Lecture 6 - Bayesian Interpretation, Maximum Likelihood Estimation (MLE) & Maximum A Posteriori (MAP)

Bayesian Interpretation

Bayesian Interpretation of the Least Squares Estimator ($\hat{\mathbf{w}}$)

We have seen some definitions of common estimators and analyzed their properties. But where did these estimators come from? Rather than guessing that some function might make a good estimator and then analyzing its bias and variance, we would like to have some principle from which we can derive specific functions that are good estimators for different models.

Let's try to understand this better. Consider the objective function:

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(t_i - y(\phi(x_i), \mathbf{w})\right)^2$$

where $\mathbf{y}(\phi(\mathbf{x}), \mathbf{w})$ is a model representation (e.g., linear regression), and $\phi(x)$ is a feature mapping function (e.g. Gaussian Basis functions).

The most common such principle is the **maximum likelihood** principle. Consider a set of N examples $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$ drawn independently from the true but unknown data-generating distribution $p_{\text{data}}(x)$.

Let $p_{\text{model}}(\phi(x); \mathbf{w})$ be a parametric family of probability distributions over the same space indexed by $\mathbf{w}$. In other words, $p_{\text{model}}(\phi(x); \mathbf{w})$ maps any configuration $\phi(x)$ to a real number estimating the true probability $p_{\text{data}}(\phi(x))$.

The maximum likelihood estimator for $\mathbf{w}$ is then defined as:

$$\begin{aligned}\mathbf{w}_{\text{MLE}} &= \arg_{\mathbf{w}} \max p_{\text{model}}(\phi(\mathbf{x}); \mathbf{w}) \\ &= \arg_{\mathbf{w}} \max \prod_{i=1}^{N} p_{\text{model}}(\phi(x_i); \mathbf{w})\end{aligned}$$

This product over many probabilities can be inconvenient for various reasons. For example, it is prone to numerical underflow. To obtain a more convenient but equivalent

optimization problem, we observe that taking the logarithm of the likelihood does not change its arg max but does conveniently transform a product.

$$\mathbf{w}_{\text{MLE}} = \arg_{\mathbf{w}} \max \sum_{i=1}^{N} \log p_{\text{model}}(\phi(x_i); \mathbf{w})$$

Because the arg max does not change when we rescale the cost function, we can divide by N to obtain a version of the criterion that is expressed as an expectation with respect to the empirical distribution p_{data} defined by the training data:

$$\mathbf{w}_{\text{MLE}} = \arg_{\mathbf{w}} \max \mathbb{E}_{\phi(x) \sim \hat{p}_{\text{data}}} [\log p_{\text{model}}(\phi(x_i); \mathbf{w})]$$

Log-Likelihood and the Least Squares Solution

The maximum likelihood estimator can readily be generalized to estimate a conditional probability $P(\mathbf{t}|\phi(\mathbf{x}); \mathbf{w})$ in order to predict t given $\phi(x)$. This is actually the most common situation because it forms the basis for most supervised learning. If $\mathbf{x}$ represents all our inputs and $\mathbf{t} all our observed targets, then the conditional maximum likelihood estimator is

$$\mathbf{w}_{\text{MLE}} = \arg_{\mathbf{w}} \max P(\mathbf{t}|\phi(\mathbf{x}); \mathbf{w})$$

If the examples are assumed to be i.i.d., then this can be decomposed into

$$\mathbf{w}_{\text{MLE}} = \arg_{\mathbf{w}} \max \sum_{i=1}^{N} \log P(t_i|\phi(x_i); \mathbf{w})$$

Least Squares Solution of Linear Regression as the Maximum Likelihood Estimation (MLE) of $\mathbf{w}$

Linear regression with the least squares objective function may be justified as a maximum likelihood procedure. Previously, we motivated linear regression as an algorithm that learns to take a feature input $\phi(x)$ and produce an output value y. The mapping from $\phi(x)$ to y is chosen to minimize the mean squared error, a criterion that we introduced more or less arbitrarily.

