ECE521 Tutorial 2

Regression, GPs, Assignment 1

ECE521 Winter 2016

Credits to Alireza Makhzani, Alex Schwing, Rich Zemel for the slides.

Outline

Types of Learning

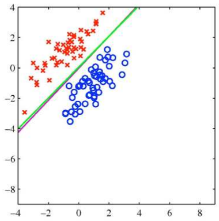
Consider observing a series of input vectors:

$$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \ldots.$$

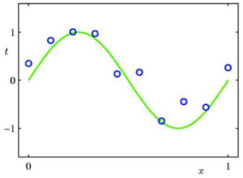
- **Supervised Learning:** We are also given target outputs (labels, responses): $y_1, y_2, \ldots$, and the goal is to predict correct output given a new input.
- **Unsupervised Learning:** The goal is to build a statistical model of **x**, which can be used for making predictions, decisions.
- **Reinforcement Learning:** the model (agent) produces a set of actions: a_1, $a_2, \ldots$ that affect the state of the world, and received rewards r_1, $r_2 \ldots$ The goal is to learn actions that maximize the reward (we will not cover this topic in this course).
- **Semi-supervised Learning:** We are given only a limited amount of labels, but lots of unlabeled data.

Supervised Learning

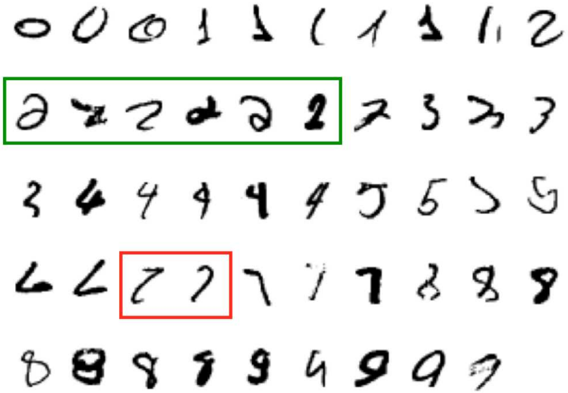
Classification: target outputs y_i are discrete class labels. The goal is to correctly classify new inputs.



Regression: target outputs y_i are continuous. The goal is to predict the output given new inputs.



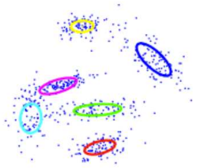
Handwritten Digit Classification



Unsupervised Learning

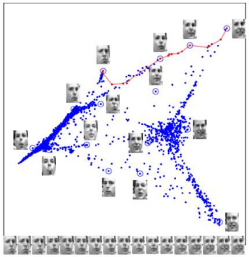
The goal is to construct statistical model that finds useful representation of data:

- Clustering
- Dimensionality reduction
- Modeling the data density
- Finding hidden causes (useful explanation) of the data



Unsupervised Learning can be used for:

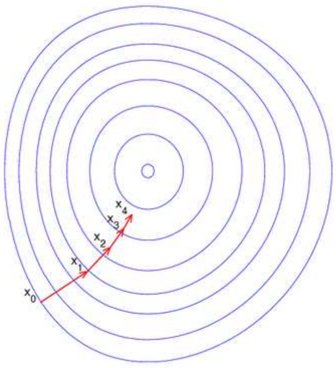
- Structure discovery
- Anomaly detection / Outlier detection
- Data compression, Data visualization
- Used to aid classification/regression tasks



Gradient Descent Algorithm

Gradient descent is based on the observation that a function decreases fastest if one goes in the direction of negative gradient.

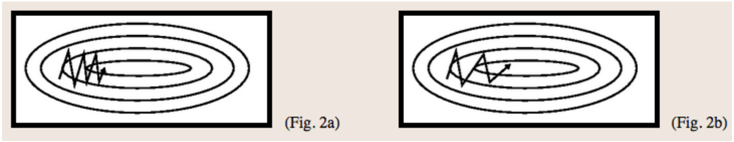
$$x_{n+1} = x_n - \mu_n \nabla f(x_n)$$



Momentum

Momentum simply adds a fraction m of the previous weight update to the current one.

$$\begin{aligned} \boldsymbol{v}_k &= m_k \boldsymbol{v}_{k\text{-}1} - \eta_k \nabla f(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k + \boldsymbol{v}_k \end{aligned} \tag{1}$$

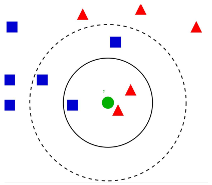


Validation Set

If the data is plentiful, we can divide the dataset into three subsets:

- Training Data: used to fitting/learning the parameters of the model.
- Validation Data: not used for learning but for selecting the model, or choosing the amount of regularization that works best.
- Test Data: used to get performance of the final model.

