Bayesian Analysis for Machine Learning

AI Friends Seminar

Ganguk Hwang

Department of Mathematical Sciences
KAIST

Bayesian Model for Linear Regression

The Standard Linear Model

Let $\mathcal{D} = \{(\mathbf{x_i}, y_i)|i = 1, \cdots, n\}$ be a training set of n observations where $\mathbf{x_i}$ is an input vector of dimension D and y is a scalar output.

Let $X = [\mathbf{x_1}, \cdots, \mathbf{x_n}]$.

The Standard Linear Model

First, we will consider the standard linear regression model with Gaussian noise

$$f(\mathbf{x_i}) = \mathbf{x_i}^\top \mathbf{w},\ y_i = f(\mathbf{x_i}) + \epsilon_i$$

where $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$ and $\{\epsilon_i\}_{i=1}^n$ are independent.

Then, the *likelihood* can be computed as

$$\begin{aligned} p(\mathbf{y}|X, \mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x_i}, \mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\Big(-\frac{(y_i - \mathbf{x_i}^\top \mathbf{w})^2}{2\sigma_n^2}\Big) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\Big(-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^\top \mathbf{w}|^2\Big) \sim \mathcal{N}(X^\top \mathbf{w}, \sigma_n^2 I) \end{aligned}$$

The Standard Linear Model

In the Bayesian treatment, we need to specify a *prior* over the parameters.
Suppose $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$.
By Bayes' rule,

$$p(\mathbf{w}|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w}, X)}{p(\mathbf{y}, \mathbf{X})} = \frac{p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}.$$

Here, $p(\mathbf{y}|X) = \int p(\mathbf{y}|X, \mathbf{w})p(\mathbf{w})d\mathbf{w}$ is the normalizing constant. (It is called as the *marginal likelihood*).

Since it is just a constant, we neglect $p(\mathbf{y}|X)$ when we compute the *posterior* $p(\mathbf{w}|X, \mathbf{y})$.

The Standard Linear Model

$$p(\mathbf{w}|X,\mathbf{y}) \propto p(\mathbf{y}|X,\mathbf{w})p(\mathbf{w})$$
$$\propto \exp\Big(-\frac{1}{2\sigma_n^2}(\mathbf{y}-X^\top\mathbf{w})^\top(\mathbf{y}-X^\top\mathbf{w})\Big)\exp\Big(-\frac{1}{2}\mathbf{w}^\top\Sigma_p^{-1}\mathbf{w}\Big)$$

Here,

$$\frac{1}{\sigma_n^2}(\mathbf{y}-X^\top\mathbf{w})^\top(\mathbf{y}-X^\top\mathbf{w})+\mathbf{w}^\top\Sigma_p^{-1}\mathbf{w}=\mathbf{w}^\top A\mathbf{w}-B\mathbf{w}-\mathbf{w}^\top C+D$$

Where $A=\frac{1}{\sigma_n^2}XX^\top+\Sigma_p^{-1}, B=\frac{1}{\sigma_n^2}(Xy)^\top, C=\frac{1}{\sigma_n^2}X\mathbf{y}, D=\frac{1}{\sigma_n^2}\mathbf{y}^\top\mathbf{y}$

The Standard Linear Model

Observe that

$$(\mathbf{w}-\overline{\mathbf{w}})^\top A(\mathbf{w}-\overline{\mathbf{w}}) = \mathbf{w}^\top A\mathbf{w} - \overline{\mathbf{w}}^\top A\mathbf{w} - \mathbf{w}^\top A\overline{\mathbf{w}} + \overline{\mathbf{w}}^\top A\overline{\mathbf{w}}$$

Now, set $\overline{\mathbf{w}} = A^{-1}C$. Then,

$$\overline{\mathbf{w}} = \frac{1}{\sigma_n^2}A^{-1}X\mathbf{y}$$

$$\overline{\mathbf{w}}^\top A = \frac{1}{\sigma_n^2}(X\mathbf{y})^\top = B$$

By neglecting the constant term, we get

$$p(\mathbf{w}|X,\mathbf{y}) \propto \exp\big(-\frac{1}{2}(\mathbf{w}-\overline{\mathbf{w}})^\top A(\mathbf{w}-\overline{\mathbf{w}})\big)$$

In other words,

$$p(\mathbf{w}|X,\mathbf{y}) \sim \mathcal{N}(\frac{1}{\sigma_n^2}A^{-1}X\mathbf{y}, A^{-1})$$

Gaussian Identities

Theorem 1

The product of two Gaussians gives another Gaussian

$$\mathcal{N}(\mathbf{x}|\mathbf{a}, A)\mathcal{N}(\mathbf{x}|\mathbf{b}, B) = Z^{-1}\mathcal{N}(\mathbf{x}|\mathbf{c}, C)$$

where

$$\mathbf{c} = C(A^{-1}\mathbf{a} + B^{-1}\mathbf{b}), C = (A^{-1} + B^{-1})^{-1}, \text{ and}$$

$$Z^{-1} = (2\pi)^{-D/2}|A + B|^{-1/2} \exp\Big(-\frac{1}{2}(\mathbf{a} - \mathbf{b})^{\top}(A + B)^{-1}(\mathbf{a} - \mathbf{b})\Big).$$

Here, Z is just the normalizing constant.

The Standard Linear Model

Predictive distribution

Definition 1

The (Posterior) predictive distribution is the distribution of possible unobserved values (test data) conditional on the observed values (training data).

To make a prediction for a test data, we use the predictive distribution. Let $\mathbf{x}_*$ be a test case and $f_* = f(\mathbf{x}_*) = \mathbf{x}_*{}^\top \mathbf{w}$. Then, the predictive distribution for f_* at $\mathbf{x}_*$ is given as

$$p(f_*|\mathbf{x}_*, X, \mathbf{y}) \sim N(\frac{1}{\sigma_n^2}\mathbf{x}_*{}^\top A^{-1} X\mathbf{y}, \mathbf{x}_*{}^\top A^{-1}\mathbf{x}_*).$$

We use the mean of the predictive distribution as our estimator for $f(\mathbf{x})$.

