

A Brief Intro to Bayesian Inference



Thomas Bayes (c. 1701 – 1761)

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

Tossing a Coin


- Suppose the coin came up Heads 65 times and Tails 35 times. Is the coin fair?
- Model: $P(\text{heads}) = \theta$
- Log-likelihood: $\log P(\text{data}|\theta) = 65 \log \theta + 35 \log(1 - \theta)$
 - Maximized at $\theta = .65$
- But would you conclude that the coin really is not fair?

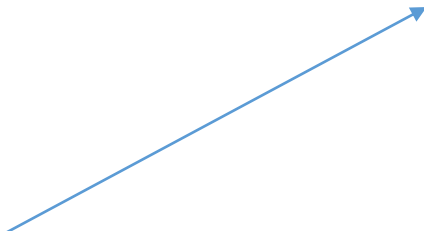
Prior Distributions

- We can encode our beliefs about what the values of the parameters could be using

$$P(\theta)$$

- Using Bayes' rule, we have

$$P(\theta = \theta_0 | \text{data}) = \frac{P(\theta = \theta_0, \text{data})}{P(\text{data})} = \frac{P(\text{data} | \theta = \theta_0) P(\theta = \theta_0)}{P(\text{data})}$$


$$= \sum_{\theta_1} P(\text{data} | \theta = \theta_1) P(\theta = \theta_1)$$


Maximum a-posteriori (MAP)

- Maximize the *posterior probability* of the parameter:

$$\operatorname{argmax}_{\theta_0} \frac{P(\text{data}|\theta = \theta_0)P(\theta = \theta_0)}{P(\text{data})}$$

$$= \operatorname{argmax}_{\theta_0} P(\text{data}|\theta = \theta_0)P(\theta = \theta_0)$$

$$= \operatorname{argmax}_{\theta_0} \log P(\text{data}|\theta = \theta_0) + \log P(\theta = \theta_0)$$

- The posterior of probability is the product of the prior and the data likelihood
- Represents the *updated* belief about the parameter, given the observed data

Aside: Bayesian Inference is a Powerful Idea

- You can think about anything like that. You have your prior belief $P(\theta)$, and you observe some new data. Now your belief about θ *must be* proportional to $P(\theta)P(data|\theta)$
 - But only if you are 100% sure that the likelihood function is correct!
 - Recall that the likelihood function is your model of the world – it represents knowledge about how the data is generated for given values of θ
 - Where do you get your original prior beliefs anyway?
- Arguably, makes more sense than Maximum Likelihood

Back to the Coin

- (In Python)

Gaussian Residuals Models

- Log-likelihood:

$$\log P(\text{data}|\theta) = \sum -\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2} - \frac{m}{2} \log(2\pi\sigma^2)$$

- Suppose we believe that $P(\theta_i) = N\left(0, \left(\frac{1}{2\lambda}\right)\right)$
 - I.e., the coefficients in θ will generally be in $[-1.5/\lambda, 1.5/\lambda]$
- $\log[P(\text{data}|\theta)P(\theta)]$ is $\log P(\text{data}|\theta) - \lambda|\theta|^2 + \text{const}$
(exercise)
- Maximize $\log[P(\text{data}|\theta)P(\theta)]$ to get the θ that you believe the most

