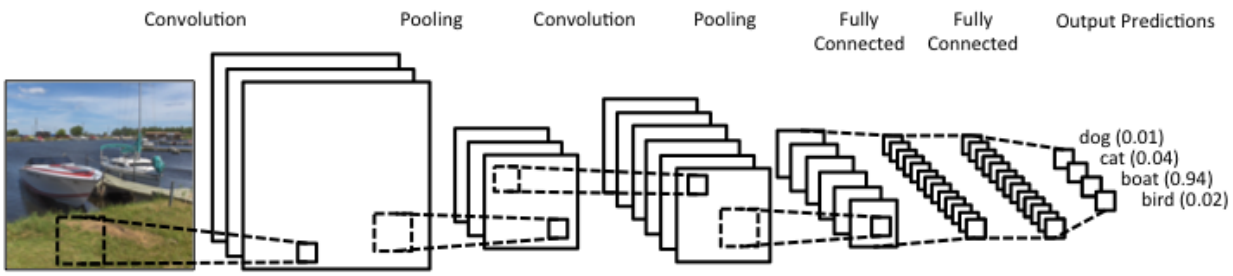
Published in the International Conference on Machine learning in 2016

Yarin Gal is (now) Associate Professor at University of Oxford and Zoubin Gharamani is Professor at the University of Cambridge and Chief Scientist at Uber.

The paper has been cited 727 times (google scholar on Feb. 11, 2019), which makes it a popular paper.

My presentation borrows ideas and figures from the authors' talks, blog posts, and phd thesis. The paper is very technical. I focus here only on the main ideas and conclusions.

Motivation



Deep learning methods (neural networks) are very effective, flexible, modular, and scalable techniques that have nowadays real-world impact.

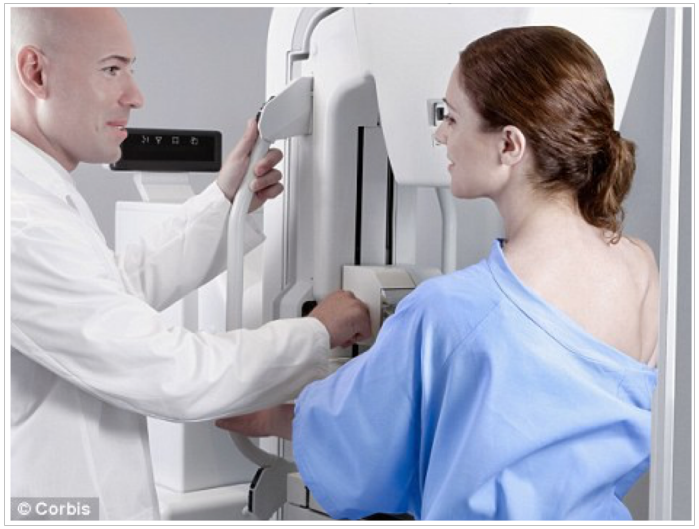
They suffer however from several limitations (Ghaharamani, 2016)

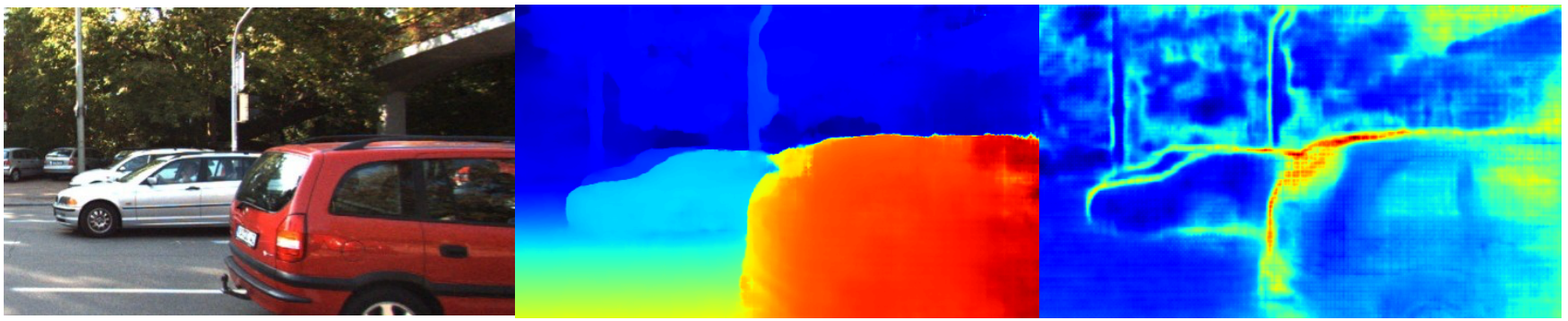
- Very data hungry and compute-intensive to train and deploy
- Uninterpretable, lacking in transparency, difficult to trust
- Easily fooled by adversarial examples (AI safety)
- Lacks solid mathematical foundations (mostly ad hoc)
- Difficult to optimise
- **Poor at representing prediction uncertainty**

Predicting uncertainty

Training set

Test example





(Kendall et al., 2017)

Bayesian methods to predict uncertainty

Bayesian methods are good candidates to address these limitations and have been explored in the deep learning community these days[1].

Existing methods are however too slow and difficult to use, which limits their practical impact.

Main contribution of this paper is to show that **dropout training**, a very common regularization technique, is identical to **approximate inference in Bayesian neural networks**.

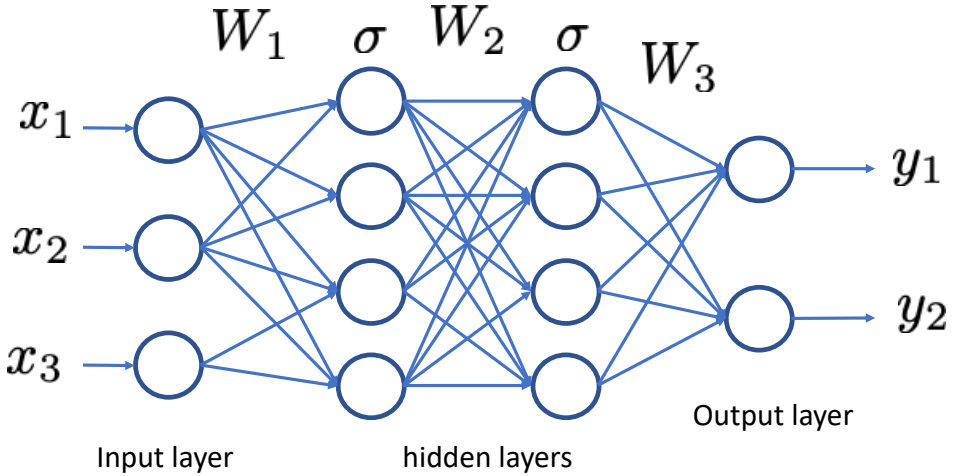
Brings a very simple and efficient approach to **train bayesian neural networks** and therefore obtain **uncertainty prediction** in deep learning.