The Standard Linear Model

Predictive distribution

The predictive distribution is also Gaussian from our previous derivation.

The predictive variance $\mathbf{x}_*{}^\top A^{-1}\mathbf{x}_*$ is a quadratic form of the test input with the posterior covariance matrix. It implies that the predictive uncertainties grow with the magnitude of test input.

Projections of Inputs into Feature Space

Consider the function $\phi(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^N$ which maps an input $\mathbf{x}$ into an N dimensional feature space.

Let $\Phi(X) = [\phi(\mathbf{x_1}), \cdots, \phi(\mathbf{x_n})]$.

The model is now given as $f(\mathbf{x_i}) = \phi(\mathbf{x_i})^\top \mathbf{w}$, $y_i = f(\mathbf{x_i}) + \epsilon_i$ with $\epsilon_i \sim N(0, \sigma_n^2)$ and they are independent.

This model linearly approximates the outputs using feature functions which form **a basis of a high dimensional space**.

Why feature funtions?

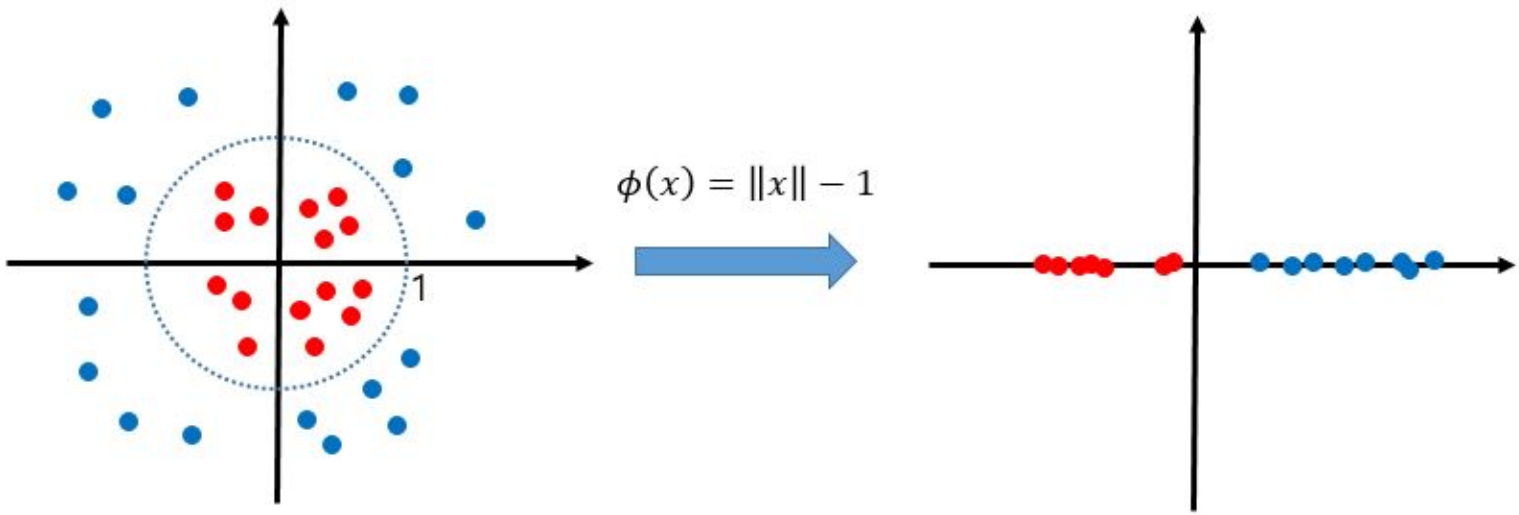


Figure: Feature function

Projections of Inputs into Feature Space

The predictive distribution

The analysis for this model is analogous to the standard linear model. Here, $X, \mathbf{x}_*$ are replaced by $\Phi(X)$ and $\phi(\mathbf{x}_*)$. For simplicity, we write $\Phi = \Phi(X), \phi_* = \phi(\mathbf{x}_*)$. Then, the predictive distribution becomes

$$f_*|x_*, X, \mathbf{y} \sim \mathcal{N}(\frac{1}{\sigma_n^2}\phi_*^\top A^{-1}\Phi\mathbf{y}, \phi_*^\top A^{-1}\phi_*)$$

where $A = \frac{1}{\sigma_n^2}\Phi\Phi^\top + \Sigma_p^{-1}$.

Alternatively, we can rewrite the predictive distribution in the following way

$$f_*|x_*, X, \mathbf{y} \sim \mathcal{N}\left(\phi_*^\top\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\mathbf{y}, \phi_*^\top(\Sigma_p - \Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\Phi^\top\Sigma_p)\phi_*\right)$$

where $K = \Phi^\top\Sigma_p\Phi$ (which is related with the Gaussian Process Regression).

Examples - Introduction

There are two examples which use the GP regression for prediction.

The first one is a noise free case where the objective function is given as $f(x) = x\cos(x)$.

The second one is a noisy case and the objective function is not given.

The training data for the second example are the numbers of **international airline passengers in USA per month** which are commonly used to test the performance of a regression operator.

Examples - Introduction

Python 3.5 is used to implement that examples. We use the following libraries

1. Numpy : It makes easy to do vector computation in python.
2. Scikit learn(sklearn) GaussianProcessRegressor, linear_model : The first one performs GP regression by optimizing the log-likelihood with respect to hyper-parameters.
   The second one is used to get linear regression which is needed to predict the mean function of test data.

There are also good tools for GP regression such as gpytorch and gpflow.

