Data-Driven Design & Analyses of Structures & Materials (3dasm)

Lecture 11

Miguel A. Bessa | **miguel_bessa@brown.edu** | Associate Professor

Outline for today

- Derivation of different Linear Regression models
    - Picking up where we left off in Lectures 8 and 9.

Reading material: This notebook + Chapter 11 of the book.

Recap of Lectures 8 and 9

Recall our view of Linear regression models from a Bayesian perspective: it's all about the choice of **likelihood** and **prior**!

Likelihood	Prior (on the weights)	Posterior	Name of the model	Book section
Gaussian	Uniform	Point estimate	Least Squares regression	11.2.2
Gaussian	Gaussian	Point estimate	Ridge regression	11.3
Gaussian	Laplace	Point estimate	Lasso regression	11.4
Student-t	Uniform	Point estimate	Robust regression	11.6.1
Laplace	Uniform	Point estimate	Robust regression	11.6.2
Gaussian	Gaussian	Gaussian	Bayesian linear regression	11.7

Let's continue along the lines of Lectures 8 and 9, and derive a few of these models for the multidimensional case.

We are now totally prepared to derive any ML model in any dimension!
In Lecture 8 and its Homework we derived linear regression models using 1D input x, 1D output y, and a polynomial basis function $\boldsymbol{\phi}(x)$.
We will quickly recap what we did then, and then show how this generalizes to multidimensional inputs $\mathbf{x}$ and for any kind of basis function $\boldsymbol{\phi}(\mathbf{x})$.

- Note: without loss of generality, we will keep considering a single output y.

Linear Least Squares: Linear regression with Gaussian likelihood, Uniform prior and posterior via Point estimate

Likelihood	Prior (on the weights)	Posterior	Name of the model	Book section
Gaussian	Uniform	Point estimate	Least Squares regression	11.2.2

This model assumes a Gaussian observation distribution with constant variance and "linear" mean (recall: linear in the unknowns $\mathbf{z}$). If considering 1D input x and 1D output y the model is written as:

1. Gaussian observation distribution: $p(y|x,\mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T\boldsymbol{\phi}(x), \sigma^2_{y|z} = \sigma^2)$

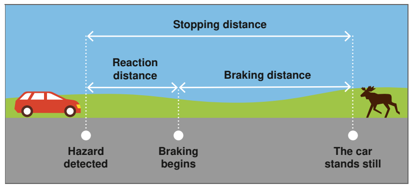
where $\mathbf{z} = (\mathbf{w}, \sigma)$ are all the unknown model parameters (hidden rv's).

1. Uniform prior distribution for each hidden rv in $\mathbf{z}$: $p(\mathbf{z}) \propto 1$
2. MLE point estimate for posterior: $\hat{\mathbf{z}}_{\text{mle}} = \underset{z}{\operatorname{argmin}} \left[-\sum_{i=1}^{N} \log p(y = y_i | x = x_i, \mathbf{z})\right]$

Final prediction is given by the PPD: $p(y|x,\mathcal{D}) = \int p(y|x,\mathbf{z})\delta(\mathbf{z}-\hat{\mathbf{z}})dz = p(y|x,\mathbf{z}=\hat{\mathbf{z}})$

Recall the car stopping distance problem

Let's focus (again) on our favorite problem, but now we will not keep the velocity of the car x fixed.
If we knew the "ground truth" of this problem, then it would be given by:
$y = z_1 \cdot x + z_2 \cdot x^2$



- y is the **output**: the car stopping distance (in meters)
- z_1 is a hidden variable: an rv representing the driver's reaction time (in seconds)
- z_2 is another hidden variable: an rv that depends on the coefficient of friction, the inclination of the road, the weather, etc. (in m^{-1}s^{-2}).
- x is the **input**: constant car velocity (in m/s).

