COMP 551 – Applied Machine Learning
Lecture 19: Bayesian Linear Regression

**Associate Instructor:** Ryan Lowe ([ryan.lowe@mail.mcgill.ca](mailto:ryan.lowe@mail.mcgill.ca))

**Slides mostly by:** Herke van Hoof ([herke.vanhoof@mcgill.ca](mailto:herke.vanhoof@mcgill.ca))

**Class web page:** [www.cs.mcgill.ca/~jpineau/comp551](http://www.cs.mcgill.ca/~jpineau/comp551)

Unless otherwise noted, all material posted for this course are copyright of the instructor, and cannot be reused or reposted without the instructor’s written permission.
Announcements

• **Assignment 2 grades are online!**
  • See TAs if you have questions about your grade
  • Will try to organize ‘joint office hour’ with all TAs who graded assignment 2 (will be announced)

• Project 4 Kaggle deadline March 21st!
  • **Report only** deadline extended 1 day, to March 22nd.
## Announcements

This leaderboard is calculated with approximately 30% of the test data. The final results will be based on the other 70%, so the final standings may be different.

<table>
<thead>
<tr>
<th>#</th>
<th>1w</th>
<th>Team Name</th>
<th>Kernel</th>
<th>Team Members</th>
<th>Score</th>
<th>Entries</th>
<th>Last</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+1</td>
<td>Sigma Mu</td>
<td></td>
<td></td>
<td>0.98399</td>
<td>20</td>
<td>18h</td>
</tr>
<tr>
<td>2</td>
<td>+5</td>
<td>hach.bo</td>
<td></td>
<td></td>
<td>0.98066</td>
<td>9</td>
<td>2d</td>
</tr>
<tr>
<td>3</td>
<td>+11</td>
<td>AI geeks</td>
<td></td>
<td></td>
<td>0.97666</td>
<td>23</td>
<td>1d</td>
</tr>
<tr>
<td>4</td>
<td>-3</td>
<td>KCM</td>
<td></td>
<td></td>
<td>0.97333</td>
<td>7</td>
<td>9d</td>
</tr>
<tr>
<td>5</td>
<td>-1</td>
<td>Team Biceps</td>
<td></td>
<td></td>
<td>0.97299</td>
<td>12</td>
<td>3h</td>
</tr>
<tr>
<td>6</td>
<td>-3</td>
<td>ASDFSWAG</td>
<td></td>
<td></td>
<td>0.96966</td>
<td>8</td>
<td>10d</td>
</tr>
<tr>
<td>7</td>
<td>+4</td>
<td>2 gb</td>
<td></td>
<td></td>
<td>0.96733</td>
<td>11</td>
<td>1d</td>
</tr>
<tr>
<td>8</td>
<td>new</td>
<td>spicyAway</td>
<td></td>
<td></td>
<td>0.96733</td>
<td>4</td>
<td>2d</td>
</tr>
<tr>
<td>9</td>
<td>-3</td>
<td>ApplicationMemoryError</td>
<td></td>
<td></td>
<td>0.96733</td>
<td>11</td>
<td>1h</td>
</tr>
<tr>
<td>10</td>
<td>+15</td>
<td>Happy Decision Tree Friends</td>
<td></td>
<td></td>
<td>0.96666</td>
<td>8</td>
<td>4h</td>
</tr>
</tbody>
</table>
Recall: Bayesian terminology

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

- **Likelihood** $p(D|w)$: our model of the data. Given our weights, how do we assign probabilities to dataset examples?
- **Prior** $p(w)$: before we see any data, what do we think about our parameters?
- **Posterior** $p(w|D)$: our distribution over weights, given the data we’ve observed and our prior
- **Marginal likelihood** $p(D)$: also called the normalization constant. Does not depend on $w$, so not usually calculated explicitly
Recall: Conjugate priors

- A prior $p(w)$ is **conjugate** to a likelihood function $p(D|w)$ if the posterior is in the same family as the prior.

- In other words, if prior * likelihood gives you the same form as the prior with different parameters, it’s a conjugate prior.

  - **Ex 1:** the Gaussian distribution is a conjugate prior to a Gaussian likelihood.