$$\text{Why } P(\theta_i) = N\left(0, \left(\frac{1}{2\lambda}\right)\right)$$

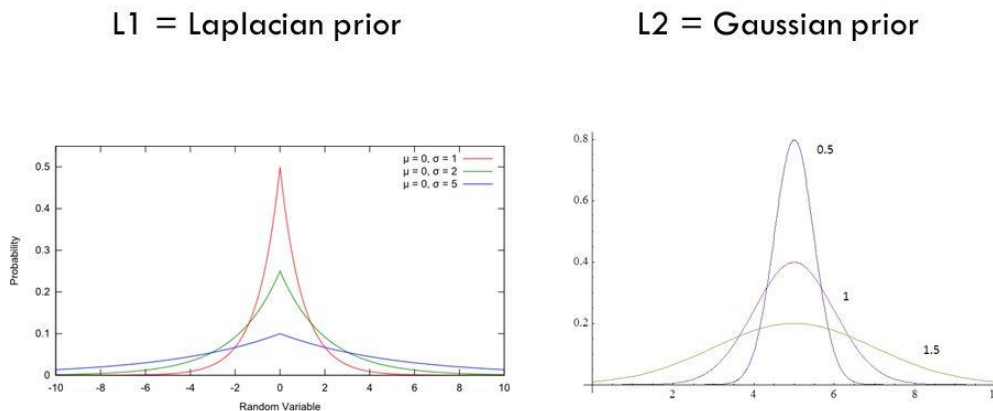
- More on this later
- If the θ_i 's are allowed to be arbitrarily large, the ratio of the influences of the different features over the decision boundary could be arbitrarily high
 - Difficult to believe that one of the features still matters, but it matters a 10000000 times less than some other feature
 - Easy to believe a feature doesn't matter at all, though
 - Only reasonable if the inputs are all on the same scale, and the output is on roughly the same scale as the inputs
 - Mostly when we fit coefficients, they don't get crazy high, so it's a reasonable prior belief

L2 Regularization

- L2 regularization:
maximize: $\log P(\text{data}|\theta) - \lambda|\theta|^2 + \text{const}$
- “L2 regularization” because numerically, the cost function penalizes the L2 norm of θ
- A way of preventing overfitting
 - If we set λ to be very high, θ will just be 0: the performance on the training and test sets will be the same (and will be bad)
 - If we set λ to be moderately high, we won't let θ_i 's be too large even if that leads to good performance on the training set. Idea: if the training set makes a θ_i very large, that probably won't be good for test set performance, since usually large θ_i 's lead to poor performance
- What about other norms?