Remember: We don't know the "true" nature of the unknowns (the $\mathbf{z}$ rv's). For example, they could be $z_1 \sim \mathcal{N}(\mu_{z_1} = 1.5, \sigma^2_{z_1} = 0.5^2)$, and $z_2 \sim \delta(z_2 - 0.1)$, i.e. z_2 could just be $z_2 = 0.1$. Either way, when we don't know them we assume that they are rv's with some prior distribution and we use Bayesian inference (or point estimates) to determine a posterior estimate.

Unsurprisingly, in Lecture 9 we saw that a linear model with a **quadratic polynomial basis function** predicts the stopping distance for this problem very well. For example, we considered the following model:

1. Gaussian observation distribution: $p(y|x, \mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T\boldsymbol{\phi}(x), \sigma^2_{y|z} = \sigma^2)$

where $\mathbf{z} = (\mathbf{w}, \sigma)$ are all the hidden rv's of the model, i.e. the model parameters.

- the vector $\mathbf{w} = [w_0, w_1, w_2 \ldots, w_{M-1}]^T$ includes the **bias** term w_0 and the remaining **weights** w_m with $m = 1, \ldots, M-1$.
- the vector $\boldsymbol{\phi}(x) = [1, x, x^2, \ldots, x^{M-1}]^T$ includes the **basis functions**, which for a 1D input x correspond to a polynomial of degree $M-1$. So, when $M = 3$ we have a quadratic polynomial basis for 1D input x, i.e. $\mu_{y|z} = w_0 + w_1 x + w_2 x^2$.

1. Uniform prior distribution for each hidden rv in $\mathbf{z}$: $p(\mathbf{z}) \propto 1$

1. MLE point estimate for posterior: $\hat{\mathbf{z}}_{\text{mle}} = \underset{z}{\text{argmin}} \left[-\sum_{n=1}^{N} \log p(y = y_n | x_n, \mathbf{z}) \right]$

For other problems, the polynomial degree of the basis functions may need to be different.

- For example, in Lecture 10 we saw that when the ground truth was $x \sin x$ then the polynomial basis functions need to have a higher degree. However, even then the approximation is not brilliant because the ground truth is not really a polynomial!

There are other basis functions that can be adopted. For example, spline basis functions (Section 11.5 in the book), among many other possibilities (kernels!).

As we also mentioned, as long as the basis functions $\phi(x)$ do not depend on any rv $\mathbf{z}$ and the mean of observation distribution is defined linearly as a function of the rv's, then we still have a linear regression model.

But now let's consider problems that still have only one output y but that can have multiple inputs $\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$ where x_d is feature d and where $d = 1, \ldots, D$.

In this case, we can write the linear regression model for multi-dimensional input as:

1. Gaussian observation distribution: $p(y|\mathbf{x}, \mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \sigma^2_{y|z} = \sigma^2)$

where $\mathbf{z} = (\mathbf{w}, \sigma)$ are all the hidden rv's of the model, i.e. the model parameters.

- the vector $\mathbf{w} = [w_0, w_1, w_2 \ldots, w_{M-1}]^T$ includes the **bias** term w_0 and the remaining **weights** w_m with $m = 0, \ldots, M-1$.
- and the basis functions remain a vector but where each element also acts on a vector $\mathbf{x}$, where x_d has D features: $\boldsymbol{\phi}(\mathbf{x}) = [\phi_0(\mathbf{x}), \phi_1(\mathbf{x}), \phi_2(\mathbf{x}) \ldots, \phi_{M-1}(\mathbf{x})]^T$

For example, for a 2D input $\mathbf{x} = [x_1, x_2]^T$ the quadratic polynomial basis has $M = 6$ leading to:

$$\mu_{y|z} = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1 x_2 + w_4 x_1^2 + w_5 x_2^2$$

and where the remaining choices for the linear regression model remain the same:

1. Uniform prior distribution for each hidden rv in $\mathbf{z}$: $p(\mathbf{z}) \propto 1$
2. MLE point estimate for posterior: $\hat{\mathbf{z}}_{\mathrm{mle}} = \underset{z}{\operatorname{argmin}} \left[- \sum_{n=1}^{N} \log p(y = y_n | \mathbf{x} = \mathbf{x}_n, \mathbf{z}) \right]$

Final prediction is given by the PPD:

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathbf{z}) \delta(\mathbf{z} - \hat{\mathbf{z}}) dz = p(y|\mathbf{x}, \mathbf{z} = \hat{\mathbf{z}})$$

Therefore, we are capable of predicting the PPD by discovering the unknowns $\mathbf{z}$ via the point estimate of the posterior, which requires solving the argmin of the negative log likelihood.