We now revisit linear regression from the point of view of maximum likelihood estimation. Instead of producing a single prediction y, we now think of the model as producing a conditional distribution $p(\mathbf{t}|\phi(\mathbf{x}))$. We can imagine that with an infinitely large training set, we might see several training examples with the same input value $\phi(\mathbf{x})$ but different values of $\mathbf{t}$. The goal of the learning algorithm is now to fit the distribution $p(t|\phi(x))$ to

all those different t values that are all compatible with $\phi(x)$. To derive the same linear regression algorithm we obtained before, we define $p(t|\phi(x)) = \mathcal{N}\left(t; f(\phi(\mathbf{x}); \mathbf{w}), \sigma^2\right)$. The function $f(\phi(\mathbf{x}); \mathbf{w})$ gives the prediction of the mean of the Gaussian.

In this example, we assume that the variance is fixed to some constant $sigma^2$ chosen by the user. We will see that this choice of the functional form of $p(\mathbf{t}|\phi(\mathbf{x}))$ causes the maximum likelihood estimation procedure to yield the same learning algorithm as we developed before. Since the examples are assumed to be i.i.d., the conditional log-likelihood is given by

$$\mathcal{L} = \log p(t_i|\phi(x_i); \mathbf{w}) = -N \log \sigma - \frac{N}{2}\log(2\pi) - \sum_{i=1}^{N} \frac{(t_i - y_i)^2}{2\sigma^2}$$

Complete Derivation

Let's see how we can arrive at this same solution from a different perpective:

$$\begin{aligned}
\arg_{\mathbf{w}} \min J(\mathbf{w}) &= \arg_{\mathbf{w}} \max -J(\mathbf{w}) \\
&= \arg_{\mathbf{w}} \max \exp(-J(\mathbf{w})) \\
&= \arg_{\mathbf{w}} \max \exp\left(-\frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2\right) \\
&= \arg_{\mathbf{w}} \max \prod_{i=1}^{N} \exp\left(-\frac{1}{2}(t_i - y_i)^2\right) \\
&\propto \arg_{\mathbf{w}} \max \prod_{i=1}^{N} \mathcal{N}\left(t_i; y_i, 1\right)
\end{aligned}$$

Recall that the univariate Gaussian pdf is defined as:
$\mathcal{N}(x; \mu, \sigma^2) \sim \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right)$.

Applying our "trick", the natural logarithm:

$$\begin{aligned}
&\arg_{\mathbf{w}} \max \prod_{i=1}^{N} \mathcal{N}\left(t_i; y_i, 1\right) \\
\propto &\arg_{\mathbf{w}} \max \ln \prod_{i=1}^{N} \mathcal{N}\left(t_i; y_i, 1\right) \\
= &\arg_{\mathbf{w}} \max \sum_{i=1}^{N} \ln \mathcal{N}\left(t_i; y_i, 1\right)
\end{aligned}$$

We can expand the last term as:

$$\begin{aligned}\sum_{i=1}^{N} \ln \mathcal{N}\left(t_i; y_i, 1\right) &= \sum_{i=1}^{N} \ln\left(\frac{1}{\sqrt{2\pi}} \exp\left\{-\frac{1}{2}(t_i - y_i)^2\right\}\right) \\ &= \sum_{i=1}^{N} -\frac{1}{2}\ln(2\pi) - \frac{1}{2}(t_i - y_i)^2 \\ &= -\frac{N}{2} - \sum_{i=1}^{N} \frac{1}{2}(t_i - y_i)^2\end{aligned}$$

The next step to find the arg $\mathbf{w}$ that maximizes this log-likelihood is to take the derivative of this function with respect to (w.r.t.) $\mathbf{w}$, set it to 0 and solve for $\mathbf{w}$.