K-nearest neighbour



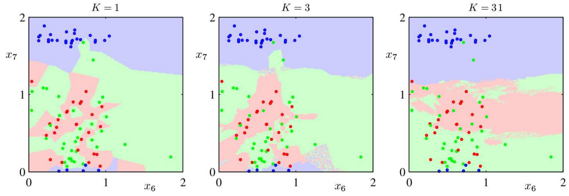


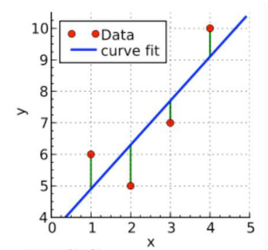
Figure 2.28 Plot of 200 data points from the oil data set showing values of x_6 plotted against x_7, where the red, green, and blue points correspond to the 'laminar', 'annular', and 'homogeneous' classes, respectively. Also shown are the classifications of the input space given by the K-nearest-neighbour algorithm for various values of K.

Linear Least Squares

- Given a vector of d-dimensional inputs $\mathbf{x} = (x_1, x_2, ..., x_d)^T$, we want to predict the target (response) using the linear model:

$$y(x, \mathbf{w}) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_d x_d = w_0 + \sum_{j=1}^{d} w_j x_j.$$

One option is to minimize **the sum of the squares of the errors** between the predictions $y(\mathbf{x}_n, \mathbf{w})$ for each data point x_n and the corresponding real-valued targets t_n.

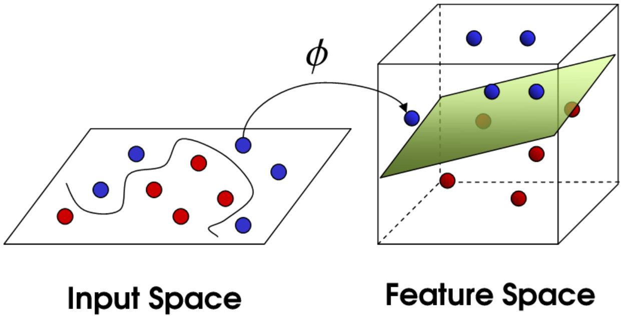


Source: Wikipedia

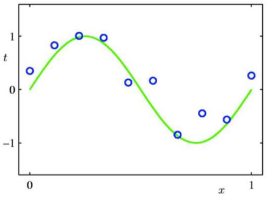
Loss function: sum-of-squared error function:

$$\begin{aligned} E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n^T\mathbf{w} - t_n)^2 \\ &= \frac{1}{2}(\mathbf{X}\mathbf{w} - \mathbf{t})^{\mathbf{T}}(\mathbf{X}\mathbf{w} - \mathbf{t}). \end{aligned}$$

Nonlinearity

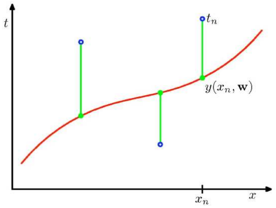


Polynomial Curve Fitting



$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + ... + w_M x^M = \sum_{j=0}^{M} w_j x^j.$$

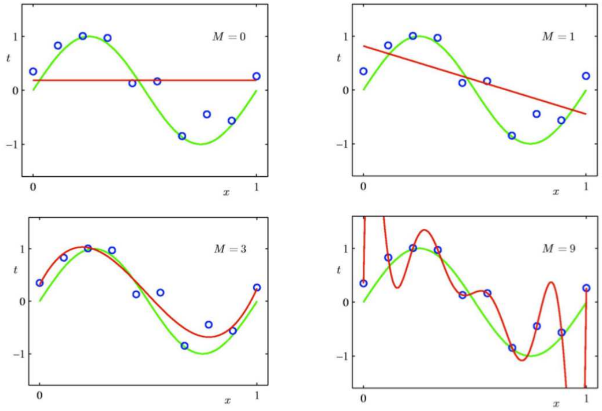
Note: the polynomial function is a nonlinear function of x, but it is a linear function of the coefficients **w** $\rightarrow$ **Linear Models**.