Now, let's focus on estimating the unknowns $\mathbf{z}$ via the MLE point estimate of the posterior (maximum likelihood estimation).
As we saw in Lecture 8, finding the MLE is the same as finding the location of the minimum of the negative log likelihood.
Since our observation distribution is a multivariate Gaussian,
$p(y|\mathbf{x}, \mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \sigma^2_{y|z} = \sigma^2)$
then the likelihood is given by (Lecture 5 but now with vectors):

$$p(y = \mathcal{D}_y|\mathbf{x} = \mathcal{D}_x, \mathbf{z}) = \prod_{n=1}^{N} p(y = y_n|\mathbf{x} = \mathbf{x}_n, \mathbf{z}) \tag{1}$$
$$= p(y = y_1|\mathbf{x} = \mathbf{x}_1, \mathbf{z})p(y = y_2|\mathbf{x} = \mathbf{x}_2, \mathbf{z}) \cdots p(y = y_N|\mathbf{x} = \mathbf{x}_N, \mathbf{z}) \tag{2}$$

which we already know that is also a multivariate Gaussian (unnormalized).

But, since we are not going fully Bayesian, the only thing we need to estimate is the location of the maximum of the likelihood (point estimate!):

$$\begin{aligned} \hat{\mathbf{z}}_{\text{mle}} &= \underset{z}{\operatorname{argmin}} \left[\text{NLL}(\mathbf{z})\right] & (3) \\ &= \underset{z}{\operatorname{argmin}} \left[-\sum_{n=1}^{N} \log p(y = y_n | \mathbf{x} = \mathbf{x}_n, \mathbf{z}) \right] & (4) \end{aligned}$$

In Lecture 9 we allowed scikit-learn to find the minimum for us! But today we will actually determine this minimum...
You already did this in the Homework of Lecture 8 for the 1D case with a linear polynomial basis and fixing x. The multivariate case for a general basis function and for different $\mathbf{x}$ is just as easy! Especially when considering the variance of the observation distribution to be the same everywhere!

$$\begin{aligned}
\hat{\mathbf{z}}_{\text{mle}} &= \underset{z}{\operatorname{argmin}} \left[-\sum_{n=1}^{N} \log p(y = y_n | \mathbf{x} = \mathbf{x}_n, \mathbf{z}) \right] && (5) \\
&= \underset{z}{\operatorname{argmin}} \left[-\sum_{n=1}^{N} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{ -\frac{1}{2\sigma^2} \left[y_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \right]^2 \right\} \right) \right] && (6) \\
&= \underset{z}{\operatorname{argmin}} \left[\frac{N}{2} \log\left(2\pi\sigma^2\right) + \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left[y_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \right]^2 \right] && (7)
\end{aligned}$$

where we recall that the unknowns are $\mathbf{z} = (\mathbf{w}, \sigma)$.

To find the minimum location we need to take the gradient of the $\mathrm{NLL}(\mathbf{z})$ wrt $\mathbf{z}$ and equal it to zero:

$$\nabla_{\mathbf{z}}\mathrm{NLL}(\mathbf{z}) = \mathbf{0}$$

which can be written as,

$$\begin{bmatrix} \frac{\partial \mathrm{NLL}(\mathbf{z})}{\partial w_0} \\ \frac{\partial \mathrm{NLL}(\mathbf{z})}{\partial w_1} \\ \vdots \\ \frac{\partial \mathrm{NLL}(\mathbf{z})}{\partial w_{M-1}} \\ \frac{\partial \mathrm{NLL}(\mathbf{z})}{\partial \sigma^2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

We can first solve this system of equations wrt $\mathbf{w}$, and then solve wrt σ