  - **Ex 2:** the Beta distribution is conjugate to a Bernoulli likelihood.

- **Why?** *Want simple form for our posterior!* Don’t want it to get more complicated every time you add more data.
Bayesian linear regression

- Previous examples (coin flip, learning the mean of a Gaussian) only had outputs $y$, no inputs $x$
- How can we learn to make predictions that are input-dependent?
- Can use an extension of linear regression: **Bayesian linear regression**
Given a dataset $\mathcal{D}$, how do we make predictions for a new input?

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$$

- **Step 1**: Define a model that represents your data (the **likelihood**): $p(\mathcal{D}|w)$
- **Step 2**: Define a **prior** over model parameters: $p(w)$
- **Step 3**: Calculate **posterior** using Bayes’ rule: 
  $$p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})}$$
- **Step 4**: Make **prediction** by integrating over model parameters:
  $$p(y^*|x^*, \mathcal{D}) = \int_{\mathbb{R}^N} p(w|\mathcal{D})p(y^*|x^*, w)dw$$
Bayesian linear regression

- We take a **specific form of the likelihood and the prior**:
  - **Step 1**: Likelihood
    \[ p(y|x, w) = \mathcal{N}(w^T x, \sigma^2) \]
  - **Step 2**: Conjugate prior
    \[ p(w) = \mathcal{N}(0, \alpha^{-1} I) \]
  - Prior precision \( \alpha \) and noise variance \( \sigma^2 \) considered known
  - Linear regression where we **learn a distribution over the parameters**
  - Output \( y \) close to learned linear function \( w^* x \), with some noise
  - Prefer small weights. (assuming no other info)
Visualizing inference

- Start with simple example (one feature x): \( y = w_0 + w_1 x + \epsilon \)

- How can we visualize what’s happening in Step 3? (finding \( p(w|\mathcal{D}) \) )

For different \( w_0, w_1 \), how likely is this data point?

How likely are different \( (w_0, w_1) \) given data so far?

Shows data points and sample functions for data so far
Visualizing inference

- **Goal:** fit lines\[ y = w_0 + w_1 x + \epsilon \]
- **Bayes theorem:**\[ p(w|D) = \frac{p(D|w)p(w)}{p(D)} \]
Visualizing inference

- Goal: fit lines \( y = w_0 + w_1 x + \epsilon \)

- Bayes theorem: \( p(w|D) = \frac{p(D|w)p(w)}{p(D)} \)

- Similar to ridge regression, expect good \( w \) to be small

Copyright C.M. Bishop, PRML
Goal: fit lines

\[ y = w_0 + w_1 x + \epsilon \]

Bayes theorem:

\[ p(w|\mathcal{D}) = \frac{p(\mathcal{D}|w)p(w)}{p(\mathcal{D})} \]

Similar to ridge regression, expect good \( w \) to be small
Visualizing inference

- Goal: fit lines
  \[ y = w_0 + w_1 x + \epsilon \]

- Bayes theorem:
  \[ p(w|D) = \frac{p(D|w)p(w)}{p(D)} \]

- Similar to ridge regression, expect good \( w \) to be small
Visualizing inference

- Goal: fit lines $y = w_0 + w_1 x + \epsilon$

- Bayes theorem: $p(w|D) = \frac{p(D|w)p(w)}{p(D)}$

- Similar to ridge regression, expect good $w$ to be small

Copyright C.M. Bishop, PRML
Visualizing inference

- Goal: fit lines
  \[ y = w_0 + w_1 x + \epsilon \]

- Bayes theorem:
  \[ p(w|D) = \frac{p(D|w)p(w)}{p(D)} \]

- Good lines should pass ‘close by’ datapoint

Copyright C.M. Bishop, PRML
Visualizing inference

• Goal: fit lines $y = w_0 + w_1x + \epsilon$

• Bayes theorem: $p(w|D) = \frac{p(D|w)p(w)}{p(D)}$

• Good lines should pass ‘close by’ datapoint
Visualizing inference

- Goal: fit lines
  \[ y = w_0 + w_1 x + \epsilon \]