---

[1] http://bayesiandeeplearning.org
Panel discussion: "Is bayesian deep learning the most brilliant thing ever?"
https://www.youtube.com/watch?v=HumFmLu3CJ8

Outline

Training neural networks



Multilayer perceptron (MLP) outputs are computed as (ignoring biases):

$$f(x; \mathcal{W}) = W_L \sigma\left(..W_2 \sigma(W_1 x)...\right)$$

Weights $\mathcal{W}$ are learned so as to minimise a given loss function, e.g. in regression:

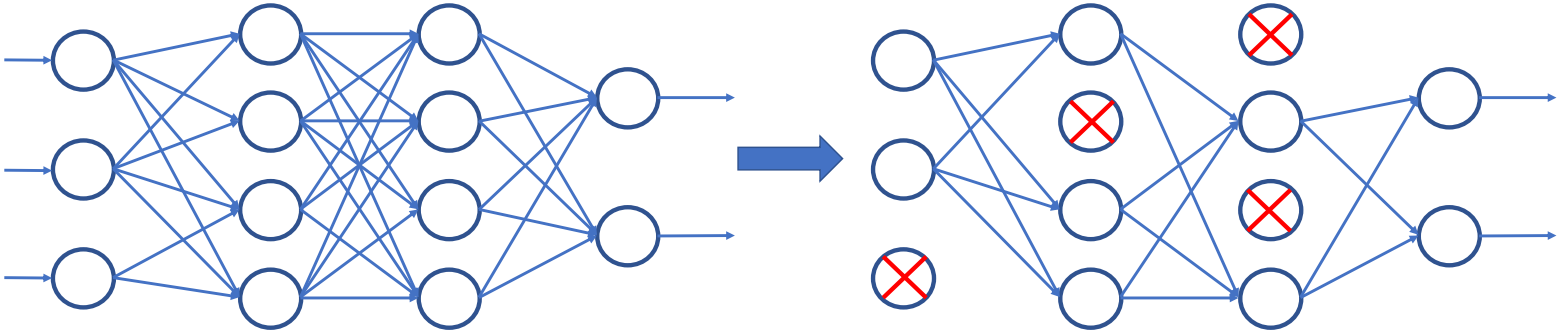
$$L_{MLP}(\mathcal{W}) = \frac{1}{N} \sum_{i=1}^{N} ||y_i - f(x; \mathcal{W})||^2 + \lambda \sum_{i=1}^{L} ||W_i||^2,$$

where the second term is a regularization term (weight decay).

Optimisation is usually carried out using stochastic gradient descent with gradients computed by the backpropagation algorithm (exploiting chain rule).

Dropout

(Srivastava et al., 2014)



Dropout is an **empirical** technique to avoid overfitting in neural networks.

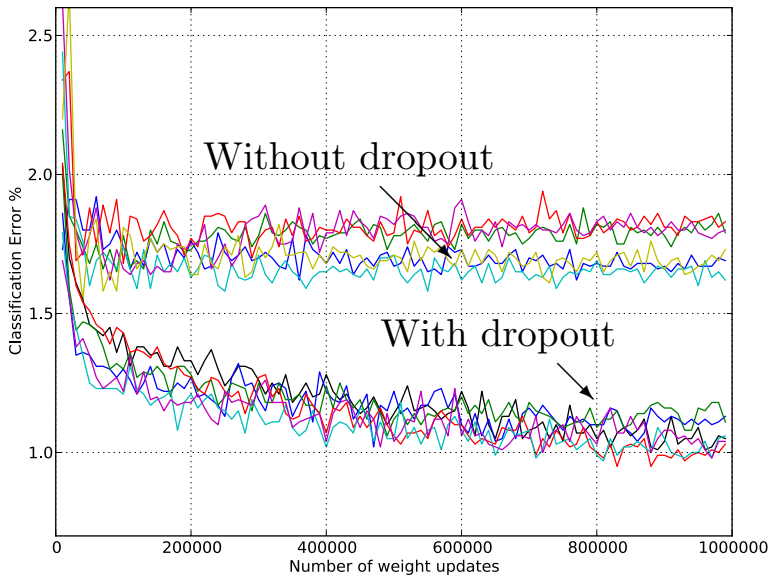
At **each training** step (i.e. for each sample within a mini-batch):

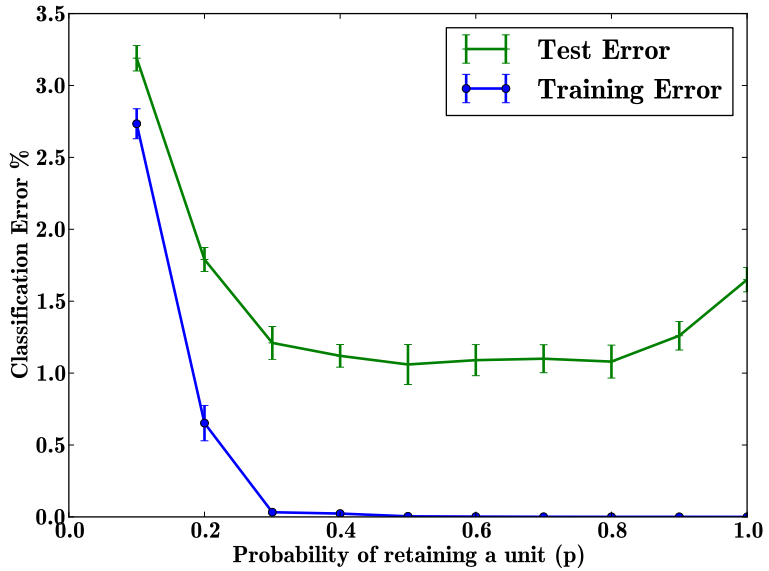
- Remove each node in the network with a probability p (e.g., $p = 0.5$)
- Update the weights of the remaining nodes with backpropagation

At **test time**, either:

- Make predictions using the trained network **without** dropout but rescaling the weights by dropout probability p *(fast and the standard approach)*
- **Sample** T neural networks using dropout and average their predictions *(**Monte-Carlo Dropout**: slower but better principled. See later)*

Dropout (Srivastava et al., 2014)

