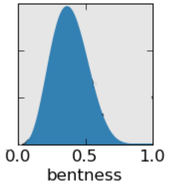
COMP421 - Machine Learning

“Bayesian” neural nets?

This is a bit of an adventure, but we’ll learn about MCMC on the way, which is good.

3 ways of predicting

- the ***maximum likelihood*** approach is to make predictions using the parameter value most likely to have generated the data:



$$P(x^{\text{new}}|\mathcal{D}) = P(x^{\text{new}}|b^{\text{ML}})$$
$$b^{\text{ML}} = \underset{b}{\text{argmax}} P(\mathcal{D}|b)$$

- the ***maximum a posteriori*** (MAP) approach is use the most plausible parameter value ("the one you believe in the most"):

$$P(x^{\text{new}}|\mathcal{D}) = P(x^{\text{new}}|b^{\text{MAP}}) \quad \text{where } b^{\text{MAP}} = \underset{b}{\text{argmax}} P(b|\mathcal{D})$$

- the ***Bayesian*** approach is *integrate out* the unknown:

$$P(x^{\text{new}}|\mathcal{D}) = \int P(x^{\text{new}}, b|\mathcal{D})\, db$$

fully Bayesian prediction in a neural net

Parameters W, say.

Bayesian predictive distribution for an unknown $y^\star$ (“output”), given all the data $\mathcal{D}$ and some new input $x^\star$:

$$\begin{aligned} P(y^\star|\mathcal{D}, \mathbf{x}^\star) &= \int dW \;\; P(y^\star, W|\mathcal{D}, \mathbf{x}^\star) \\ &= \int dW \;\; P(y^\star|W, \mathcal{D}, \mathbf{x}^\star) P(W|\mathcal{D}, \mathbf{x}^\star) \\ &= \int dW \;\; P(y^\star|W, \mathbf{x}^\star) P(W|\mathcal{D}) \end{aligned}$$

How to do that integral over the posterior weights distribution then?

We could approximate the predictive distribution (the integral) by taking lots of samples from $P(W|\mathcal{D})$, and evaluating $P(y^\star|W, \mathbf{x}^\star)$ for each one. How to do this?

rejection sampling?

Make some samples, and throw out all those that don’t give the correct $\mathcal{D}$

Q: what’s wrong with this?

Markov Chain Monte Carlo sampling

w exists in a “state space” with lots of dimensions.

Basic idea: make a dynamical system that jumps around this space, and is guaranteed to visit states with probabilities equal to those of the posterior.

Transitions between states:

$$M_{w \to w'} = \Pr(\text{state } w \text{ jumps to state } w')$$

The stationary or equilibrium distribution $\boldsymbol{\pi}$ is a distribution for which

$$\boldsymbol{\pi} = \mathbf{M}\boldsymbol{\pi}$$

If w were discrete, and you knew $\mathbf{M}$, you could find π by solving this (huge) equation.
But let’s not.

reversible Markov Chains, and detailed balance

If a Markov Chain is *reversible*, we have the much stronger condition that, known as “detailed balance”:

$$\underbrace{M_{w \to w'}\, \boldsymbol{\pi}_w}_{\text{flow from } w \text{ to } w'} = \underbrace{M_{w' \to w}\, \boldsymbol{\pi}_{w'}}_{\text{flow from } w' \text{ to } w}$$

for all w, w'.

We will now **design** a rule for transitions (M) for which the above is true and $\boldsymbol{\pi}$ is the distribution that we want to sample from.

In fact there are two rules that do this “Markov Chain Monte Carlo” (MCMC):

1. Metropolis Sampler (do now)
2. Gibbs Sampler (not covered)

Metropolis Sampler

A reversible Markov Chain obeys detailed balance. For all w, w':

$$\underbrace{M_{w \to w'}\, \boldsymbol{\pi}_w}_{\text{flow from } w \text{ to } w'} = \underbrace{M_{w' \to w}\, \boldsymbol{\pi}_{w'}}_{\text{flow from } w' \text{ to } w} \quad \Leftrightarrow \quad \frac{M_{w \to w'}}{M_{w' \to w}} = \frac{\boldsymbol{\pi}_{w'}}{\boldsymbol{\pi}_w}$$

Two steps, for each “jump”:

1. **Propose** w' with probability $Q_{w'|w}$
   and we choose Q to be some distribution that’s really easy to sample from,
2. **Accept** transition with probability $A_{w'|w}$