- Bayes theorem:
  \[ p(w|D) = \frac{p(D|w)p(w)}{p(D)} \]

- For all values of \( w \), multiply prior and likelihood (and re-normalize)
Bayesian linear regression: inference

Copyright C.M. Bishop, PRML
Bayesian linear regression: inference

Copyright C.M. Bishop, PRML
Bayesian linear regression: inference

Copyright C.M. Bishop, PRML
Bayesian linear regression: inference

Copyright C.M. Bishop, PRML
Bayesian linear regression: inference
Bayesian linear regression: inference

Copyright C.M. Bishop, PRML
Bayesian linear regression: inference

As new data points are added, posterior converges on true value of parameters
Step 3: calculate posterior

- Can calculate posterior by multiplying prior and likelihood:

\[ p(w|D) = \mathcal{N}(\sigma^{-2}S_N X^T y, S_N) \]

\[ S_N = (\alpha I + \sigma^{-2}X^T X)^{-1} \]

- \( X \) has one input per row, \( y \) has one target output per row
- If prior precision \( \alpha \) goes to 0, mean becomes maximum likelihood solution (ordinary linear regression)
- Infinitely wide likelihood variance \( \sigma^2 \), or 0 datapoints, means distribution reduces to prior

(derivation similar to case with no inputs — slide 30 of lecture 18)
Aside: finding the MAP

- We can investigate the maximum of the posterior (MAP)
- Log-transform posterior: log is sum of prior + likelihood

\[
\max \log p(w|y) = \max -\frac{\sigma^{-2}}{2} \sum_{n=1}^{N} (y_n - w^T x_n)^2 - \frac{\alpha}{2} w^T w + \text{const.}
\]
Aside: finding the MAP

• We can investigate the maximum value of the posterior (MAP)

• Calculate in log space: \( \log \text{posterior} = \log \text{prior} + \log \text{likelihood} \)

\[
\max \log p(w|y) = \max -\frac{\sigma^{-2}}{2} \sum_{n=1}^{N} (y_n - w^T x_n)^2 - \frac{\alpha}{2} w^T w + \text{const.}
\]

Recall:
\[
\min \sum_{n=1}^{N} (y_n - w^T x_n)^2 + \lambda w^T w
\]
Ridge regression, Lecture 4 (linear regression)

• Same objective function as for ridge regression!

• Penalty term: \( \lambda = \frac{\alpha \sigma^2}{\text{prior precision}} \) likelihood variance

Note: since posterior is Gaussian, MAP = mean of posterior
Step 4: prediction

• Prediction for new datapoint:

\[ p(y^* | x^*, D) = \int_{\mathbb{R}^N} p(w|D)p(y^* | x^*, w)dw \]

• For Gaussians, can compute solution analytically:

\[ p(y^* | D) = \mathcal{N}(\sigma^{-2}x^*^T S_N X^T y, \sigma^2 + x^T S_N x) \]

- mean from before
- from weight uncertainty
- new input
- from observation noise

• Variance tends to go down with more data until it reaches \( \sigma^2 \)
Step 4: prediction

- Every $\mathbf{w}$ makes a prediction, weighted by posterior

\[
\int_{\mathbb{R}^{N}} p(\mathbf{w}|\mathcal{D})p(y^*|\mathbf{x}^*, \mathbf{w})d\mathbf{w} = \text{medium} \quad p(\mathbf{w}|\mathcal{D})
\]
Step 4: prediction

- Every \( w \) makes a prediction, weighted by posterior

\[
\int_{\mathbb{R}^N} p(w|D)p(y^*|x^*, w) \, dw
\]

\( x \) medium \( p(w|D) \)

\( x \) small \( p(w|D) \)
Step 4: prediction

- Every \( \mathbf{w} \) makes a prediction, weighted by posterior

\[
p(\mathbf{w} | \mathcal{D}) = \int_{\mathbb{R}^N} p(\mathbf{w} | \mathcal{D}) p(y^* | \mathbf{x}^*, \mathbf{w}) d\mathbf{w}
\]

\( x \) medium
\[
p(\mathbf{w} | \mathcal{D})
\]

\( + \)
\[
p(\mathbf{w} | \mathcal{D})
\]