Then, solving first for the weights $\mathbf{w}$:

$$\nabla_{\mathbf{w}} \mathrm{NLL}(\mathbf{w}, \sigma^2) = \mathbf{0}$$

we note that,

$$\begin{aligned} \nabla_{\mathbf{w}} \mathrm{NLL}(\mathbf{w}, \sigma^2) &= \mathbf{0} && (8) \\ \nabla_{\mathbf{w}} \left[\frac{N}{2} \log\left(2\pi\sigma^2\right) + \frac{1}{2\sigma^2} \sum_{n=1}^{N} \left[y_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\right]^2 \right] &= \mathbf{0} && (9) \\ \nabla_{\mathbf{w}} \left[\underbrace{\frac{1}{2} \sum_{n=1}^{N} \left[y_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n)\right]^2}_{\mathrm{RSS}(\mathbf{w})} \right] &= \mathbf{0} && (10) \end{aligned}$$

Note: in Statistics the term in the argument is called **residual sum of squares**.

We can rewrite the above expression in simpler form:

$$\nabla_{\mathbf{w}} \left[\frac{1}{2} \sum_{n=1}^{N} \left[y_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \right]^2 \right] = 0 \tag{11}$$

$$\nabla_{\mathbf{w}} \left[\frac{1}{2} (\mathbf{\Phi} \mathbf{w} - \mathbf{y})^T (\mathbf{\Phi} \mathbf{w} - \mathbf{y}) \right] = 0 \tag{12}$$

where we group all output measurements y_n into a $N \times 1$ vector $\mathbf{y}$ and where we group all N evaluations of the basis functions into the $N \times M$ matrix:

$$\mathbf{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{bmatrix}$$

Setting the gradient wrt all $\mathbf{w}$ to zero gives,

$$\nabla_{\mathbf{w}}\left[\frac{1}{2}(\mathbf{\Phi w}-\mathbf{y})^T(\mathbf{\Phi w}-\mathbf{y})\right]=\mathbf{0} \tag{13}$$

$$\frac{1}{2}\left[\left(\mathbf{\Phi}^T\mathbf{\Phi}+\mathbf{\Phi}^T\mathbf{\Phi}\right)\mathbf{w}-\mathbf{\Phi}^T\mathbf{y}-\mathbf{\Phi}^T\mathbf{y}\right]=\mathbf{0} \tag{14}$$

where we used the identity $\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}}=\left(\mathbf{A}+\mathbf{A}^T\right)\mathbf{x}$. (See Section 7.8 of Murphy's book if you need to revise matrix calculus).
From which we reach the MLE prediction for the weights:

$$\hat{\mathbf{w}}_{\text{mle}}=\left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{y}$$

We conclude that our point estimate for the posterior (MLE) is: $\hat{\mathbf{w}}_{\text{MLE}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{y}$
where the quantity

$$\mathbf{\Phi}^\dagger = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T$$

is known as the Moore-Penrose pseudo-inverse of the matrix $\mathbf{\Phi}$. It can be regarded as a generalization of the notion of matrix inverse to **nonsquare matrices**. In the special case of $\mathbf{\Phi}$ being square and invertible, then using the property $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ we see that $\mathbf{\Phi}^\dagger = \mathbf{\Phi}^{-1}$.

Also note that we could have calculated separetely the bias term w_0 (which is convenient because for other models the bias usually has a uniform prior, unlike the remaining weights). If we do that we obtain:

$$\hat{w}_0 = \bar{y} - \sum_{m=1}^{M-1} w_m \bar{\phi}_m$$

where we defined $\bar{y} = \frac{1}{N}\sum_{n=1}^{N} y_n$ and $\bar{\phi}_m = \frac{1}{N}\sum_{n=1}^{N} \phi_m(\mathbf{x}_n)$.