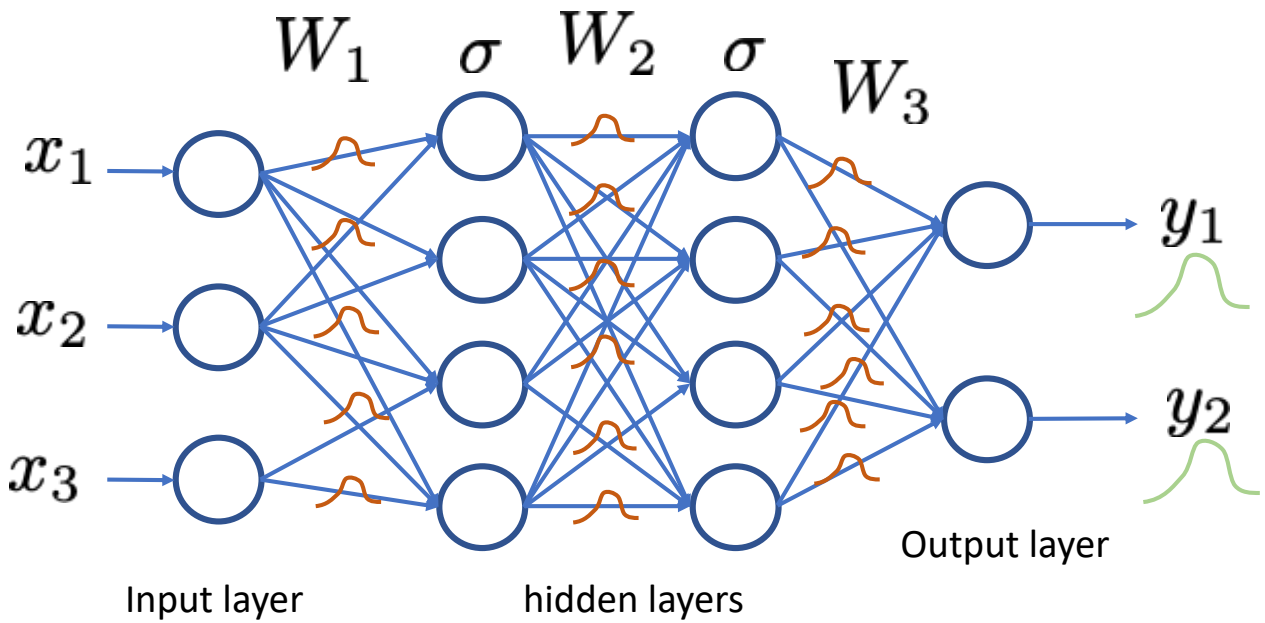




Why does dropout work?

- It makes the learned weights of a node less sensitive to the weights of the other nodes
- This forces the network to learn several independent representations of the patterns and thus decreases overfitting
- **It approximates bayesian model averaging** (this paper)

Bayesian neural networks



Bayesian neural networks are neural networks trained using Bayesian methods, with probability distribution over the network weights and outputs.

The approach is very similar to bayesian linear and logistic regression.

Bayesian neural networks

Denoting all the network weights by $\mathcal{W}$ and by $f(x; \mathcal{W})$ $(\in \mathcal{R}^d)$ the network output:

- Define a **prior** on the network weights, eg., $p(\mathcal{W}) = \mathcal{N}(\mathcal{W}; 0, s^2 I)$.
- Assume some **likelihood** model. E.g.:
  - For regression, a Gaussian likelihood:

$$p(y|x, \mathcal{W}) = \mathcal{N}(y|f(x; \mathcal{W}), \sigma^2 I)$$

  - For classification, a softmax likelihood:

$$p(y = i|x, \mathcal{W}) = \frac{\exp(f_i(x, \mathcal{W}))}{\sum_k \exp(f_k(x, \mathcal{W}))}$$

- Given a dataset $(X, Y) \in (\mathcal{R}^{N \times p}, \mathcal{R}^{N \times d})$, compute the **posterior** $p(\mathcal{W}|X, Y)$ using Bayes rule:

$$p(\mathcal{W}|X, Y) = \frac{p(Y|X, \mathcal{W})p(\mathcal{W})}{p(Y|X)}$$

with

$$p(Y|X, \mathcal{W}) = \prod_{i=1}^{N} p(y_i|x_i, \mathcal{W}) \text{ and } p(Y|X) = \int P(Y|X, \mathcal{W})P(\mathcal{W})d\mathcal{W}$$

Bayesian neural networks

- Compute the **predictive distribution** given new input x_{new}:

$$p(y_{new}|x_{new}, X, Y) = \int p(y_{new}|x_{new}, \mathcal{W}) p(\mathcal{W}|X, Y) d\mathcal{W}$$

Unfortunaly, the posterior $p(\mathcal{W}|X, Y)$ is very difficult to evaluate.

$\Rightarrow$ One must thus rely on approximate inference methods.

Bayesian neural networks

Starting in the early 90's, many inference methods have been proposed in the literature for training Bayesian neural networks[2]:

- Laplace approximation (Denker and LeCun, 1991; MacKay, 1992),
- MCMC techniques (HMC, Neal, 1993),
- Variational inference (Hinton and Van Camp, 1993, Barber and Bishop, 1998),
- Combination of SGD and MCMC (Welling and Whye Teh, 2011),
- Probabilistic adaptation of back-propagation (Hernandez-Lobato and Adams, 2015),
- ...

Despite their advantages, Bayesian approaches are still not the standard for neural network training, mainly because of their high computational cost.

[2] See "Uncertainty in deep learning", Yarin Gal, Phd thesis, 2016 for a review of these works.

Using variational inference

Approximate the posterior $p(\mathcal{W}|X, Y)$ with a parametric distribution $q(\mathcal{W}; \alpha)$ by minimising KL divergence $D_{KL}(q(\mathcal{W}; \alpha)||p(\mathcal{W}|X, Y))$.

As shown earlier, this is equivalent to maximising the **evidence lower bound** (ELBO):

$$L(\alpha) = \int q(\mathcal{W}; \alpha) \log(Y|X, \mathcal{W}) d\mathcal{W} - D_{KL}(q(\mathcal{W}; \alpha)||p(\mathcal{W}))$$

which can be done in a stochastic way:

Repeat:

- Sample $\hat{\mathcal{W}} \sim q(\mathcal{W}; \alpha)$
- Do one step of maximisation (w.r.t. α) of:

$$\hat{L}(\alpha) = \log(p(Y|X, \hat{\mathcal{W}})) - D_{KL}(q(\mathcal{W}; \alpha)||p(\mathcal{W}))$$

Using a specific approximate distribution q, **optimal variational parameters** can be obtained directly from the weights of standard neural networks trained **with dropout**.

Approximate distribution q corresponding to dropout

Let us split the weights $\mathcal{W}$ per layer:

$$\mathcal{W} = \{W_1, W_2, \ldots, W_L\},$$

where W_i is further split per neuron

$$W_i = \{w_{i,1}, \ldots, w_{i,K_i}\},$$

with K_i the number of neurons in layer i and $w_{i,k}$ the weights of the connections going out of the kth neuron in layer i.

Variational parameters α are split similarly into $\alpha = \{M_1, \ldots, M_L\}$ with $M_i = \{m_{i,1}, \ldots, m_{i,K_i}\}$.

The proposed **approximate distribution** $q(\mathcal{W}; \alpha)$ is defined as follows:

$$\begin{aligned} q(\mathcal{W}; \alpha) &= \prod_{i=1}^{L} q(W_i; M_i) \\ q(W_i; M_i) &= \prod_{k=1}^{K_i} q(w_{i,k}; m_{i;k}) \\ q(w_{i,k}; m_{i,k}) &= p\delta_0(w_{i,k}) + (1-p)\delta_{m_{i,k}}(w_{i,k}) \end{aligned}$$

where $\delta_a(x)$ denotes a (multivariate) delta dirac distribution centered at a.