Noise free case

In this example, the objective function is given as $f(x) = x\cos(x)$. The training data are given as $\{(x_1, f(x_1)), \cdots, (x_n, f(x_n))\}$ where $x_1, \cdots, x_n$ are randomly chosen.
Here, we use the squared exponential covariance function.

$$k(x_i, x_j) = \sigma_f^2 \exp\Big(-\frac{1}{2l^2}(x_i - x_j)^2 \Big)$$

The following figure shows the GP prediction which comes from the randomly chosen points in $y = f(x)$.

Noise free case

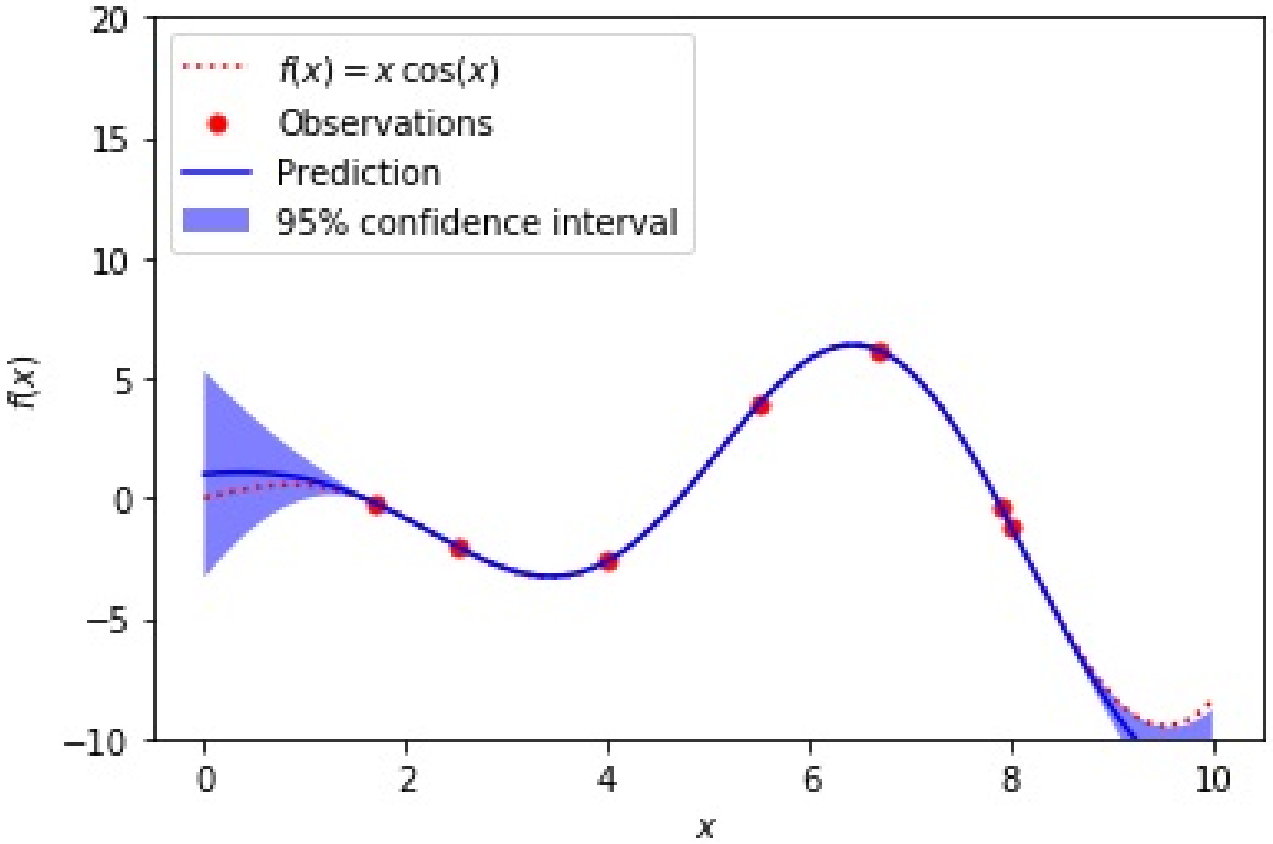


Figure: Noise-free Case

Noise free case

The red dotted line is our objective function. The blue line is our GP regressor.

Hyper-parameter tuning is done by optimizing the log-marginal likelihood function using an optimization method.

The above figure is obtained from the optimized kernel

$$k(x_i, x_j) = 8.06^2 \exp\Big(- \frac{1}{2(2.19)^2}(x_i - x_j)^2\Big)$$

with the log-marginal likelihood value $= -13.67$.

It is the case where the GP regressor is very accurate.

Noisy case

This example deals with a real data for the international airline passengers in USA from 1949.01 to 1960.12.

Here, we use the first 85 percent of data as a training data and use the remaining data as a **validation set**. The validation set is used to test the estimator.

Here, we use the following covariance function

$$k(x_i, x_j) = \sigma_1^2\big(x_i \cdot x_j + \sigma_2^2\big)\exp\Big(-2\big(\frac{\sin^2\big(\frac{\pi(x-x')^2}{\sigma_3}\big)}{\sigma_4^2}\big)\Big)$$