Having found the solution for all $\mathbf{w}$, we just need to find one last unknown from the point estimate of the posterior:

$$\nabla_{\sigma^2}\mathrm{NLL}(\mathbf{w}, \sigma^2) = 0$$

which is particularly simple:

$$\hat{\sigma}_{\mathrm{mle}} = \frac{1}{N}\sum_{n=1}^{N}\left[y_n - \hat{\mathbf{w}}_{\mathrm{mle}}^T \boldsymbol{\phi}(\mathbf{x}_n)\right]^2$$

In summary, the MLE point estimate of the posterior leads to the following estimation of parameters:

$$\hat{\mathbf{w}}_{\text{mle}} = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{y}$$

$$\hat{\sigma}_{\text{mle}} = \frac{1}{N}\sum_{n=1}^{N}\left[y_n - \hat{\mathbf{w}}_{\text{mle}}^T\boldsymbol{\phi}(\mathbf{x}_n)\right]^2$$

where the Moore-Penrose pseudo inverse needs to be calculated $\mathbf{\Phi}^\dagger = \left(\mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T$.
This calculation can be done efficiently by many libraries, including Numpy.

- For example, scikit-learn uses a solver based on SVD (Single Value Decomposition: the most common dimensionality reduction method) which is efficient when $N > M$ (overdetermined system). Book section 7.5 has an excellent summary of SVD, if you are curious.
- Of course, if $N = M$ then there is a unique solution (you'll verify this in Homework 4) and the error on the training set becomes zero (linear regression becomes fully interpolatory).

Ridge regression: Linear regression with Gaussian likelihood, Gaussian prior and posterior via Point estimate

Likelihood	Prior (on the weights)	Posterior	Name of the model	Book section
Gaussian	Gaussian	Point estimate	Ridge regression	11.3

1. Gaussian observation distribution: $p(y|\mathbf{x}, \mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \sigma^2_{y|z} = \sigma^2)$

where $\mathbf{z} = (\mathbf{w}, \sigma)$ are all the unknown model parameters (hidden rv's).

1. But using a Gaussian prior for the weights $\mathbf{w}$: $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \overset{<}{\sigma}_w \mathbf{I})$
2. MAP point estimate for posterior: $\hat{\mathbf{z}}_{\text{map}} = \underset{z}{\operatorname{argmin}} \left[-\sum_{i=1}^{N} \log p(y = y_i|\mathbf{z}) - \log p(\mathbf{w}) \right]$

Final prediction is given by the PPD: $p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathbf{z}) \delta(\mathbf{z} - \hat{\mathbf{z}}_{\text{map}}) dz = p(y|\mathbf{x}, \mathbf{z} = \hat{\mathbf{z}}_{\text{map}}, \mathcal{D})$

Computing the MAP estimate is very similar to what we did for the MLE:

$$\hat{\mathbf{w}}_{\text{map}} = \underset{w}{\text{argmin}} \left[\frac{1}{2\sigma^2} (\mathbf{\Phi w} - \mathbf{y})^T (\mathbf{\Phi w} - \mathbf{y}) + \frac{1}{2\sigma_w^2} \mathbf{w}^T \mathbf{w} \right] \tag{15}$$

$$= \underset{w}{\text{argmin}} \left[\text{RSS}(\mathbf{w}) + \lambda ||\mathbf{w}||_2^2 \right] \tag{16}$$

where $\lambda = \frac{\sigma^2}{\sigma_w^2}$ is proportional to the strength of the prior, and

$||\mathbf{w}||_2^2 = \sqrt{\sum_{m=1}^{M-1} |w_m|^2} = \sqrt{\mathbf{w}^T \mathbf{w}}$

is called the l_2 norm of the vector $\mathbf{w}$. Thus, we are penalizing weights that become too large in magnitude. In ML literature this is usually called l_2 **regularization** or **weight decay**, and is very widely used.

In Homework 4, you will experience the difference between Linear Least Squares and Ridge regression, reporting on the influence of the prior strength for the latter.