Sampling from q

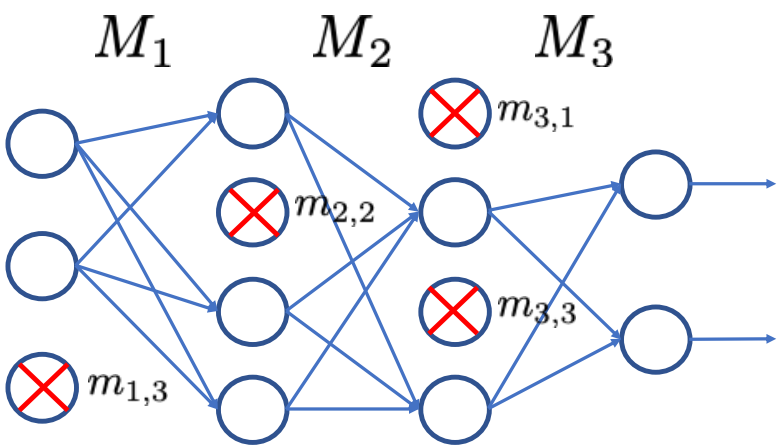
Given the previous definition, **sampling** parameters $\hat{\mathcal{W}} = \{\hat{W}_1, \ldots, \hat{W}_L\}$ **from** q is done as follows:

- Draw binary random variables $\hat{z}_{i,k} \sim \text{Bernouilli}(1-p)$ for each layer i and neuron k
- Compute $\hat{W}_i = M_i.\text{diag}([\hat{z}_{i,k}]_{k=1}^{K_i})$

where W_i, resp. M_i, is a matrix composed of the columns $w_{i,k}$, resp. $m_{i,k}$.

Matrices $\hat{W}_i$ are thus obtained by setting columns of M_i to zero with probability p.

This is **strictly equivalent to dropout**, i.e, removing neurons from the network with probability p.



Stochastic variational inference

The algorithm to fit variational parameters becomes:

Repeat:

- Sample $\hat{\mathcal{W}} \sim q(\mathcal{W}; \alpha) \Leftrightarrow$ Randomly set units of the network to zero $\Leftrightarrow$ dropout
- Do one step of maximisation of:

$$\hat{L}(\alpha) = \log(p(Y|X, \hat{\mathcal{W}})) - D_{KL}(q(\mathcal{W}; \alpha)||p(\mathcal{W}))$$

w.r.t. $\alpha = \{M_i\}_{i=1}^{L}$.

One can show that the second step is equivalent to one iteration of standard training of the (pruned) network with a specific loss function.

Intuitive explanation

Maximisation step is equivalent to minimizing:

$$-\hat{L}(\alpha) = -\log(p(Y|X,\hat{\mathcal{W}})) + D_{KL}(q(\mathcal{W};\alpha)||p(\mathcal{W}))$$

w.r.t. $\alpha = \{M_i\}_{i=1}^{L}$.

- Given the likelihood model $p(y|x,\mathcal{W}) = \mathcal{N}(y; f(X;\mathcal{W}), \sigma^2 I)$, the first term is proportional to mean squared error:

$$-\log(p(Y|X,\hat{\mathcal{W}})) \propto \frac{1}{N}\sum_{i=1}^{N}||y_i - f(x_i;\hat{\mathcal{W}})||^2$$

- Using a gaussian prior $P(\mathcal{W}) = \mathcal{N}(\mathcal{W}; 0, s^2 I)$, the paper shows that:

$$D_{KL}(q(\mathcal{W};\alpha)||p(\mathcal{W})) \propto \sum_{i=1}^{L}||M_i||_2^2$$

Intuitive explanation

Optimization step of variational inference thus amounts at:

- Setting the network weights according to current M_i values taking into account dropped out neurons
- Optimising these weights (with back-propagation) using the following loss (mean squared error with weight decay):

$$\frac{1}{N}\sum_{i=1}^{N}||y_i - f(x_i;\hat{\mathcal{W}})||^2 + \lambda\sum_{i=1}^{L}||M_i||$$

Note: In the paper, λ, p, and σ^2 are fixed by the user (or tuned by cross-validation, see later), and s^2 is derived from them.

Getting predictions and uncertainty estimates

When **training a network with dropout**, we thus fit the parameter α of a distribution $q(\mathcal{W}; \alpha)$ to **match the posterior distribution of the weights** in a Bayesian framework.

Main benefit: a prediction together with **uncertainty estimates** at some point x_{new} can be obtained in a principled way using Monte-Carlo integration (proofs are in the paper):

- Draw T sets of network parameters $\{\hat{\mathcal{W}}^t\}_{t=1}^T$ from $q(\mathcal{W}; \alpha)$
- Propagate x_{new} in the T networks to get predictions $\{f(x_{new}; \mathcal{W}^t)\}_{t=1}^T$
- Approximate the predictive mean and variance as follows:

$$E_{p(y|x,\mathcal{D})}\{y\} \approx \frac{1}{T}\sum_{t=1}^{T} f(x_{new}; \mathcal{W}^t)$$

$$\text{Var}_{p(y|x,\mathcal{D})}\{y\} \approx \sigma^2 I + \frac{1}{T}\sum_{t=1}^{T} f(x_{new}; \mathcal{W}^t) f(x_{new}; \mathcal{W}^t)^T - E\{y\}E\{y\}^T$$

Better understanding dropout

This equivalence sheds new light on dropout:

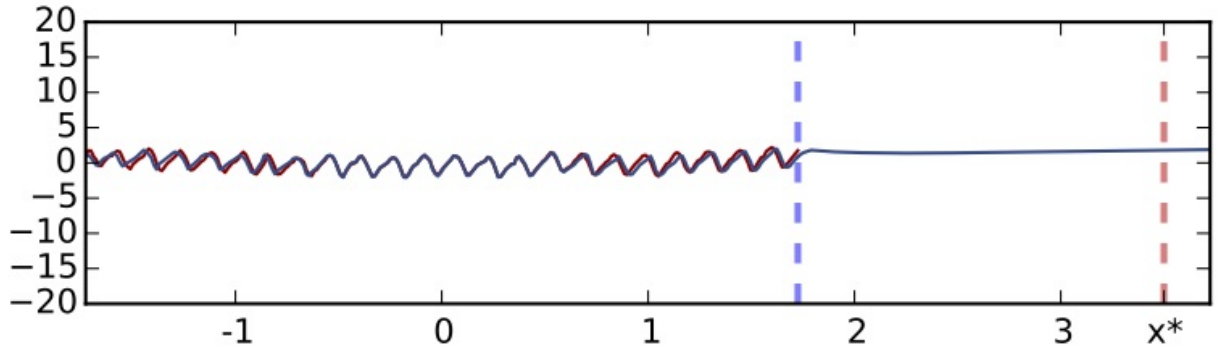
- Dropout reduces overfitting because it approximately **integrates over model parameters**.
- **Single pass implementation** of dropout at test time uses the mean parameters over the posterior ($E\{W_i\} = pM_i$)
- Dropout implicitely **constrains the weights to be near zero**:
  - One can show that $\text{Var}(W_i) = M_i \cdot M_i^T \cdot p(1-p)$
  - Since posterior uncertainty decreases with more data, average weights have to decrease with more data at fixed p
  - The closer p to 0.5, the stronger the regularization
- Other **stochastic regularization techniques** (drop-connect, multiplicative Gaussian noise...) correspond to different variational distributions.
- Suggested **new ways** to apply dropout with convolutional and recurrent networks (see the first author's publications).