and for noisy data we use

$$k_y(x_i, x_j) = k(x_i, x_j) + \sigma_5^2\delta_{ij}.$$

Noisy case

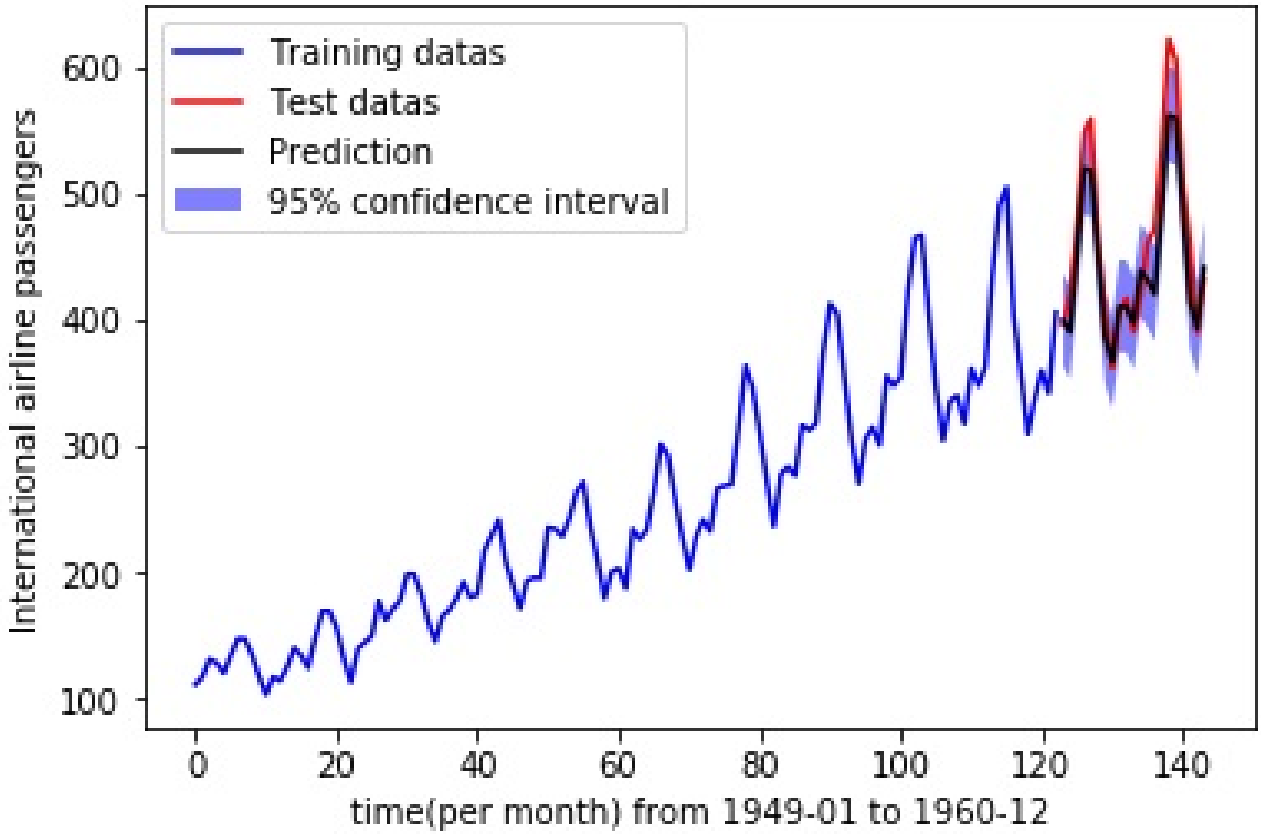


Figure: Noisy Case

Noisy case

The red line is our validation set (or, equivalently called as the test data) and the blue line is our training data. Black line is our GP regressor.

The above figure is obtained from the optimized kernel

$$k_y(x_i, x_j) = (0.94)^2\big(x_i \cdot x_j + 10^{-5}\big)\exp\Big(- 2\big(\frac{\sin^2\big(\frac{\pi(x-x')^2}{0.902}\big)}{(0.255)^2}\big)\Big) + 279\delta_{ij}$$

with the log-margianl likelihood value = -539.31.

Remarks

There are some remarks.

- In general, the log - marginal likelihood function may not be concave with respect to its parameters. So, there may be many local optimums and many gradient based optimization methods may fall into local optimum.
- In GP regression, **choosing a proper covariance function** is the main issue. For both examples, we choose our covariance functions heuristically. So, there may exist some other better covariance functions which make the GP regressor more accurate.

Remarks

- Recall that our GP regression is based on the GP with **zero mean function**. So, we need to centralize our training data. In our second example, the training data are centralized by subtracting its moving average

$$\mathrm{MA}(x_i) = \frac{x_{i-[k/2]} + \cdots + x_{i+[k/2]+1}}{k} \text{ (We use } k = 9)$$

  There are another centralization methods such as subtracting sample mean. You may get better results by choosing other window size k.
- Your predictor **should not train any information of validation set**.

Bayesian Models for Classification

Classification

We consider classification problems.

$\mathbf{x}$: data, y : the class label

When we consider $p(y, \mathbf{x})$, we have two approaches by Bayes' theorem.

- The generative approach considers $p(y, \mathbf{x}) = p(\mathbf{x}|y)p(y)$.
- The discriminative approach focusses on modeling $p(y|\mathbf{x})$ directly.

Classification Problems

The generative approach

For $y = \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_C$, the posterior probability of each class is

$$p(y|\mathbf{x}) = \frac{p(y, \mathbf{x})}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{\sum_{c=1}^{C} p(\mathcal{C}_c)p(\mathbf{x}|\mathcal{C}_c)}$$

A simple and common choice for the class-conditional density is

$$p(\mathbf{x}|\mathcal{C}_c) = \mathcal{N}(\mu_c, \Sigma_c).$$

However, it is unclear whether this choice is appropriate.

Classification Problems

The discriminative approach

Basic idea

$$\boxed{\mathbf{x}} \longrightarrow_{\mathcal{GP}} \boxed{f(\mathbf{x})} \longrightarrow_{\sigma} \boxed{\sigma(f(\mathbf{x}))}$$

For the binary case, we usually use a response function $\sigma(z)$ which squashes its argument into $[0, 1]$, guaranteeing a valid probabilistic interpretation.

The response function $\sigma(z)$ can be any sigmoid function. (A sigmoid function is a monotically increasing function mapping from $\mathbb{R}$ to [0,1].)

Classification Problems

The discriminative approach

Two examples for response functions are

- Linear logistic regression model

$$p(\mathcal{C}_i|\mathbf{x}) = \lambda(\mathbf{w}^T\mathbf{x}), \quad \text{where } \lambda(z) = \frac{1}{1+\exp(-z)}$$

- Probit regression model

$$p(\mathcal{C}_i|\mathbf{x}) = \Phi(\mathbf{w}^T\mathbf{x}), \quad \text{where } \Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx.$$

From now on we only consider the discriminative approach.