If we consider the special case of a linear regression model, we have:
$y_i = \sum_{j=0}^{M} w_j x_i^j = \mathbf{w}^T \phi(x_i)$, where $\phi(x)$ is the polynomial basis function. Hence:

$$\begin{aligned}\mathcal{L} &= -\frac{N}{2} - \sum_{i=1}^{N} \frac{1}{2}(t_i - \mathbf{w}^T \phi(x_i))^2 \\ &= -\frac{N}{2} - \frac{1}{2}\|\mathbf{t} - \mathbf{X}\mathbf{w}\|_2^2\end{aligned}$$

As before, if we take the derivative

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0$$

We will arrive at the same solution:

$$\mathbf{w}_{\text{MLE}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{t}$$

Ridge Regression as Maximum A Posteriori (MAP)

How can we interpret the optimization problem when we consider a regularization term (ridge or lasso) in the objective function?

Consider

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(t_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=0}^{M} w_j^2$$

As before, we have

$$
\begin{aligned}
\arg_{\mathbf{w}} \min J(\mathbf{w}) &= \arg_{\mathbf{w}} \max -J(\mathbf{w}) \\
&= \arg_{\mathbf{w}} \max \exp(-J(\mathbf{w})) \\
&= \arg_{\mathbf{w}} \max \exp\left(-\frac{1}{2} \sum_{i=1}^{N} (t_i - y_i)^2 - \frac{\lambda}{2} \sum_{j=0}^{M} w_j^2 \right) \\
&= \arg_{\mathbf{w}} \max \prod_{i=1}^{N} \exp\left(-\frac{1}{2} (t_i - y_i)^2 \right) \prod_{j=0}^{M} \exp\left(-\frac{\lambda}{2} w_j^2 \right) \\
&\propto \arg_{\mathbf{w}} \max \prod_{i=1}^{N} \mathcal{N}\left(t_i; y_i, 1\right) \prod_{j=0}^{M} \mathcal{N}\left(w_j; 0, \frac{1}{\lambda}\right) \\
&= \arg_{\mathbf{w}} \max p(\mathbf{t}|\mathbf{y}(\mathbf{x};\mathbf{w}))p(\mathbf{w}|\lambda) \\
&\propto \arg_{\mathbf{w}} \max p(\mathbf{w}|\mathbf{t})
\end{aligned}
$$

We see that adding a regularization penalty term to the objective function is equivalent to adding a prior probability on the parameters.

For the ridge penalty, the probabilistic model of the prior probability is a Gaussian distribution with mean 0 and variance $1/\lambda$.

Lasso Regression as Maximum A Posteriori (MAP)

Consider

$$
J(\mathbf{x};\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (t_i - y_i)^2 + \frac{\lambda}{2} \sum_{j=0}^{M} |w_j|
$$

As before, we have

$$
\begin{aligned}
\arg_{\mathbf{w}} \min J(\mathbf{w}) &= \arg_{\mathbf{w}} \max -J(\mathbf{w}) \\
&= \arg_{\mathbf{w}} \max \exp(-J(\mathbf{w})) \\
&= \arg_{\mathbf{w}} \max \exp\left(-\frac{1}{2} \sum_{i=1}^{N} (t_i - y_i)^2 - \frac{\lambda}{2} \sum_{j=0}^{M} |w_j| \right) \\
&= \arg_{\mathbf{w}} \max \prod_{i=1}^{N} \exp\left(-\frac{1}{2} (t_i - y_i)^2 \right) \prod_{j=0}^{M} \exp\left(-\frac{\lambda}{2} |w_j| \right) \\
&\propto \arg_{\mathbf{w}} \max \prod_{i=1}^{N} \mathcal{N}\left(t_i; y_i, 1\right) \prod_{j=0}^{M} \mathcal{L}\text{aplacian}\left(w_j; 0, \frac{1}{\lambda}\right) \\
&= \arg_{\mathbf{w}} \max p(\mathbf{t}|\mathbf{y}(\mathbf{x};\mathbf{w}))p(\mathbf{w}|\lambda) \\
&\propto \arg_{\mathbf{w}} \max p(\mathbf{w}|\mathbf{t})
\end{aligned}
$$

For the lasso penalty, the probabilistic model of the prior probability is a Laplacian distribution with parameters $\mu = 0$ and $b = 1/\lambda$.