\( + \)
\[
p(\mathbf{w} | \mathcal{D})
\]

\( + \) many other models

\[
p(\mathbf{w} | \mathcal{D})
\]
Step 4: prediction

- Every $\mathbf{w}$ makes a prediction, weighted by posterior

$p(\mathbf{w} | D)$

\[
p(y^*, \mathbf{x}^* | D) = \int p(y^*, \mathbf{x}^*, \mathbf{w} | D) d\mathbf{w} = \int p(y^* | \mathbf{w}, \mathbf{x}^* \ D) p(\mathbf{w}|D) p(D) \\
\]

\[
p(\mathbf{w}|D) = \frac{p(D|\mathbf{w}) p(\mathbf{w})}{p(D)}
\]

Each vertical line defines $p(y^* | \mathbf{x}^*, D)$ for that $\mathbf{x}^*$
Bayesian linear regression

- Like ordinary linear regression, can use non-linear basis

\[ f_w(x) = w_0 + w_1 x + w_2 x^2 \]

Copyright J. Pineau

Lecture 4, linear regression
Bayesian linear regression

- Like ordinary linear regression, can use non-linear basis

\[ \hat{y} = \sum_{i=1}^{M} w_i \phi_i(x) \]
Bayesian linear regression: polynomial bases

- Example: Bayesian linear regression with polynomial bases

- **Green line:** true function.  **Blue circles:** data points.
- **Red line:** MAP prediction.  **Shaded red:** posterior predictive distribution.
Bayesian linear regression: polynomial bases
Bayesian linear regression: polynomial bases
Recall: inspection task example

Could produce this graph using Bayesian linear regression
Beyond linear regression

- Non-linear data sets can be handled by using non-linear features
- Features specify the class of functions we consider (hypothesis class)
  \[ \hat{y} = \sum_{i=1}^{M} w_i \phi_i(x) \]
- What if we do not know good features?
- Some features (polynomial, RBF) work for many problems
Application: black-box optimization

- **Problem:** find value x for which function f(x) is maximized

- **Constraints:**
  - f(x) is a ‘black box’ function: we only know the value f(x) for small set of points x that we evaluate
  - Evaluating f(x) is relatively expensive
  - f(x) might have local optima
  - Derivatives might not be known

- **Example:** finding the hyperparameters of a neural network

- **How can we approach this problem?**
Black-box optimization

- **Problem:** find value x for which function f(x) is maximal
- Example of black box function

![Diagram showing the relationship between learning rate and final neural network performance](Image)

Final neural network performance

learning rate
So far, we have mainly done gradient ascent

But gradient ascent requires an estimate of the gradient

- Might need many function evaluations (costly)
- Can get stuck in local minima

Can we do better?
Black-box optimization

- How might a problem look like?
- Where to sample next, if we have a budget for, say, 10 samples?

points that were already evaluated

$f(x)$

input variable $x$
Black-box optimization

• How might a problem look like?

we could sample here, might be near local maximum but here we know very little, could help find better solutions later

\[ f(x) \]

input variable \( x \)
Black-box optimization

• How might a problem look like?
• How about now?

input variable $x$

$f(x)$
Bayesian optimization

• **Idea:** to make a good decision we should *imagine what the whole function should look like*

• It seems important to take into account how certain we are for various input values $x$

• **Bayesian linear regression** might do the job here!

• This implies Bayesian point of view: **Bayesian optimisation** (a method to do black-box optimization)
Bayesian optimisation

- Bayesian posterior over function

- Where to sample next?

previous function evaluations
Bayesian optimisation

- Where to sample next?
- What happens if we simply sample where mean is highest?
Bayesian optimisation

- We don’t sample on the right at all!
- We might miss the real maximum
Bayesian optimisation

• Where to sample next?

• Two objectives:
  
  • **Exploitation**: sample where we think high values are
    If we know the samples will be low, it does not make sense to sample there
    Maybe: sample highest mean?
  