So the overall probability of transition is

$$M_{w'|w} = Q_{w'|w} A_{w'|w}$$

Suppose we have a “desired” distribution, P_w for short. Satisfy yourself that detailed balance with $\boldsymbol{\pi}_w = P_w$ follows if we choose the right $A_{w'|w}$, which is...

deriving the right acceptance probability

We want to make

$$\frac{M_{w \to w'}}{M_{w' \to w}} = \frac{P_{w'}}{P_w}$$

plugging in M...

$$\frac{Q_{w'|w} A_{w'|w}}{Q_{w|w'} A_{w|w'}} = \frac{P_{w'}}{P_w}$$

rearranging...

$$\frac{A_{w'|w}}{A_{w|w'}} = \frac{P_{w'}\; Q_{w|w'}}{P_w\; Q_{w'|w}}$$

so we choose...

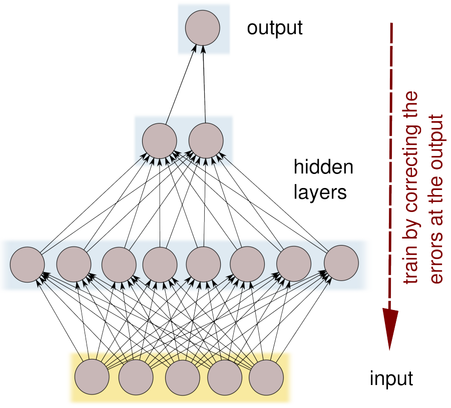
$$A_{w'|w} = \min\left(1, \frac{P_{w'}\; Q_{w|w'}}{P_w\; Q_{w'|w}}\right)$$

You should convince yourself that the last line ensures that the one before will true, and thus we have detailed balance.

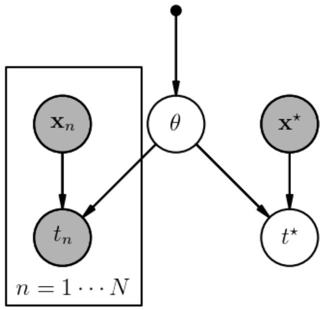
So that's the Metropolis Sampler.

With any MCMC method you need to have a burn-in period, and ensure that many Markov Chain samples separate each sample used for the Monte Carlo estimate.

MLP with weights we're uncertain about



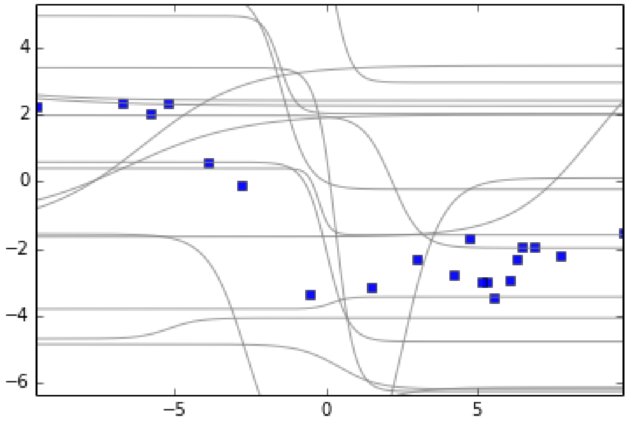
As a PGM (probabilistic graphical model)



We will use a Gaussian prior for $P(W)$, and get samples from the posterior $P(W|\mathcal{D})$ using Metropolis algorithm.

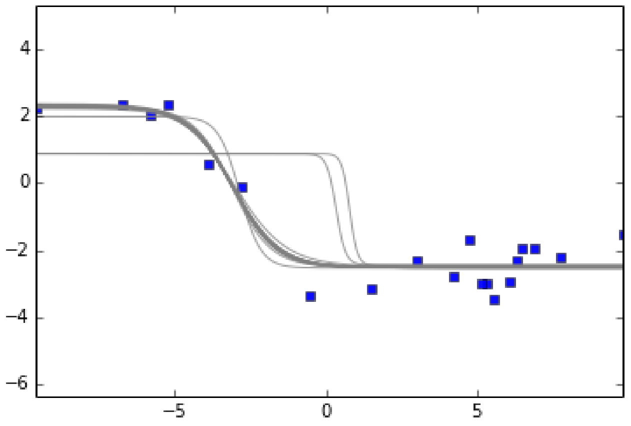
Prior

1 hidden units.