Loss function: sum-of-squared error function:

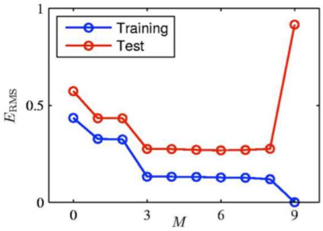
$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} (y(x_n, \mathbf{w}) - t_n)^2.$$

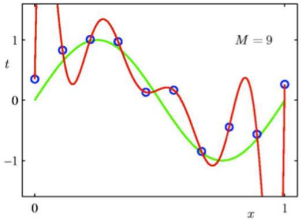
Some Fits to the Data



Overfitting

- Consider a separate **test set** containing 100 new data points generated using the same procedure that was used to generate the training data.



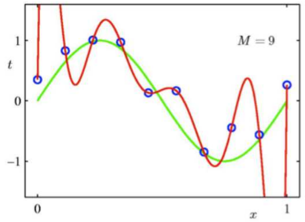


Generalization

- The goal is achieve good **generalization** by making accurate predictions for new test data that is not known during learning.
- Choosing the values of parameters that minimize the loss function on the training data may not be the best option.
- We would like to model the true regularities in the data and ignore the noise in the data:
  - It is hard to know which regularities are real and which are accidental due to the particular training examples we happen to pick.

Overfitting

	$M = 0$	$M = 1$	$M = 3$	$M = 9$
$w_0^\star$	0.19	0.82	0.31	0.35
$w_1^\star$		-1.27	7.99	232.37
$w_2^\star$			-25.43	-5321.83
$w_3^\star$			17.37	48568.31
$w_4^\star$				-231639.30
$w_5^\star$				640042.26
$w_6^\star$				-1061800.52
$w_7^\star$				1042400.18
$w_8^\star$				-557682.99
$w_9^\star$				125201.43



- As M increases, the magnitude of coefficients gets larger.
- For M=9, the coefficients have become finely tuned to the data.
- Between data points, the function exhibits large oscillations.

More flexible polynomials with larger M tune to the random noise on the target values.

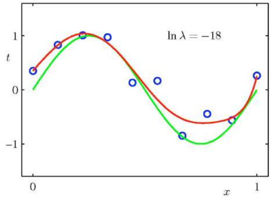
Preventing Overfitting

penalized error function

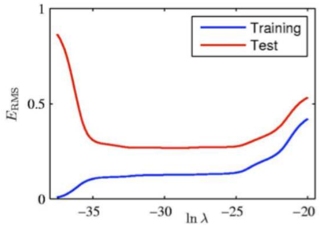
target value

regularization parameter

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n,\mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$$



Regularization

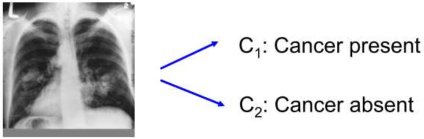


	$\ln\lambda = -\infty$	$\ln\lambda = -18$	$\ln\lambda = 0$
$w_0^\star$	0.35	0.35	0.13
$w_1^\star$	232.37	4.74	-0.05
$w_2^\star$	-5321.83	-0.77	-0.06
$w_3^\star$	48568.31	-31.97	-0.05
$w_4^\star$	-231639.30	-3.89	-0.03
$w_5^\star$	640042.26	55.28	-0.02
$w_6^\star$	-1061800.52	41.32	-0.01
$w_7^\star$	1042400.18	-45.95	-0.00
$w_8^\star$	-557682.99	-91.53	0.00
$w_9^\star$	125201.43	72.68	0.01

Classification

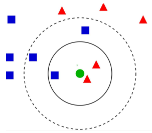
- The goal of classification is to assign an input **x** into one of K discrete classes C_k, where k=1,..,K.
- Typically, each input is assigned only to one class.

- Example: The input vector **x** is the set of pixel intensities, and the output variable t will represent the presence of cancer, class C_1, or absence of cancer, class C_2.



x -- set of pixel intensities

K-nearest neighbour



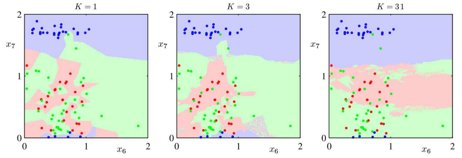


Figure 2.28 Plot of 200 data points from the oil data set showing values of x_6 plotted against x_7, where the red, green, and blue points correspond to the 'laminar', 'annular', and 'homogeneous' classes, respectively. Also shown are the classifications of the input space given by the K-nearest-neighbour algorithm for various values of K.

$$P(C_k) = \frac{K_k}{K}$$

Linear Classification

• The goal of classification is to assign an input **x** into one of K discrete classes C_k, where k=1,..,K.