Then, solving the MAP first for the weights $\mathbf{w}$, as we did for the MLE:

$$\nabla_{\mathbf{w}} \left[\frac{1}{2\sigma^2} (\mathbf{\Phi w} - \mathbf{y})^T (\mathbf{\Phi w} - \mathbf{y}) + \frac{1}{2\sigma_w^{<2}} \mathbf{w}^T \mathbf{w} \right] = \mathbf{0} \tag{17}$$

$$\nabla_{\mathbf{w}} \left[(\mathbf{\Phi w} - \mathbf{y})^T (\mathbf{\Phi w} - \mathbf{y}) + \lambda \mathbf{w}^T \mathbf{w} \right] = \mathbf{0} \tag{18}$$

from which we determine the MAP estimate for the the weights $\mathbf{w}$ as:

$$\hat{\mathbf{w}}_{\text{map}} = \left(\mathbf{\Phi}^T \mathbf{\Phi} + \lambda \mathbf{I}_M \right)^{-1} \mathbf{\Phi}^T \mathbf{y} = \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) \boldsymbol{\phi}(\mathbf{x}_n)^T + \lambda \mathbf{I}_M \right)^{-1} \left(\sum_{n=1}^{N} \boldsymbol{\phi}(\mathbf{x}_n) y_n \right)$$

Once again, this can be solved using SVD or other methods to ensure that the Moore-Penrose pseudo-inverse is calculated properly.

Lasso regression: Linear regression with Gaussian likelihood, Laplace prior and posterior via Point estimate

Likelihood	Prior (on the weights)	Posterior	Name of the model	Book section
Gaussian	Laplace	Point estimate	Lasso regression	11.4

1. Gaussian observation distribution: $p(y|\mathbf{x},\mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}), \sigma^2_{y|z} = \sigma^2)$

where $\mathbf{z} = (\mathbf{w}, \sigma)$ are all the unknown model parameters (hidden rv's).

1. But using a **Laplace** prior for the weights $\mathbf{w}$: $p(\mathbf{w}) = \prod_{m=1}^{M-1} \mathrm{Lap}\left(w_m|0, 1/\overset{<}{\lambda}_w\right) \propto \prod_{m=1}^{M-1} \exp\left[-\overset{<}{\lambda}_w|w_m|\right]$
2. MAP point estimate for posterior: $\hat{\mathbf{z}}_{\text{map}} = \underset{z}{\operatorname{argmin}}\left[-\sum_{i=1}^{N}\log p(y = y_i|\mathbf{z}) - \log p(\mathbf{w})\right]$

Final prediction is given by the PPD: $p(y|\mathbf{x},\mathcal{D}) = \int p(y|\mathbf{x},\mathbf{z})\delta(\mathbf{z} - \hat{\mathbf{z}}_{\text{map}})dz = p(y|\mathbf{x},\mathbf{z} = \hat{\mathbf{z}}_{\text{map}}, \mathcal{D})$

Computing the MAP estimate:

$$\hat{\mathbf{w}}_{\text{map}} = \underset{w}{\operatorname{argmin}} \left[(\mathbf{\Phi w} - \mathbf{y})^T (\mathbf{\Phi w} - \mathbf{y}) + \overset{<}{\lambda}_w ||\mathbf{w}||_1 \right] \tag{19}$$

where $||\mathbf{w}||_1 = \sum_{m=1}^{M-1} |w_m|$ is called the l_1 norm of $\mathbf{w}$. In ML literature this is called l_1 regularization.

- Calculating the MAP for Lasso is not done the same way as for Ridge because the term $||\mathbf{w}||_1$ is not differentiable whenever $w_m = 0$. In this case, the solution is found using hard- or soft-thresholding (See Section 11.4.3 in the book) because the gradient becomes a branch function.

- A more important point is to see that **different types of prior distributions** introduce **different regularizations** on the weights, aleviating overfitting in a different manner.
    - For example, in the case of Lasso, since the Laplace prior puts more density around the mean (which is zero here) than the Gaussian prior, then it has a tendency to lead the weights to zero, i.e. it introduces **sparsity** when estimating the weights via MAP. The book has a beautiful discussion about this.

Bayesian linear regression: Linear regression with Gaussian likelihood, Gaussian prior and Gaussian posterior (Bayesian solution)

As we saw in the beginning of the Lecture, there are many more models we can define! The book covers quite a few!