Classification Problems

Decision Theory for Classification

Let $\mathcal{L}(c, c')$ be the loss incurred by making decision c' if the true class is $\mathcal{C}_c$. Usually $\mathcal{L}(c, c') = 0$ when $c = c'$.Then the expected loss is

$$R_{\mathcal{L}}(c'|\mathbf{x}) = \sum_c \mathcal{L}(c, c')p(\mathcal{C}_c|\mathbf{x}).$$

The optimal class is

$$c_* = \underset{c}{\operatorname{argmin}}\, \mathcal{R}_{\mathcal{L}}(c|\mathbf{x}).$$

One common choice for the loss function is the zero-one loss, i.e.,

$$\mathcal{L}(c, c') = 1 - \delta_{cc'}.$$

Linear Models for Classification

$$\boxed{\mathbf{x}} \longrightarrow_{\mathbf{w}\sim\mathcal{N}} \boxed{\mathbf{w}^T\mathbf{x}} \longrightarrow_{\sigma} \boxed{\sigma(\mathbf{w}^T\mathbf{x})}$$

As we did in regression, we start with a linear model ($f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$).

Let the labels and the likelihood be

$$y = \pm 1$$

$$p(y = 1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}),$$

and we use the sigmoid function as the response function.

When $\sigma(z) = \lambda(z)$, the model is called a linear logistic regression.
When $\sigma(z) = \Phi(z)$, the model is called a linear probit regression.

Linear Models for Classification

For symmetric $\sigma(z)$, i.e., $\sigma(-z) = 1 - \sigma(z)$,

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \sigma(y_i f_i),$$

where $f_i = f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i$ because

$$p(y_i = 1|\mathbf{x}_i, \mathbf{w}) = \sigma(\mathbf{w}^T\mathbf{x}_i) = \sigma(f_i),$$

$$p(y_i = -1|\mathbf{x}_i, \mathbf{w}) = 1 - \sigma(\mathbf{w}^T\mathbf{x}) = 1 - \sigma(f_i) = \sigma(-f_i).$$

So we can write $p(y|\mathbf{x}, \mathbf{w})$ consistently regardless of the value of y.

Linear Models for Classification

Remark: The logit transformation is defined as

$$\text{logit}(\mathbf{x}) = \log \frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}.$$

For a linear logistic regression, $\text{logit}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$.

Linear Models for Classification

In binary classification, what we want to do is predicting y conditional on $\mathbf{x}$. To do this we assume the existence of a function $a(\mathbf{x})$ which models the logit as a function of $\mathbf{x}$. Thus

$$P(y = 1|\mathbf{x}, a(\mathbf{x})) = \sigma(a(\mathbf{x})).$$

There are two approaches to complete this model.

- One way is considering $a(\mathbf{x})$ as a parametrized function $a(\mathbf{x}; \mathbf{w})$ and give $\mathbf{w}$ a prior which is given below.
- The other approach is to model $a(\mathbf{x})$ using Gaussian process which is omitted here.

Linear Models for Classification

Given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, \ldots, n\}$, we assume that the labels are generated independently, conditional on $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$. Using the Gaussian prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$,

$$\begin{aligned} p(\mathbf{w}|\mathcal{D}, \mathbf{y}) &\propto p(\mathbf{y}|\mathbf{w}, \mathcal{D})p(\mathbf{w}|\mathcal{D}) \\ &= \prod_{i=1}^{n} \sigma(y_i \mathbf{w}^T \mathbf{x}_i) \exp(-\frac{1}{2}\mathbf{w}^T \Sigma_p^{-1} \mathbf{w}) \end{aligned}$$

So

$$\log p(\mathbf{w}|\mathcal{D}, \mathbf{y}) =^c -\frac{1}{2}\mathbf{w}^T \Sigma_p^{-1} \mathbf{w} + \sum_{i=1}^{n} \log(\sigma(y_i \mathbf{w}^T \mathbf{x}_i))$$

Linear Models for Classification

Maximum Likelihood

For $\sigma(z) = \lambda(z)$, the log posterior is a concave function of $\mathbf{w}$ for fixed $\mathcal{D}$.

Proposition 1

$f(\mathbf{w}) = -\frac{1}{2}\mathbf{w}^T\Sigma_p^{-1}\mathbf{w}$ is concave in $\mathbf{w}$.

Proposition 2

$\log(\lambda(y\mathbf{w}^T\mathbf{x}))$ is concave where $\lambda(z) = \frac{1}{1+\exp(-z)}$.

For $\sigma(z) = \Phi(z)$, the log posterior is also concave.

Proposition 3

$\log(\Phi(y\mathbf{w}^T\mathbf{x}))$ is concave.

Linear Models for Classification

Maximum Likelihood

Since the log posterior is a concave function in $\mathbf{w}$, it is relatively easy to find its unique maximum. We can use the Newton's method to find the maximum.

Linear Models for Classification

Predictions

To make predictions based the training set $\mathcal{D}$ for a test point $\mathbf{x}_*$, we have

$$p(y_* = 1|\mathbf{x}_*, \mathcal{D}) = \int p(y_* = 1|\mathbf{w}, \mathbf{x}_*)p(\mathbf{w}|\mathcal{D})\ d\mathbf{w}.$$

Linear Models for Classification

In the multi-class case, we use the softmax function

$$p(y = \mathcal{C}_c|\mathbf{x}, W) = \frac{\exp(\mathbf{x}^T \mathbf{w}_c)}{\sum_{c'} \exp(\mathbf{x}^T \mathbf{w}_{c'})}$$

where $\mathbf{w}_c$ is the weight vector for class c, and $W = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$.