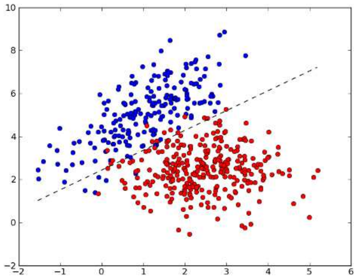
• The input space is divided into decision regions whose boundaries are called decision boundaries or decision surfaces.

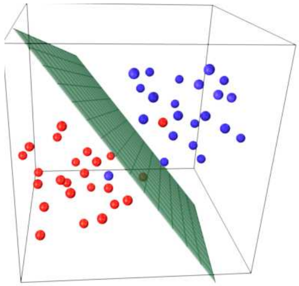
• We will consider linear models for classification. Remember, in the simplest linear regression case, the model is linear in parameters:

$$y(\mathbf{x}, \mathbf{w}) = \underbrace{\mathbf{x}^T \mathbf{w} + w_0}_{\text{adaptive parameters}}. \qquad y(\mathbf{x}, \mathbf{w}) = \underbrace{f}_{\text{fixed nonlinear function: activation function}}(\mathbf{x}^T \mathbf{w} + w_0).$$

• For classification, we need to predict discrete class labels, or posterior probabilities that lie in the range of (0,1), so we use a nonlinear function.

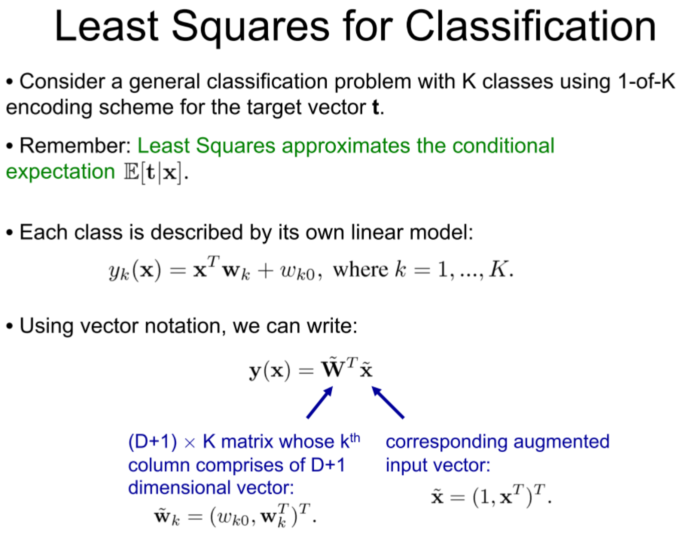
Linear Classification





$$\mathbf{x}^T\mathbf{w} + w_0 = w_0 + w_1 x_1 + w_2 x_2$$

Linear regression for classification



Least Squares for Classification

- Consider a general classification problem with K classes using 1-of-K encoding scheme for the target vector **t**.
- Remember: Least Squares approximates the conditional expectation $\mathbb{E}[\mathbf{t}|\mathbf{x}]$.
- Each class is described by its own linear model:

$$y_k(\mathbf{x}) = \mathbf{x}^T \mathbf{w}_k + w_{k0}, \text{ where } k = 1, ..., K.$$

- Using vector notation, we can write:

$$\mathbf{y}(\mathbf{x}) = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}$$

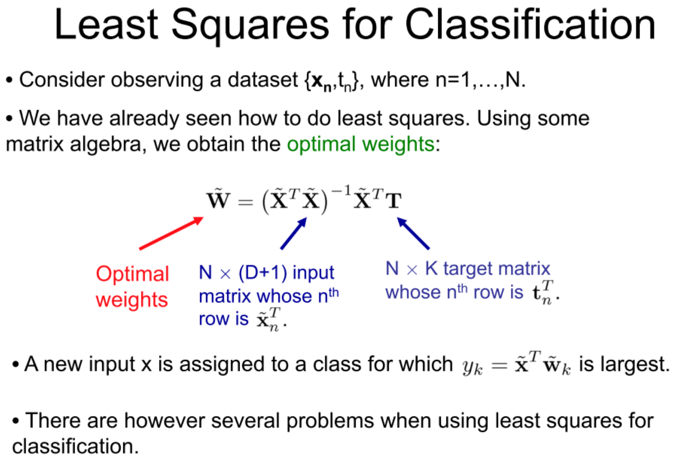
(D+1) $\times$ K matrix whose kth column comprises of D+1 dimensional vector:

$$\tilde{\mathbf{w}}_k = (w_{k0}, \mathbf{w}_k^T)^T.$$

corresponding augmented input vector:

$$\tilde{\mathbf{x}} = (1, \mathbf{x}^T)^T.$$

Linear regression for classification



Least Squares for Classification

- Consider observing a dataset $\{\mathbf{x}_n, t_n\}$, where n=1,…,N.
- We have already seen how to do least squares. Using some matrix algebra, we obtain the optimal weights:

$$\tilde{\mathbf{W}} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{T}$$

Optimal weights

N $\times$ (D+1) input matrix whose nth row is $\tilde{\mathbf{x}}_n^T$.

N $\times$ K target matrix whose nth row is $\mathbf{t}_n^T$.

- A new input x is assigned to a class for which $y_k = \tilde{\mathbf{x}}^T\tilde{\mathbf{w}}_k$ is largest.
- There are however several problems when using least squares for classification.

Methods for Regression

We are given n data points $\boldsymbol{x}$ and corresponding regression targets y:

$$\mathcal{D} = \{\boldsymbol{x}^{(i)}, y_i\}_{i=1}^{n}$$

Two approaches

1. Restrict the class of functions (e.g., only linear)
2. Assign a probability to every possible function

1. Restrict the class of functions (e.g., only linear)

- Use data to learn the parameters $\boldsymbol{w}$ of a parametric model $p(y \mid \boldsymbol{x}, \boldsymbol{w})$
- Use parameters and test data $\boldsymbol{x}_*$ to obtain prediction $y_* = \arg\max_y p(y \mid \boldsymbol{x}_*, \boldsymbol{w})$

Examples:

- Logistic regression
- Support vector machine
- Neural networks

Problem: How to find the right model?

2. Assign a probability to every possible function

- Use data to design a posterior $p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})$
- Use posterior and test data $\boldsymbol{x}_*$ to obtain predictive distribution (average over all models)

$$p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_* \mid \boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{w}$$

Note difference to non-Bayesian approach: single parameter vs. averaging

Problem: How to assign a value to every possible function?

Solution: Gaussian Processes

Weight-space view: Bayesian treatment of linear model

Dataset:

$$\mathcal{D} = \left\{ \boldsymbol{x}^{(i)}, y_i \right\}_{i=1}^{N} \qquad \boldsymbol{x}^{(i)} \in \mathbb{R}^D, y_i \in \mathbb{R}$$

$$\boldsymbol{X} = \left[\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)} \right], \qquad \boldsymbol{y} = [y_1, \ldots, y_N]^\top$$

Given $\mathcal{D}$ we now

- Compute posterior $p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})$
- Compute predictive distribution $p(\boldsymbol{y}_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$

Posterior for standard linear model

$$y = f(\boldsymbol{x}) + \epsilon \qquad f(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{w} \qquad \epsilon \sim \mathcal{N}(0, \sigma_n^2)$$

Likelihood:

$$\begin{aligned}
p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) &= \prod_i p(y_i \mid \boldsymbol{x}^{(i)}, \boldsymbol{w}) \\
&= \prod_i \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left(-\frac{1}{2\sigma_n^2}(y_i - \boldsymbol{w}^\top \boldsymbol{x}^{(i)})\right) \\
&= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}\|\boldsymbol{y} - \boldsymbol{X}^\top \boldsymbol{w}\|_2^2\right) \\
&= \mathcal{N}(\boldsymbol{X}^\top \boldsymbol{w}, \sigma_n^2 \boldsymbol{I})
\end{aligned}$$

Prior:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p)$$

Posterior for standard linear model

Likelihood:

$$p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{X}^\top \boldsymbol{w}, \sigma_n^2 \boldsymbol{I})$$

Prior:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p)$$

Expression for posterior:

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \boldsymbol{X})}$$

Expression for marginal likelihood:

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) p(\boldsymbol{w}) d\boldsymbol{w}$$

Posterior for standard linear model

Recall: Bayes' Theorem for multivariate Gaussian

Given
$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\mu, \boldsymbol{\Lambda}^{-1}) \\ p(\boldsymbol{y} \mid \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{L}^{-1}) \end{aligned}$$