Recall that the Laplacian pdf has two parameters, μ and b ($b > 0$), and its function is defined as: $\mathcal{L}\text{aplacian}(\mu, b) \sim \frac{1}{2b}\exp\left\{-\frac{|x-\mu|}{b}\right\}$.

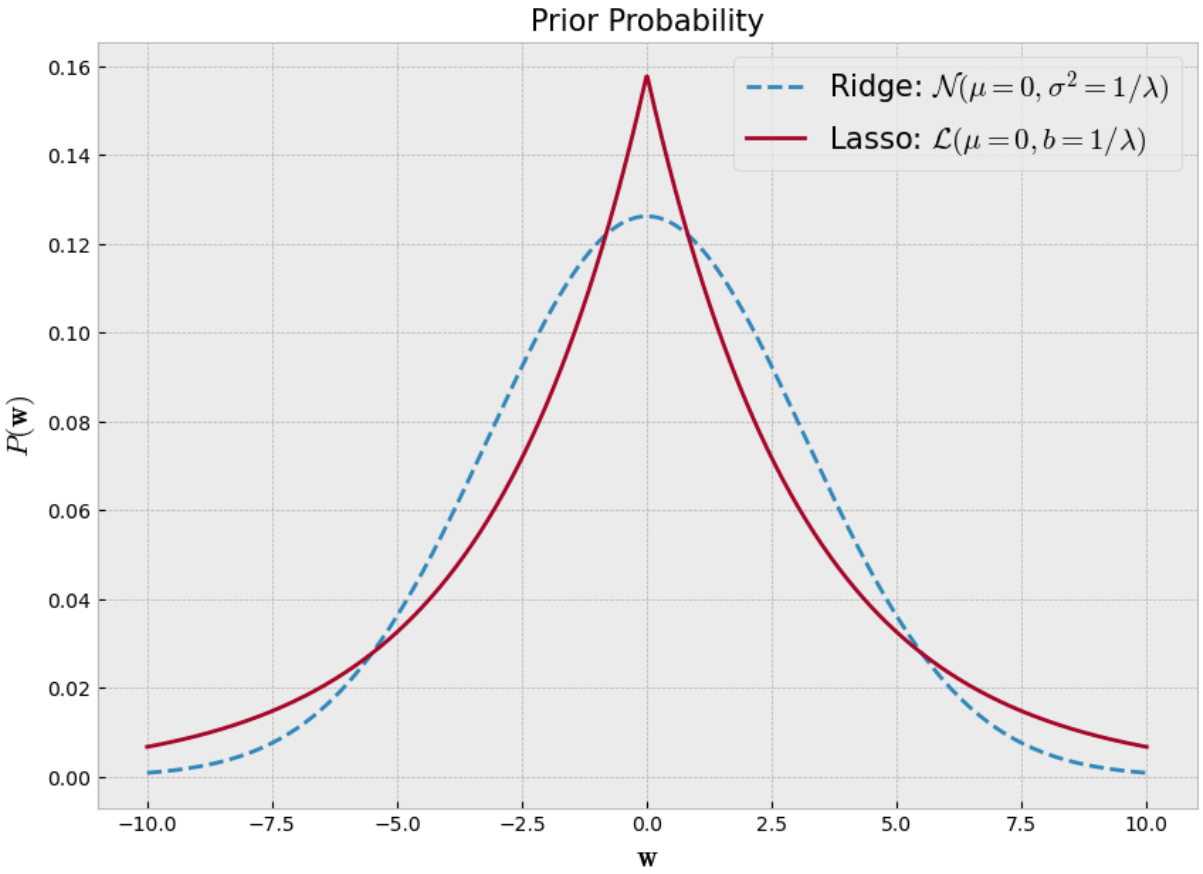
In [1]:
```
import math
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')
import scipy.stats as stats
```