Outline

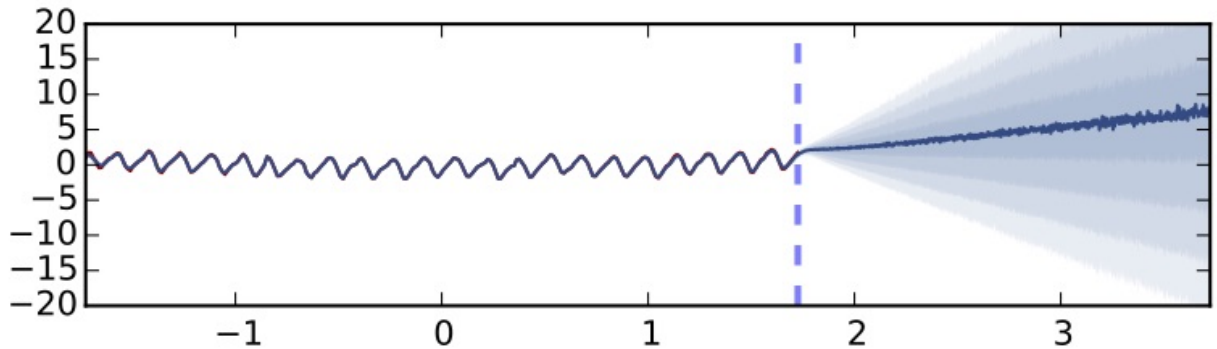
Illustrations in regression

Prediction of the CO_2 concentration level in Mauna Loa, Hawaii, in 20 years' time.

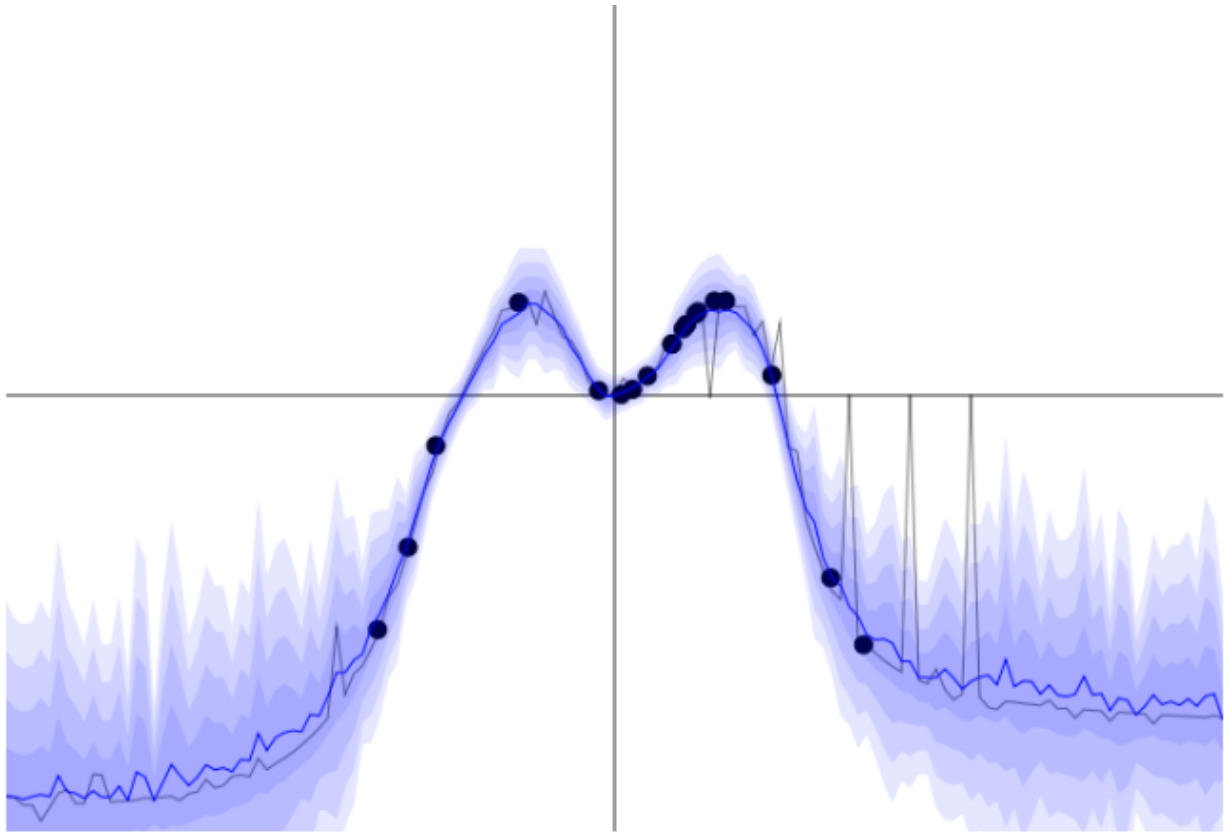
- Normal dropout (weight averaging, 5 layers, 1024 hidden units, ReLU units):



- Same network, Bayesian perspective:



Illustrations in regression



Online demo

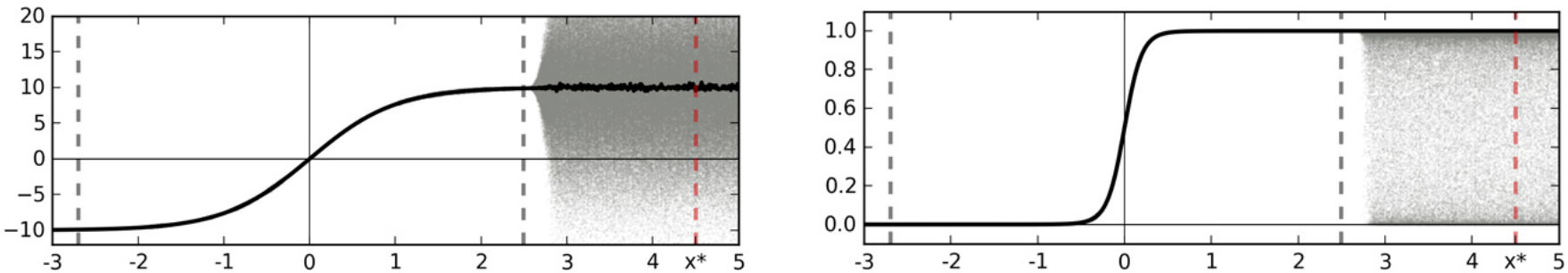
Illustrations in classification

In classification, a non-bayesian neural network outputs a probability for each class using a softmax function:

$$p(y = i|x, \mathcal{W}) = \frac{\exp(f_i(x; \mathcal{W}))}{\sum_k \exp(f_k(x; \mathcal{W}))}$$

Like for logistic regression with the MAP estimate (see Slide 37), the **softmax output** of a neural network **does not provide** a good indication of prediction **uncertainty**.