We obtain
$$\begin{aligned} p(\boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{A}\mu + \boldsymbol{b}, \boldsymbol{L}^{-1} + \boldsymbol{A}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}^\top\right) \\ p(\boldsymbol{x} \mid \boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{\Gamma}\left(\boldsymbol{A}^\top \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{b}) + \boldsymbol{\Lambda}\mu\right), \boldsymbol{\Gamma}\right) \\ \boldsymbol{\Gamma} &= \left(\boldsymbol{\Lambda} + \boldsymbol{A}^\top \boldsymbol{L}\boldsymbol{A}\right)^{-1} \end{aligned}$$

In our case:
$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) &= \mathcal{N}(\boldsymbol{X}^\top \boldsymbol{w}, \sigma_n^2 \boldsymbol{I}) \\ p(\boldsymbol{w}) &= \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p) \end{aligned}$$

Posterior:
$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{\Gamma}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{\Gamma}) \qquad \boldsymbol{\Gamma} = \left(\boldsymbol{\Sigma}_p^{-1} + \frac{1}{\sigma_n^2}\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}$$

Posterior for standard linear model

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{\Gamma}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{\Gamma}) \qquad \boldsymbol{\Gamma} = \left(\boldsymbol{\Sigma}_p^{-1} + \frac{1}{\sigma_n^2}\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}$$

For Gaussian distributions: mean = mode (MAP estimate)

Note similarity to Ridge regression

Given $\mathcal{D}$ we managed to

- Compute posterior $p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})$

We still need to

- Compute predictive distribution

$$p(\boldsymbol{y}_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(\boldsymbol{y}_* \mid \boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{w}$$

Predictive distribution

Recall: Bayes' Theorem for multivariate Gaussian

Given
$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\mu, \boldsymbol{\Lambda}^{-1}) \\ p(\boldsymbol{y} \mid \boldsymbol{x}) &= \mathcal{N}(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}, \boldsymbol{L}^{-1}) \end{aligned}$$

We obtain
$$\begin{aligned} p(\boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{A}\mu + \boldsymbol{b}, \boldsymbol{L}^{-1} + \boldsymbol{A}\boldsymbol{\Lambda}^{-1}\boldsymbol{A}^\top\right) \\ p(\boldsymbol{x} \mid \boldsymbol{y}) &= \mathcal{N}\left(\boldsymbol{\Gamma}\left(\boldsymbol{A}^\top \boldsymbol{L}(\boldsymbol{y} - \boldsymbol{b}) + \boldsymbol{\Lambda}\mu\right), \boldsymbol{\Gamma}\right) \\ \boldsymbol{\Gamma} &= \left(\boldsymbol{\Lambda} + \boldsymbol{A}^\top \boldsymbol{L}\boldsymbol{A}\right)^{-1} \end{aligned}$$

Prediction: average over all parameter values, weighted by posterior

$$\begin{aligned} p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) &= \int p(y_* \mid \boldsymbol{x}_*, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{w} \\ &= \int \mathcal{N}(\boldsymbol{x}_*^\top \boldsymbol{w}, \sigma_n^2) \mathcal{N}(\frac{1}{\sigma_n^2} \boldsymbol{\Gamma}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{\Gamma}) d\boldsymbol{w} \\ &= \mathcal{N}(\frac{1}{\sigma_n^2} \boldsymbol{x}_*^\top \boldsymbol{\Gamma}\boldsymbol{X}\boldsymbol{y}, \sigma_n^2 + \boldsymbol{x}_*^\top \boldsymbol{\Gamma}\boldsymbol{x}_*) \end{aligned}$$

Summary:

- Prior: $p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_p)$
- Likelihood: $p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{X}^\top \boldsymbol{w}, \sigma_n^2 \boldsymbol{I})$
- Posterior: $p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{\Gamma}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{\Gamma})$ with $\boldsymbol{\Gamma} = \left(\boldsymbol{\Sigma}_p^{-1} + \frac{1}{\sigma_n^2}\boldsymbol{X}\boldsymbol{X}^\top\right)^{-1}$
- Predicitive distribution:

$$p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{x}_*^\top \boldsymbol{\Gamma}\boldsymbol{X}\boldsymbol{y}, \sigma_n^2 + \boldsymbol{x}_*^\top \boldsymbol{\Gamma}\boldsymbol{x}_*)$$