Likelihood	**Prior (on the weights)**	**Posterior**	**Name of the model**	**Book section**
Gaussian	Gaussian	Gaussian	Bayesian linear regression	11.7

1. Gaussian observation distribution (with **known** variance): $p(y|\mathbf{x},\mathbf{z}) = \mathcal{N}(y|\mu_{y|z} = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}), \sigma^2_{y|z} = \sigma^2)$

where $\mathbf{z} = \mathbf{w}$ are all the unknown model parameters (hidden rv's), and where σ is **not** treated as an unknown, i.e. it is a **hyperparameter**.

1. Gaussian prior for the weights $\mathbf{w}$: $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\overset{<}{\boldsymbol{\mu}}_w, \overset{<}{\boldsymbol{\Sigma}}_w)$
2. Gaussian posterior (obtained from Bayes rule)

Final prediction is given by the PPD: $p(y|\mathbf{x},\mathcal{D}) = \int p(y|\mathbf{x},\mathbf{z})p(\mathbf{z}|\mathcal{D})dz$

At this point you may notice that we have already derived this model in Lecture 7.
The only differences are that now we have multiple weights $\mathbf{z}$, multidimensional inputs $\mathbf{x}$ and that we allow them to have different values.
Yet, the derivation is the same! We just need to bold the letters. Let's do it:
The likelihood is a (multivariate) Gaussian distribution (product of MVN evaluated at each data point):

$$p(\mathcal{D}|\mathbf{w}, \sigma^2) = \prod_{n=1}^{N} p(y_n|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{\Phi}\mathbf{w}, \sigma^2\mathbf{I}_N)$$

where $\mathbf{I}_N$ is the $N \times N$ identity matrix, as defined previously.

To calculate the posterior, we also use the product of Gaussians rule (in Lecture 5 we also defined this rule for multivariate Gaussians!):

$$p(\mathbf{w}|\boldsymbol{\Phi}, \mathbf{y}, \sigma^2) \propto \mathcal{N}(\mathbf{y}|\boldsymbol{\Phi}\mathbf{w}, \sigma^2\mathbf{I}_N)\mathcal{N}(\mathbf{w}|\overset{<}{\boldsymbol{\mu}}_w, \overset{<}{\boldsymbol{\Sigma}}_w) = \mathcal{N}(\mathbf{w}|\overset{>}{\boldsymbol{\mu}}_w, \overset{>}{\boldsymbol{\Sigma}}_w)$$

where the mean and covariance of the posterior are given by:

$$\overset{>}{\boldsymbol{\mu}}_w = \overset{>}{\boldsymbol{\Sigma}}_w \left(\overset{<}{\boldsymbol{\Sigma}}_w^{-1} \overset{<}{\boldsymbol{\mu}}_w + \frac{1}{\sigma^2}\boldsymbol{\Phi}^T\mathbf{y} \right)$$

$$\overset{>}{\boldsymbol{\Sigma}}_w = \left(\overset{<}{\boldsymbol{\Sigma}}_w^{-1} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^T\boldsymbol{\Phi} \right)^{-1}$$

Often, we use a prior with zero mean $\overset{<}{\boldsymbol{\mu}}_w = \mathbf{0}$ and diagonal covariance $\overset{<}{\boldsymbol{\Sigma}}_w = \overset{<}{\sigma}_w^2\mathbf{I}_M$ like the prior we used in Ridge regression. In this case, the posterior mean becomes the same as the MAP estimate obtained from Ridge regression: $\overset{>}{\boldsymbol{\mu}}_w = (\lambda\mathbf{I}_M + \boldsymbol{\Phi}^T\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\mathbf{y}$.