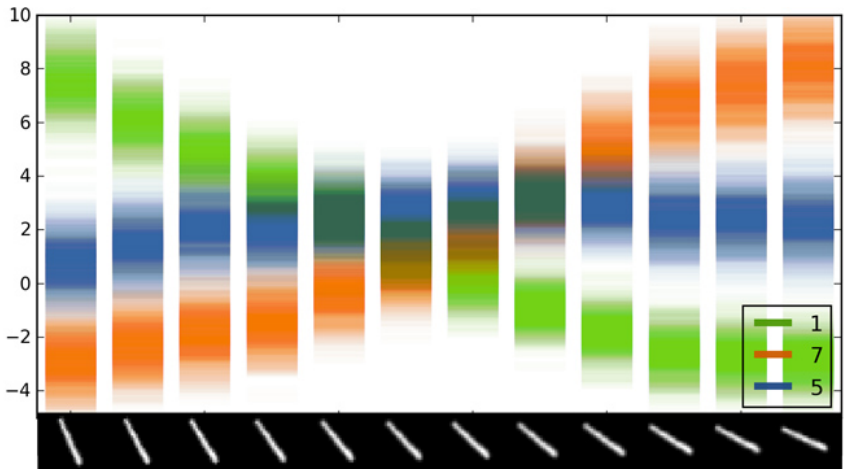
An illustration from the paper:



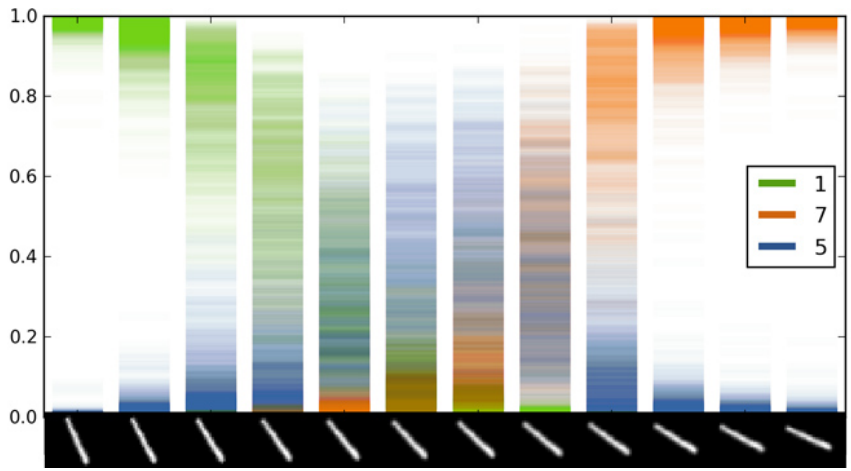
Left: softmax input, right: softmax output for an hypothetical classification problem. Average prediction in solid line, uncertainty in grey.

Illustrations in classification

Real experiment on MNIST:



Softmax input



Softmax output

Scatter of 100 forward passes of the softmax input and output for dropout LeNet for a rotated image of digit 1.

How good are the uncertainty estimates?

Common experiment for assessing uncertainty prediction methods:

- Take a regression dataset *(repeat for several datasets)*
- Split it into training, validation, and test set
  (repeat and average over several splits)
- Train a model on the training set, use the validation set to tune hyper-parameters and validate the final model on the test set

Two evaluation metrics:

- **Root mean squared error** (RMSE): $\sqrt{\frac{1}{N}\sum_{i=1}^{N_{TS}}(y_i^{obs} - y_i^{pred})^2}$
- Average predictive **log-likelihood** (LL):

$$\frac{1}{N_{LS}}\sum_{i=1}^{N_{TS}} \log p(y_i^{obs}|x_i^{obs}, X, Y)$$

RMSE only assesses the quality of point predictions. LL also assesses the quality of the uncertainty.

Predictive log-likelihood to assess uncertainty

For a test example (x_{new}, y_{new}):

$$\begin{aligned}
\log p(y_{new}|x_{new}, X, Y) &= \log \int p(y_{new}|x_{new}, \mathcal{W}) p(\mathcal{W}|X, Y) d\mathcal{W} \\
&\approx \log \int p(y_{new}|x_{new}, \mathcal{W}) q(\mathcal{W}; \alpha) d\mathcal{W} \\
&\approx \log \left(\frac{1}{T} \sum_{t=1}^{T} p(y_{new}|x_{new}, \mathcal{W}^t) \right)
\end{aligned}$$

with $\mathcal{W}^t$ drawn from $q(\mathcal{W}; \alpha)$.

Given the likelihood model $p(y|x, \mathcal{W}) = \mathcal{N}(y|f(x; \mathcal{W}), \sigma^2 I)$, we have:

$$\log p(y_{new}|x_{new}, X, Y) \approx \text{logsumexp} \left(-\frac{1}{2\sigma^2} ||y_{new} - f(x; \mathcal{W}^t)||^2 \right) - \frac{1}{2} log\sigma^2 + Cst,$$

where σ is tuned on the validation set.

Too large uncertainty is penalized by the second term but too low uncertainty (over-confidence) on bad predictions is penalized by the first term.