The corresponding log likelihood is of the form

$$\sum_{i=1}^{n} \sum_{c=1}^{C} \delta_{c,y_i} [\mathbf{x}_i^T \mathbf{w}_c - \log(\sum_{c'} \exp(\mathbf{x}_i^T \mathbf{w}_{c'}))]$$

which is also a concave function of W.

In Gaussian Process Classification, we use Gaussian processes and there are two approaches.

- Laplace Approximation
- Expectation Propagation

For the details, please refer to

- Rasmussen and C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.

Example - Introduction

There are two examples which use the GP regression for classification.

The first one just a toy problem on our textbook pages 60 and 61.

The second one is a multi-class classification which classifies the types of iris. The training data for the second example are **the iris data sets** which are commonly used to test the performance of classification.

Example - Introduction

Python 3.5 is also used to implement the examples. We use the following libraries

- 1. Numpy : It makes easy to do vector computation in python.
- 2. Scikit learn(sklearn) GaussianProcessClassifier : It performs GP classification by optimizing the log-likelihood with respect to hyper-parameters. It uses the logistic response function and **Laplace approximation** to approximate the non-Gaussian likelihood.

A Toy Problem

15 data points are randomly chosen in the square $[0, 1]^2$. The 2 classes are labeled as x (-1) and o $(+1)$. We use the squared exponential covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma^2 \exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2l^2} \Big)$$

The following figure shows the contour plots of the predictive probability $E_q[\pi(\mathbf{x}_*)|\mathbf{f}]$ and the training data points.

A Toy Problem

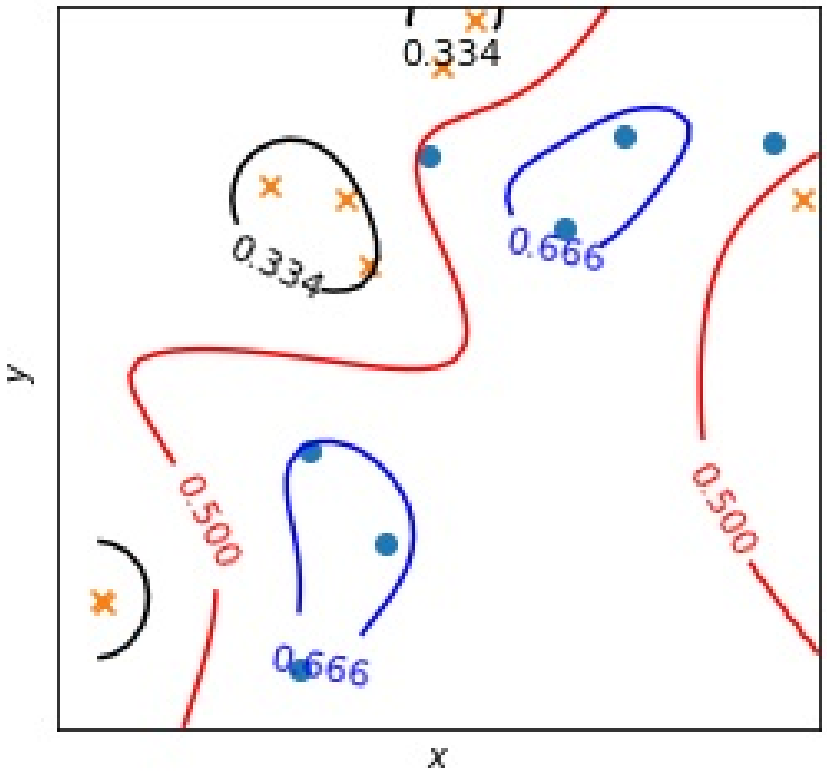


Figure: Classification results

A Toy Problem

Here, test data are all points in $[0, 1]^2$.

The value in each contour denotes the predictive probability of class $+1$.

Similarly as in GP regression, Hyper-parameter tuning is done by optimizing the approximated log-marginal likelihood function.

The above figure is obtained from the optimized kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = 1.64^2 \exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2(0.116)^2} \Big)$$

with the log-marginal likelihood value $= -10.142$.

Classification of iris data

Introduction

This example deals with a real data on the types of the irises. There are 3 main types of irises: setosa, versicolor, and virginica. The iris data is composed with 150 data of irises with their types and their lengths and widths of sepals and petals. The 3 classes, 'setosa', 'versicolor', and 'virginica' are labeled as red, green and blue.

We want to classify a given iris by using its lengths and widths of the sepal and petal using GP classification.

Classification of iris data

Introduction

Since each data has 4 features (the lengths and widths of the sepal and petal), it may be hard to get a graphical demonstration. So, we select 2 features: the widths and lengths of the sepal.