Having determined the posterior, we can determine what we really want: the PPD.

$$\begin{aligned}
p(y|\mathbf{x},\mathcal{D}) &= \int p(y|\mathbf{x},\mathbf{z})p(\mathbf{z}|\mathcal{D})d\mathbf{z} && (20)\\
p(y|\mathbf{x},\mathcal{D},\sigma^2) &= \int p(y|\mathbf{x},\mathbf{w},\sigma^2)p(\mathbf{w}|\mathcal{D})d\mathbf{w} && (21)\\
&= \int \mathcal{N}(y|\boldsymbol{\phi}(\mathbf{x})^T\mathbf{w},\sigma^2)\mathcal{N}(\mathbf{w}|\overset{>}{\boldsymbol{\mu}}_w,\overset{>}{\boldsymbol{\Sigma}}_w)d\mathbf{w} && (22)\\
&= \mathcal{N}\left(y \mid \boldsymbol{\phi}(\mathbf{x})^T\overset{>}{\boldsymbol{\mu}}_w\,,\ \sigma^2+\boldsymbol{\phi}(\mathbf{x})^T\overset{>}{\boldsymbol{\Sigma}}_w\boldsymbol{\phi}(\mathbf{x})\right) && (23)
\end{aligned}$$

where we recall the meaning of each term:

- $(\mathbf{x},\, y)$ is the point were we want to make a prediction
- $\boldsymbol{\phi}(\mathbf{x})$ is an $M\times 1$ vector of basis functions
- $\overset{>}{\boldsymbol{\mu}}_w$ is the $M\times 1$ vector with the mean of the posterior for the weights
- and $\overset{>}{\boldsymbol{\Sigma}}_w$ is the $M\times M$ covariance matrix of the posterior for the weights.

Observing the obtained PPD,

$$p(y|\mathbf{x},\mathcal{D},\sigma^2) = \mathcal{N}\left(y \mid \overset{>}{\boldsymbol{\mu}}_w^T \boldsymbol{\phi}(\mathbf{x})\,,\, \sigma^2 + \boldsymbol{\phi}(\mathbf{x})^T \overset{>}{\boldsymbol{\Sigma}}_w \boldsymbol{\phi}(\mathbf{x})\right) \tag{27}$$

we see something very interesting: the variance of the PPD at a point $\mathbf{x}$ after seeing N data points depends on two terms:

1. the variance of the observation noise, σ^2 that we defined to be constant
2. and the variance in the parameters obtained by the posterior $\overset{>}{\boldsymbol{\Sigma}}_w$

This means that the predicted uncertainty increases when $\mathbf{x}$ is located far from the training data $\mathcal{D}$
, just like we want it to be! We are less certain about points **away** from our observations (training data).
You will see that more advanced methods such as Gaussian processes are not too different from Bayesian linear regression...

Other linear regression models

As we saw in the beginning of the Lecture, there are many more models we can define! The book covers quite a few!

Likelihood	Prior (on the weights)	Posterior	Name of the model	Book section
Gaussian	Uniform	Point estimate	Least Squares regression	11.2.2
Gaussian	Gaussian	Point estimate	Ridge regression	11.3
Gaussian	Laplace	Point estimate	Lasso regression	11.4
Gaussian	Gaussian$\times$Laplace	Point estimate	Elastic net	11.4.8
Student-t	Uniform	Point estimate	Robust regression	11.6.1
Laplace	Uniform	Point estimate	Robust regression	11.6.2
Gaussian	Gaussian	Gaussian	Bayesian linear regression	11.7

For example, the "Elastic net" is literally the combination of Ridge regression and Lasso by defining a prior that depends on both a Gaussian and a Laplace distribution. This model was proposed in 2005. Five years later the "Bayesian Elastic Net" was also proposed, where the posterior is calculated in a Bayesian way (just like what we did for Bayesian linear regression).

In summary

Almost every ML model is derived following 4 steps:

1. Define the Observation distribution and compute the likelihood.
2. Define the prior and its parameters.
3. Compute the posterior to estimate the unknown parameters (whether via a Bayesian approach or via a Point estimate).
4. Compute the PPD.

Done.

If you only care about the mean of the PPD (the mean of your prediction), then you can compute it directly without even thinking about uncertainty... But then, make sure you characterize the quality of your predictions using an appropriate **error metric** and use strategies like **cross-validation**, as you will explore in Homework 4!

See you next class

Have fun!