Dropout versus other bayesian deep learning methods

In the original 2016 paper (in terms of log-likelihood):

	Avg. Test LL and Std. Errors		
Dataset	**VI**	**PBP**	**Dropout**
Boston Housing	-2.90 ±0.07	-2.57 ±0.09	**-2.46 ±0.25**
Concrete Strength	-3.39 ±0.02	-3.16 ±0.02	**-3.04 ±0.09**
Energy Efficiency	-2.39 ±0.03	-2.04 ±0.02	**-1.99 ±0.09**
Kin8nm	0.90 ±0.01	0.90 ±0.01	**0.95 ±0.03**
Naval Propulsion	3.73 ±0.12	3.73 ±0.01	**3.80 ±0.05**
Power Plant	-2.89 ±0.01	-2.84 ±0.01	**-2.80 ±0.05**
Protein Structure	-2.99 ±0.01	-2.97 ±0.00	**-2.89 ±0.01**
Wine Quality Red	-0.98 ±0.01	-0.97 ±0.01	**-0.93 ±0.06**
Yacht Hydrodynamics	-3.43 ±0.16	-1.63 ±0.02	**-1.55 ±0.12**
Year Prediction MSD	-3.622 ±NA	-3.603 ±NA	**-3.588 ±NA**

More recent experiments against other methods (Mukhoti et al., 2018)

Dataset	**Dropout (Timed Setting)**	**Dropout (Convergence)**	**Dropout (Hyperparameter tuning)**	**VMG**	**HS-BNN**	**PBP-MV**	**SGHMC (Tuned per dataset)**	**SGHMC (Scale Adapted)**
Boston Housing	-2.46 ± 0.06	$\mathbf{-2.40 \pm 0.04}$	$\mathbf{-2.40 \pm 0.04}$	-2.46 ± 0.09	-2.54 ± 0.15	-2.54 ± 0.08	-2.49 ± 0.15	-2.54 ± 0.04
Concrete Strength	-3.04 ± 0.02	-2.97 ± 0.02	$\mathbf{-2.93 \pm 0.02}$	-3.01 ± 0.03	-3.09 ± 0.06	-3.04 ± 0.03	-4.17 ± 0.72	-3.38 ± 0.24
Energy Efficiency	-1.99 ± 0.02	-1.72 ± 0.01	-1.21 ± 0.01	-1.06 ± 0.03	-2.66 ± 0.13	$\mathbf{-1.01 \pm 0.01}$	$--$	$--$
Kin8nm	0.95 ± 0.01	0.97 ± 0.00	1.14 ± 0.01	1.10 ± 0.01	1.12 ± 0.03	$\mathbf{1.28 \pm 0.01}$	$--$	$--$
Naval Propulsion	3.80 ± 0.01	3.91 ± 0.01	4.45 ± 0.00	2.46 ± 0.00	$\mathbf{5.52 \pm 0.10}$	4.85 ± 0.06	$--$	$--$
Power Plant	-2.80 ± 0.01	-2.79 ± 0.01	-2.80 ± 0.01	-2.82 ± 0.01	-2.81 ± 0.03	$\mathbf{-2.78 \pm 0.01}$	$--$	$--$
Protein Structure	-2.89 ± 0.00	-2.87 ± 0.00	-2.87 ± 0.00	-2.84 ± 0.00	-2.89 ± 0.00	$\mathbf{-2.77 \pm 0.01}$	$--$	$--$
Wine Quality Red	-0.93 ± 0.01	$\mathbf{-0.92 \pm 0.01}$	-0.93 ± 0.01	-0.95 ± 0.01	-0.95 ± 0.05	-0.97 ± 0.01	-1.29 ± 0.28	-1.04 ± 0.17
Yacht Hydrodynamics	-1.55 ± 0.03	-1.38 ± 0.01	-1.25 ± 0.01	-1.30 ± 0.02	-2.33 ± 0.01	-1.64 ± 0.02	-1.75 ± 0.19	$\mathbf{-1.10 \pm 0.08}$

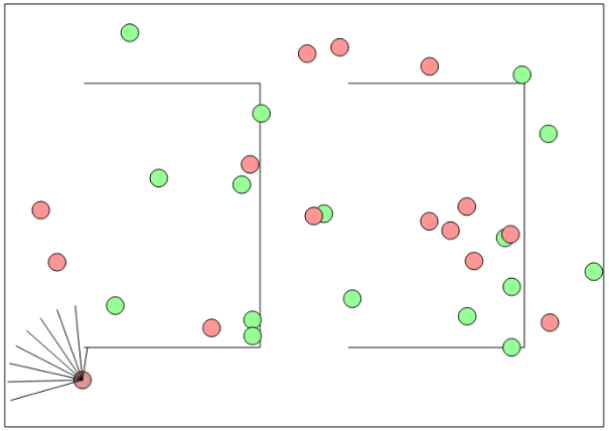
(See the paper for a description of competitors)

Overall, despite its simplicity, Dropout is very competitive in terms of both RMSE and LL. It is only inferior to PBP-MV (Sun et al., 2017).

Other applications

Reinforcement learning

(online demo here)



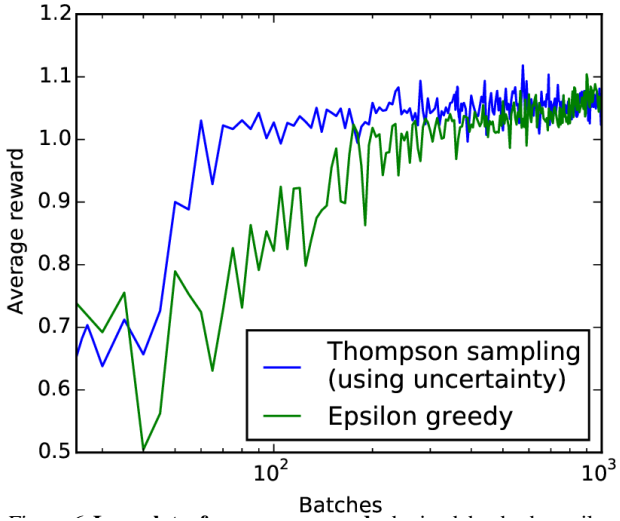
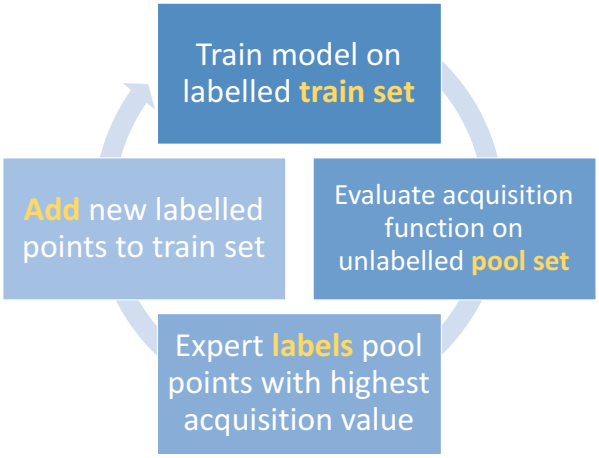
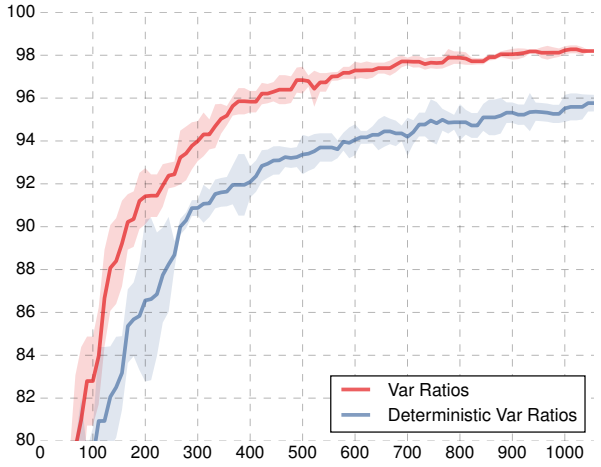


Figure 5. Depiction of the reinforcement learning problem used in

Active learning

(Gal et al., 2017)





Outline

Discussion: contributions of the paper

Bayesian methods are a very natural way to obtain principled uncertainty predictions from deep learning models.