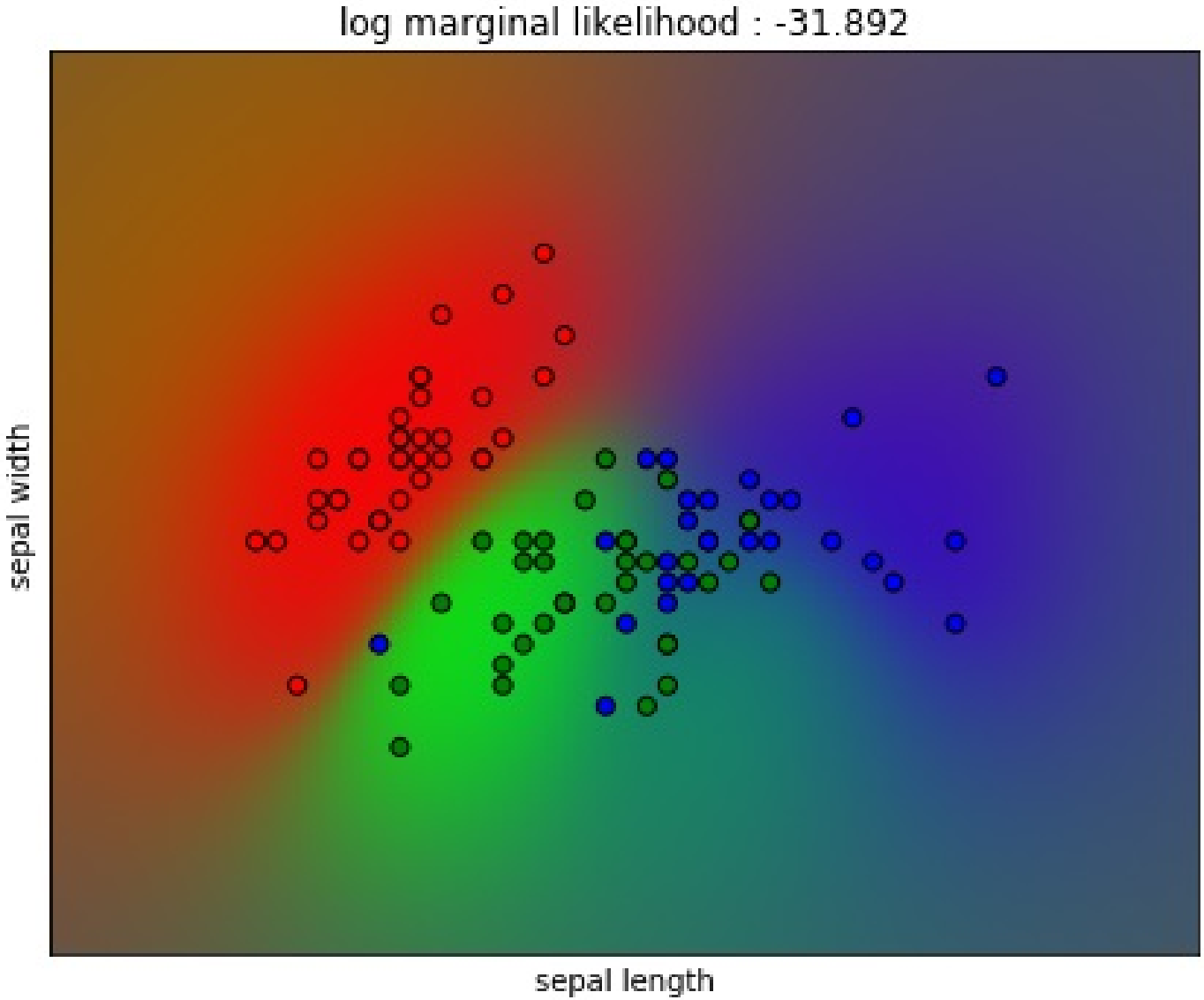
Next, to test our classifier, we **divide the data sets into 2 parts of size 100 and 50. 100 data are used to train our model, and 50 data are used for validation**.

We again use the squared exponential covariance function with noise term

$$k_y(\mathbf{x}_i, \mathbf{x}_j) = \sigma_1^2 \exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2l^2} \Big) + \sigma_2^2 \delta_{ij}.$$

The following figure shows the locations of training sets and classification results with respect to the widths and lengths of sepals.

Classification of iris data



Classification of iris data

Here, the test data are all points in the 2d box
$\big[\min_{\text{training data}}(\text{sepal length}) - 1, \max_{\text{training data}}(\text{sepal length}) + 1\big] \times$
$\big[\min(\text{sepal width}) - 1, \max(\text{sepal width}) + 1\big]$.

The above figure is obtained from the optimized kernel

$$k_y(\mathbf{x}_i, \mathbf{x}_j) = (8.84)^2 \exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2(2.71)^2} + (2.66)^2 \delta_{ij} \Big)$$

with log-marginal likelihood value = -31.892

Classification of iris data

From the above figure, we can say that our GP classifier works well for classifying setosa. However, it doesn't work well in classifying versicolor and virginica. Actually, its average error rate with respect to validation is 20.5 percent. Does this mean that our GP classification works poorly? Let's consider the following figure.