In [6]:
```
x = np.linspace(-10,10,1000)

lam = 0.1

G = stats.norm(0, np.sqrt(1/lam))
L = stats.laplace(0, np.sqrt(1/lam))

plt.figure(figsize=(10,7))
plt.plot(x, G.pdf(x), '--', label='Ridge: $\mathcal{N}(\mu=0,\sigma^2=1/\lam
plt.plot(x, L.pdf(x), label='Lasso: $\mathcal{L}(\mu=0,b=1/\lambda)$')
plt.legend(loc='best', fontsize=15); plt.xlabel('$\mathbf{w}$', size=15)
plt.ylabel('$P(\mathbf{w})$', size=15); plt.title('Prior Probability',size=1
```



Maximum Likelihood Estimation (MLE) & Maximum A Posteriori (MAP)

Maximum Likelihood Estimation (MLE)

(Frequentist approach)

In **Maximum Likelihood Estimation** (also referred to as **MLE**) we want to *find the set of parameters* that **maximize** the data likelihood $P(\mathbf{t}|\mathbf{y}(\mathbf{x};\mathbf{w}))$. We want to find the *optimal* set of parameters under some assumed distribution such that the data is most likely.

- MLE focuses on maximizing the data likelihood, which *usually* provides a pretty good estimate
- A common trick to maximize the data likelihood is to maximize the log likelihood
- MLE is purely data driven
- MLE works best when we have lots and lots of data
- MLE will likely overfit when we have small amounts of data or, at least, becomes unreliable
- It estimates relative frequency for our model parameters. Therefore it needs incredibly large amounts of data (infinite!) to estimate the true likelihood parameters
    - This is a problem when we want to make inferences and/or predictions outside the range of what the training data has learned

Maximum A Posteriori (MAP)

(Bayesian approach)

In **Maximum A Posteriori** (also referred as **MAP**) we want to *find the set of parameters* that **maximize** the posterior probability $P(\mathbf{w}|\mathbf{t})$. We want to find the *optimal* set of parameters under some assumed distribution such that the parameters are most likely to have been drawn off of given some prior beliefs.

- MAP focuses on maximizing the posterior probability - data likelihood with a prior
- A common trick to maximize the posterior probability is to maximize the log likelihood
- MAP is data driven

- MAP is mostly driven by the prior beliefs
- MAP works great with small amounts of data *if* our prior was chosen well
- We need to assume and select a distribution for our prior beliefs
    - A wrong choice of prior distribution can impact negatively our model estimation
- When we have lots and lots of data, the data likelihood will take over and the posterior will depend less and less on the prior

---

Illustration Example

Suppose you flip a coin 3 times. Let H_i be the event that we observe Heads on flip i. Consider the event $E = H_1 \cap H_2 \cap H_3$, i.e. all flips were Heads. What is the probability that the next flip is Heads?

From classical probability (frequentist), we look only at data to compute the probability of flipping Heads:

$$P(H) = \frac{\text{No. of observed Heads}}{\text{No. of flips}} = \frac{3}{3} = 1$$

From Bayesian statistics, we use Bayesian inferencing: What is the **hidden state** in this problem?

- Hidden state: what type of coin was use in the experiment
- Let's assume that we think only two types of coins could have been used, one fair coin and one 2-headed coin. So, by the **Law of Total Probability**:

$$P(H) = P(H|\text{fair})P(\text{fair}) + P(\overline{H}|\text{2-headed})P(\text{2-headed})$$

Furthermore, we can test different hypothesis by checking which hypothesis has the largest posterior probability value, e.g. if $P(\text{fair}|E) > P(\text{2-headed}|E)$, then hypothesis "fair" is more likely and that is what we will use to make predictions.

Note that the events H_i are **conditionally independent**, that is:
$P(H_1 \cap H_2|\text{fair}) = P(H_1|\text{fair})P(H_2|\text{fair})$.

---

to be continued...