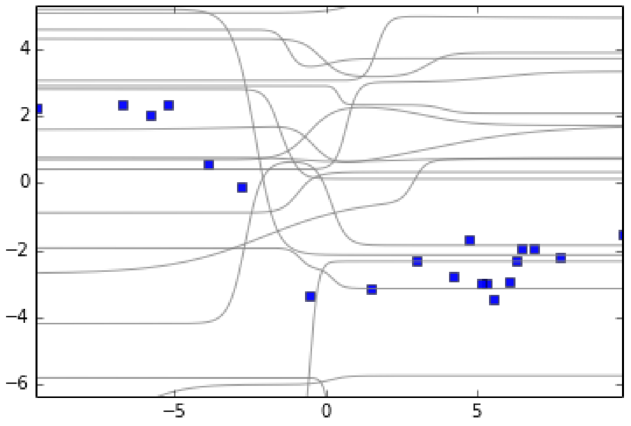
Posterior

1 hidden units.



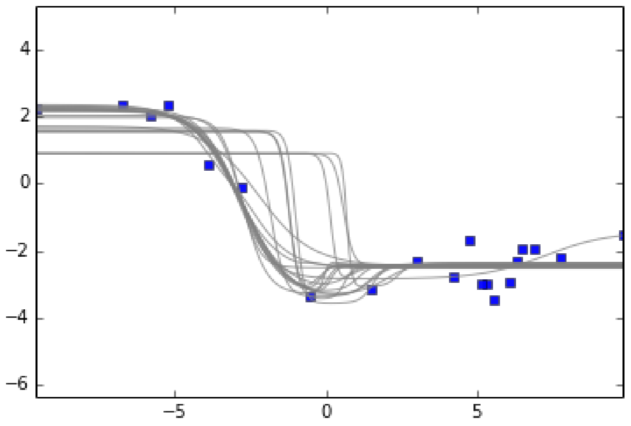
Prior

1 hidden units.



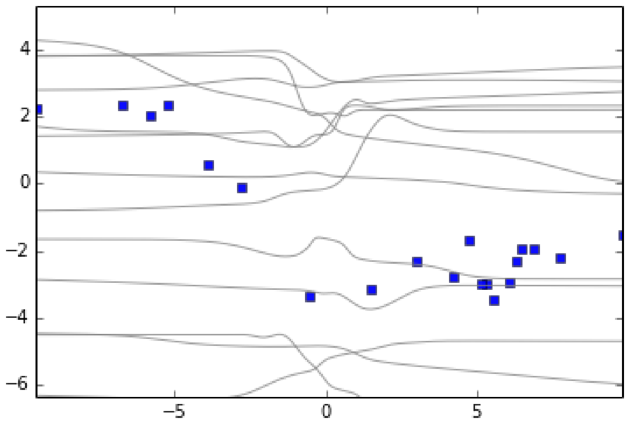
Posterior

1 hidden units.



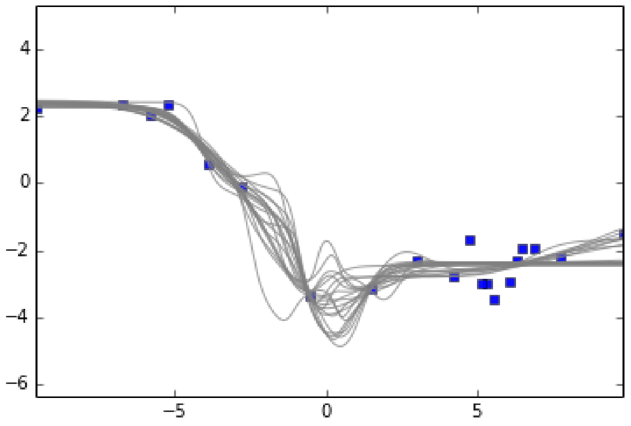
Prior

1 hidden units.



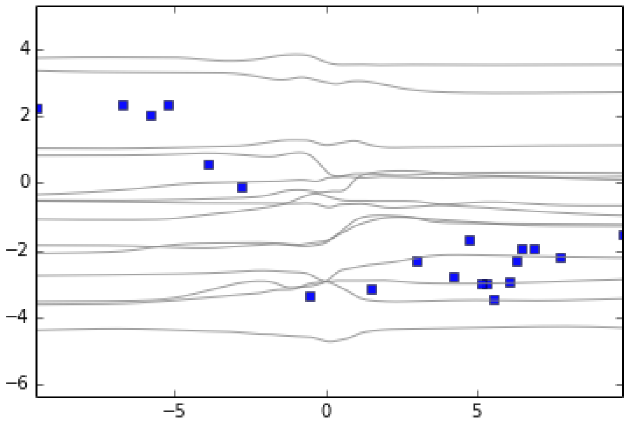
Posterior

1 hidden units.



Prior

50 hidden units.



Posterior

1 hidden units.