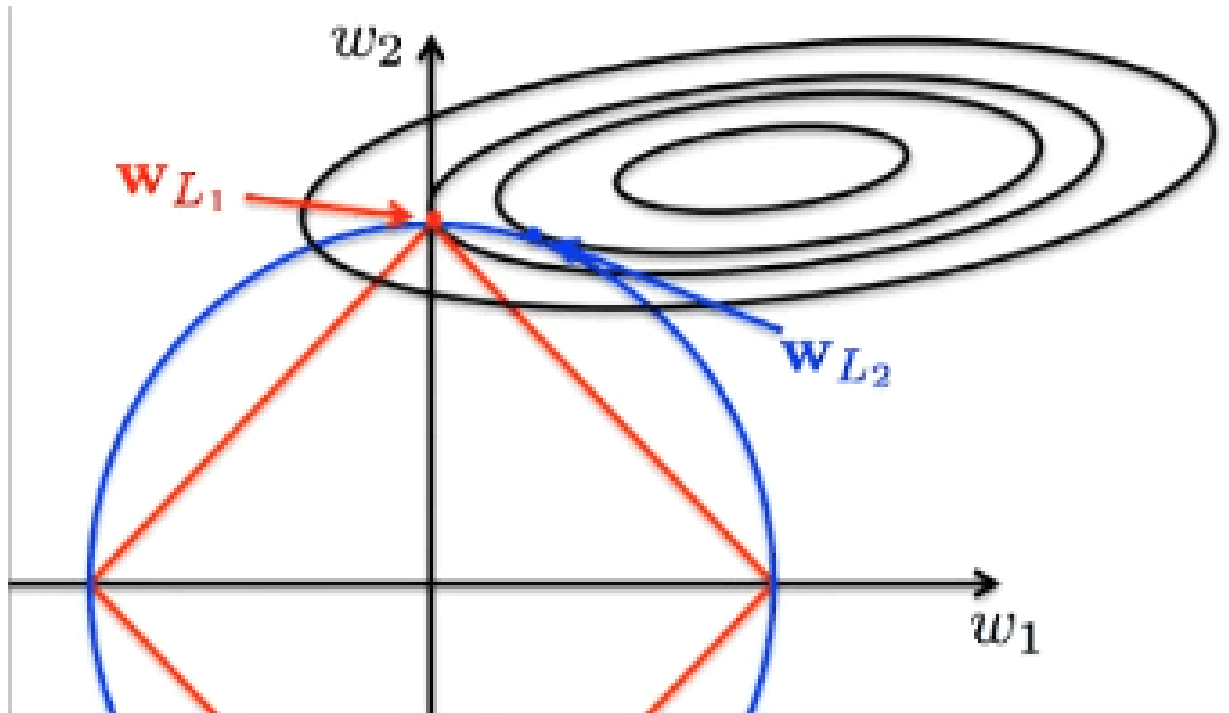
L1 Regularization

- Alternative: L1 regularization:
maximize: $\log P(\text{data}|\theta) - \lambda|\theta|_1 + \text{const}$
- Equivalent to using a Laplacian prior:



- Encourages sparsity (feature selection)
 - Sparsity: most θ_i are zero

L2 vs L1 regularization



L2 vs L1 regularization

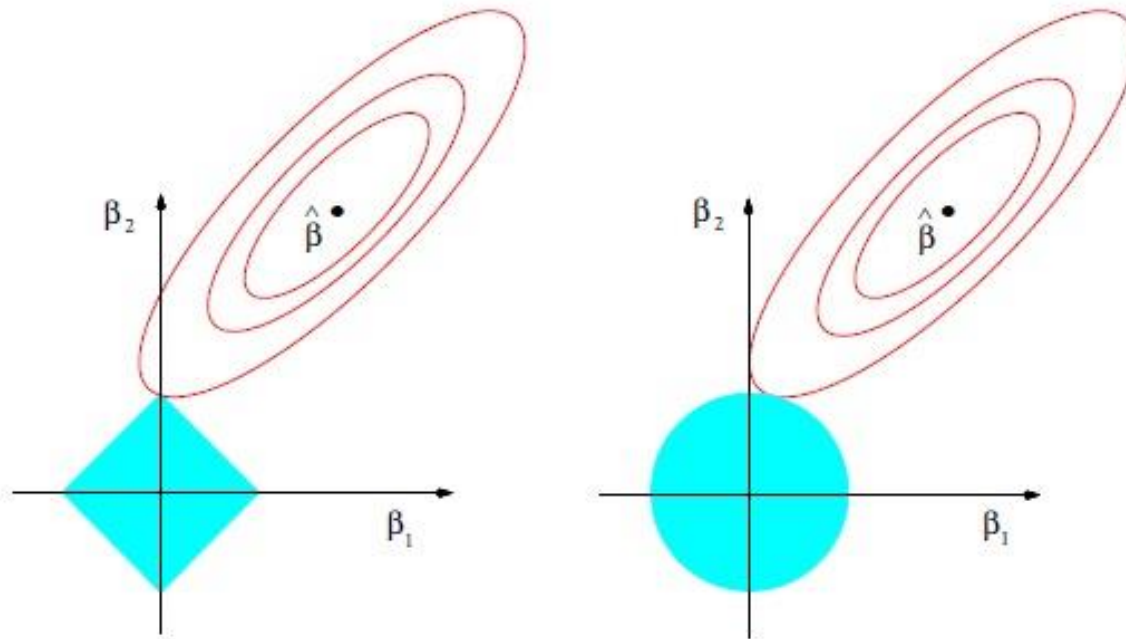


FIGURE 3.11. Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions $|\beta_1| + |\beta_2| \leq t$ and $\beta_1^2 + \beta_2^2 \leq t^2$, respectively, while the red ellipses are the contours of the least squares error function.

Early Stopping

- Initialize the θ s to small initial values
- Run Gradient Descent, but stop early
 - Before finding the minimum of the cost function applied to the training set
 - More on this later this week
- Can be shown to be equivalent of L2 regularization (under certain assumptions)
- Why does it work?