The approach proposed in the paper has several strong advantages:

- It is extremely simple and generic (MLP, CNN, RNN, etc.)
- It is very efficient (no cost if models have already been trained with Dropout)
- It works well on benchmark problems in comparison with other techniques (in terms of RMSE and log-likelihood)
- It has been successfully exploited in several practical applications by the authors (in vision, RL, active learning, etc.).

In addition, the paper gives new insight on why Dropout works and the framework should help designing new stochastic regularization techniques or bayesian approaches for neural networks.

Discussion: limitations

The variational distribution has been derived specifically to match Dropout.

There is thus no guarantee that this is a good candidate to approximate the posterior and thus no guarantee that the uncertainty predictions will be accurate.

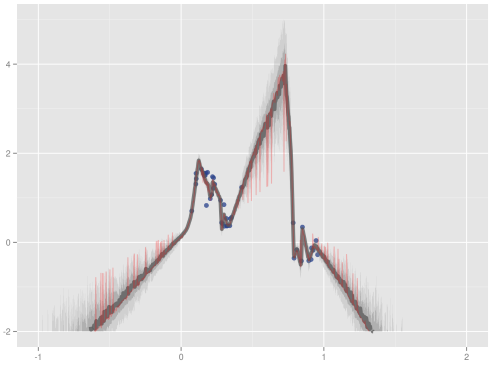
Actually, given the very discrete nature of this distribution, it is very unlikely that it will match very well the true posterior.

Some papers have observed unexpected results with dropout in some specific situations, e.g. (Osband et al., 2016) or (Osband, 2016).

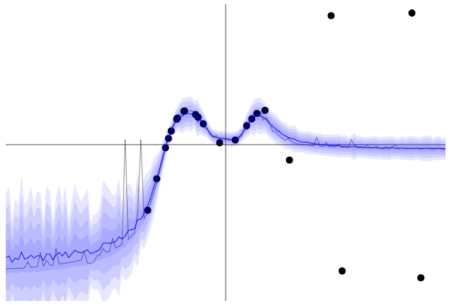
The first author has corrected some of the problems (see here). In addition, despite these problems, experimental results with dropout are still very good in comparison with other methods (see (Mukhoti et al., 2018)).

Discussion: limitations

Problem with dropout highlighted in (Osband et al., 2016):

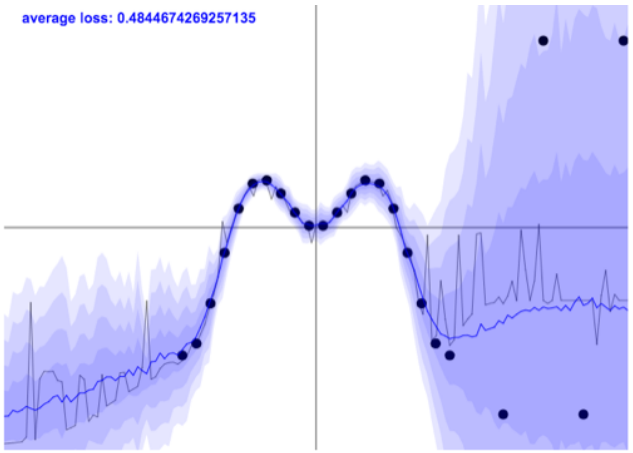


(a) Dropout gives strange uncertainty estimates.



(b) Screenshot from accompanying web demo to [7]. Dropout converges with high certainty to the mean value.

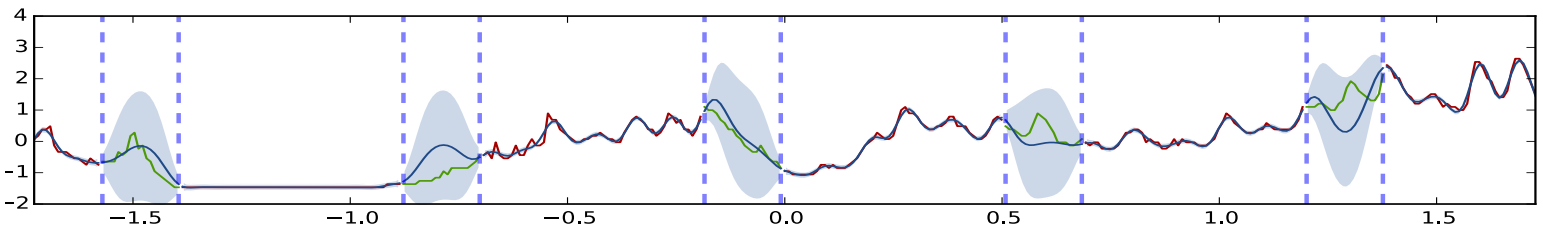
The Problem with the right figure can be solved by making noise σ^2 a function of x (heteroscedastic model) (see here):



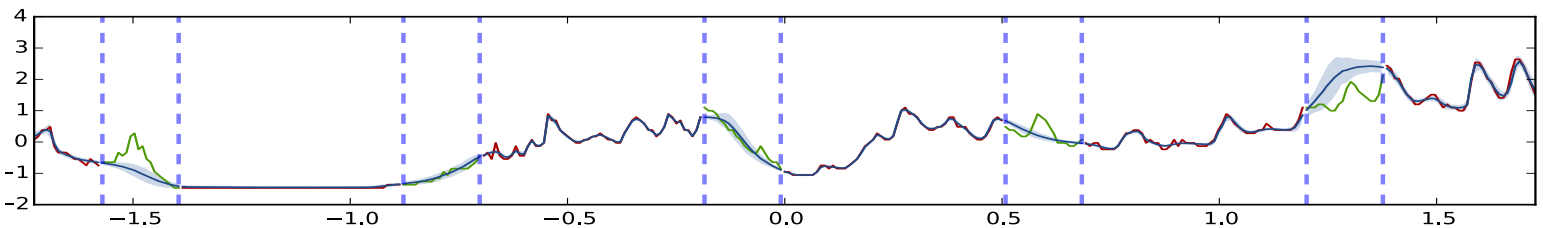
Discussion: limitations

In the recent literature, bayesian neural networks do not seem to be compared with other techniques such as Gaussian processes, which are very good at measuring uncertainty.

From the appendix of the paper:



(a) Gaussian process with SE covariance function



(b) MC dropout with ReLU non-linearities

It is not clear how to incorporate arbitrary priors within the approach, which remains one of the interest of bayesian methods.

References

- Most of the references in the presentation are hyperlinks to the papers.
- Yarin Gal's webpage: http://www.cs.ox.ac.uk/people/yarin.gal/website/