  • **Exploration**: If we always sample where we think the highest value is, we might miss other values
    Maybe: sample where uncertainty is highest?
Bayesian optimisation

- Several strategies exist for combining these two objectives
- Can give ‘score’ to possible examples using acquisition function
- Very straightforward method: upper confidence bound (UCB)

\[ a_{UCB}(x^*; D) = \mu(x^*; D) + \kappa \sigma(x^*; D) \]

- Acquisition functions gives a ‘score’ to each sample point
- UCB has good theoretical properties
Bayesian optimisation

- Upper confidence bound acquisition function

UCB acquisition function ($\kappa=2$)

Maximum of acquisition function
Bayesian optimisation

- Upper confidence bound acquisition function

first sample  second sample  third sample
Bayesian optimisation

- We now explore sufficiently well to get close to the maximum

fourth sample

fifth sample, comparison to true function
Bayesian optimisation

• Different acquisition functions exist:
  • Probability of improvement
    • Probability sampled value > current maximum?
    • Sometimes too greedy
  • Expected improvement
    • Weights probability with amount of improvement
    • Can be overly greedy
  • Upper confidence bound
    • Strong theoretical properties
    • Need to set tuning parameter $\kappa$
Bayesian optimisation

- **Pros**
  - Attempt at global optimisation
  - Need relatively few samples to get close to optimum
  - Software packages available

- **Cons**
  - Computational expensive
    - Need to fit a model and hyperparameters in every iteration
    - Need to maximise non-convex acquisition function
  - Sensitive to choice of model
  - Only works well with few input (up to ~10 dimensions)
Bayesian hyperparameter optimisation

• One application of Bayesian optimisation is hyperparameter optimisation

• Example: Tune learning rate in deep neural net
  • Nonconvex function with local optima
  • Evaluating a learning rate is expensive: we must train the network with that rate to know how good it is
Inference vs. Learning

- Different (overlapping!) communities use different terminology, can be confusing

- In *traditional machine learning*:
  - **Learning**: adjusting the parameters of your model to fit the data (by optimization of some cost function)
  - **Inference**: given your model + parameters and some data, make some prediction (e.g. the class of an input image)

- In *Bayesian statistics*, inference is to say something about the process that generated some data (*includes parameter estimation*)

- *Take-away*: in an ML problem, we can find a good value of params by optimization (*learning*) or calculate a distribution over params (*inference*)
Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
  - Overfitting in e.g. linear regression models
  - MLE of coin flip probabilities with three sequential ‘heads’
Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
- **We might desire or need an estimate of uncertainty**
  - Can use uncertainty in decision making
  - Can use uncertainty to decide which data to acquire (active learning, experimental design)
Why Bayesian probabilities?

• Maximum likelihood estimates can have large variance
• We might desire or need an estimate of uncertainty
• **Have small dataset, unreliable data, or small batches of data**
  • Account for reliability of different pieces of evidence
  • Possible to update posterior incrementally with new data
  • Variance problem especially bad with small data sets
Why Bayesian probabilities?

- Maximum likelihood estimates can have large variance
- We might desire or need an estimate of uncertainty
- Have small dataset, unreliable data, or small batches of data
- Use prior knowledge in a principled fashion
Why Bayesian probabilities?

• Maximum likelihood estimates can have large variance
• We might desire or need an estimate of uncertainty
• Have small dataset, unreliable data, or small batches of data
• Use prior knowledge in a principled fashion
• In practice, using prior knowledge and uncertainty particularly makes difference with small data sets
Why not Bayesian probabilities?

- Prior induces bias
- Misspecified priors: if prior is wrong, posterior can be far off
- Prior often chosen for mathematical convenience, not actually knowledge of the problem
- In contrast to frequentist probability, uncertainty is subjective, different between different people / agents
Beyond linear regression

- Relying on features can be problematic
- We tried to avoid using features before...
  - Lecture 8, instance based learning. Use distances!
  - Lecture 12, support vector machines. Use kernels!

- **Next class**: extend regression to nonparametric models
  - *Gaussian processes!*
What you should know

• Bayesian terminology (prior, posterior, likelihood, etc.)
• Conjugate priors, what they mean, showing a distribution is a conjugate prior
• Bayesian linear regression and its properties
• When and why to use Bayesian methods
• Core concepts behind Bayesian optimization