Classification of iris data

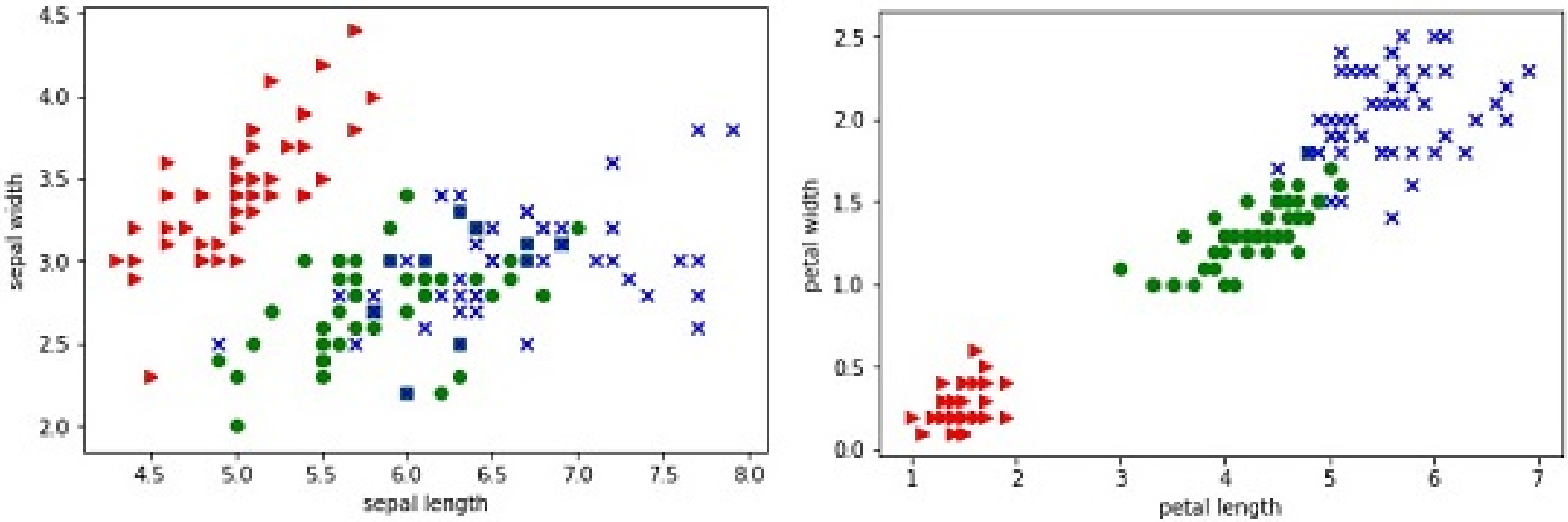


Figure: Comparison between two different feature sets

Classification of iris data

The right figure is the locations of training sets with respect to the widths and lengths of **petals**. By considering both, we can observe that the **widths and lengths of petals** are better feature sets for learning the GP classifier. By using the new features we get better results which are provided below.

Classification of iris data

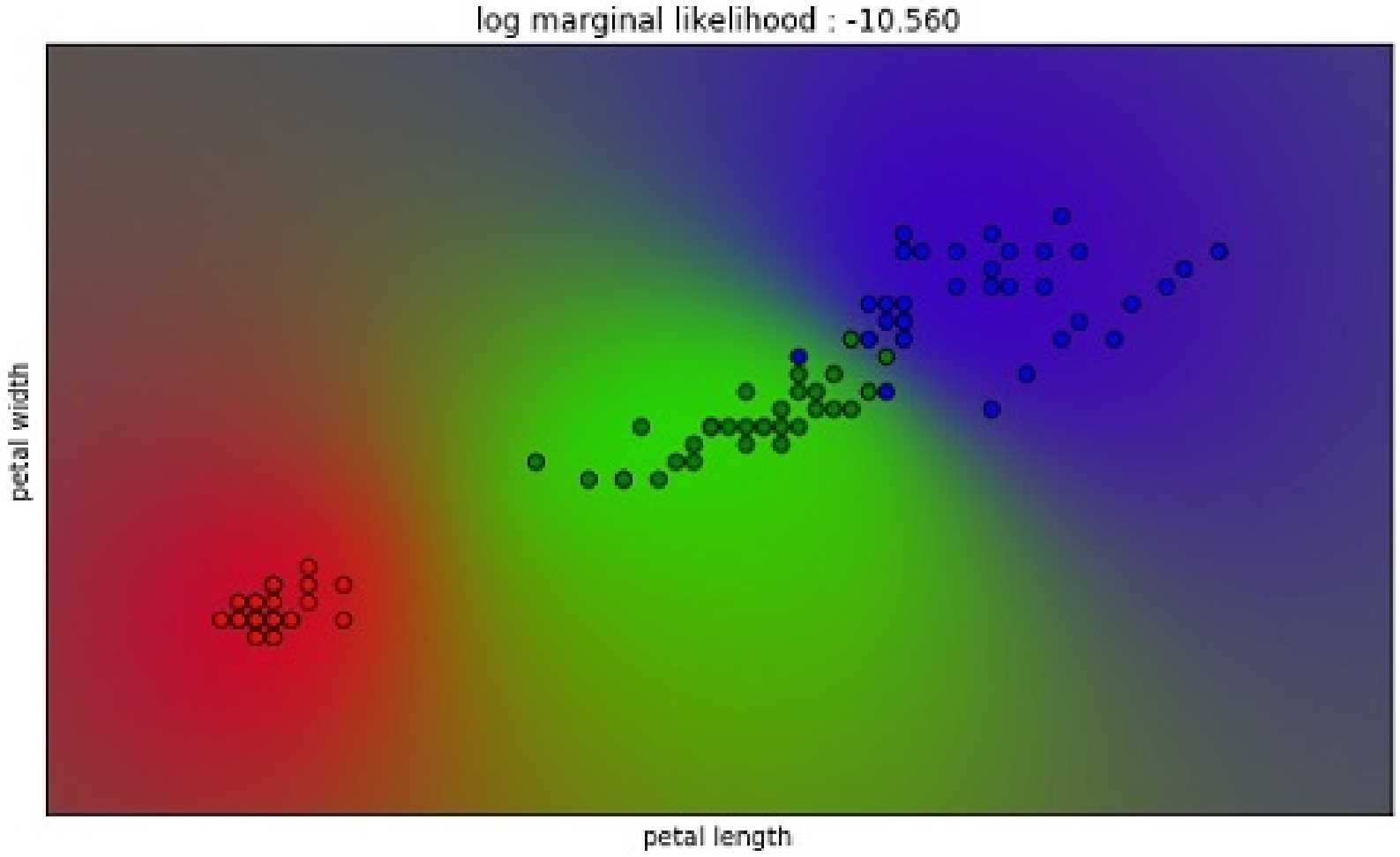


Figure: Classification with the widths and lengths of petals

Classification of iris data

The test data are all points in the 2d box
$\left[\min_{\text{training data}}(\text{Petal length}) - 1, \max_{\text{training data}}(\text{Petal length}) + 1\right] \times$
$\left[\min(\text{Petal width}) - 1, \max(\text{Petal width}) + 1\right]$.

The above figure is obtained from the optimized kernel

$$k_y(\mathbf{x}_i, \mathbf{x}_j) = (7.77)^2 \exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2(4.49)^2} + (5.55)^2 \delta_{ij} \Big)$$

with log-marginal likelihood value $=$ -10.560 which is much larger than the previous case. Moreover, its averaged error rate with respect to a validation set is only 4 percent.

References

- The iris data : http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html#sklearn.datasets.load_iris
- K.P. Murphy, Machine Learning, The MIT Press, 2012.
- C.E. Rasmussen and C.K.I. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.