Multivariate Gaussian Distribution

$$
\begin{aligned}
&p(x_1, \dots, x_n \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \\
&= \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu})\right) \\
&= \frac{1}{\sqrt{(2\pi)^n \det(\boldsymbol{\Lambda})^{-1}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Lambda}(\boldsymbol{x} - \boldsymbol{\mu})\right) \\
&= p(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})
\end{aligned}
$$

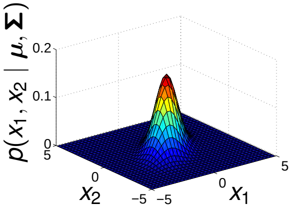
$\boldsymbol{x}$	$\in \mathbb{R}^n$	
$\boldsymbol{\mu}$	$\in \mathbb{R}^n$:	mean vector
$\boldsymbol{\Sigma}$	$\in \mathbb{R}^{n \times n}$:	covariance matrix
$\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$	$\in \mathbb{R}^{n \times n}$:	precision matrix

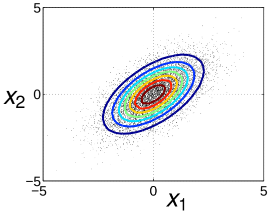
Short hand:

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$$

Samples:

$$\boldsymbol{\Sigma}^{1/2} \cdot \text{randn(2,10000)} + \boldsymbol{\mu}$$





How do the plots look like if x_1 and x_2 are independent random variables? What are the entries of the covariance matrix?

Tools to deal with Gaussian distributions

Completing the square:

$$-\frac{1}{2}(\boldsymbol{x}-\mu)^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu) = -\frac{1}{2}\boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \boldsymbol{x}^\top \boldsymbol{\Sigma}^{-1}\mu + \text{const}$$

- express a given quadratic form as shown on the right hand side
- read of $\boldsymbol{\Sigma}$ and μ

Example to practice completing the square:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix}^{-1} \right)$$

Suppose $\boldsymbol{x}_b$ given, what is $p(\boldsymbol{x}_a \mid \boldsymbol{x}_b)$?

$$-\frac{1}{2}\left(\begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix} \right)^\top \begin{bmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{bmatrix} \left(\begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix} - \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix} \right) =$$

$$-\frac{1}{2}\boldsymbol{x}_a^\top \boldsymbol{\Lambda}_{aa} \boldsymbol{x}_a + \boldsymbol{x}_a^\top \left(\boldsymbol{\Lambda}_{aa}\boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{ab}(\boldsymbol{x}_b - \boldsymbol{\mu}_b) \right) + \text{const}$$

- Covariance: $\boldsymbol{\Lambda}_{aa}^{-1}$
- Mean: $\boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\boldsymbol{x}_b - \boldsymbol{\mu}_b)$

$$p(\boldsymbol{x}_a \mid \boldsymbol{x}_b) = \mathcal{N}(\boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\boldsymbol{x}_b - \boldsymbol{\mu}_b), \boldsymbol{\Lambda}_{aa}^{-1})$$

Express $p(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\boldsymbol{\mu}_a - \boldsymbol{\Lambda}_{aa}^{-1}\boldsymbol{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b), \boldsymbol{\Lambda}_{aa}^{-1})$ in terms of covariance matrix

$$\begin{pmatrix} \boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb} \end{pmatrix}^{-1} = \begin{pmatrix} \boldsymbol{\Lambda}_{aa} & \boldsymbol{\Lambda}_{ab} \\ \boldsymbol{\Lambda}_{ba} & \boldsymbol{\Lambda}_{bb} \end{pmatrix}$$

Matrix identity:

$$\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{M} & -\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{M} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{M}\mathbf{B}\mathbf{D}^{-1} \end{pmatrix}$$

with

$$\mathbf{M} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}$$

$$\begin{aligned} \boldsymbol{\Lambda}_{aa} &= (\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1} \\ \boldsymbol{\Lambda}_{ab} &= -(\boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})^{-1}\boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1} \end{aligned}$$

$$p(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b), \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})$$

Conditioning of Multivariate Gaussian

$$\mathbf{x} = \left[\begin{array}{c} \mathbf{x}_a \\ \mathbf{x}_b \end{array}\right] \sim \mathcal{N}\left(\left[\begin{array}{c} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{array}\right], \left[\begin{array}{cc} \mathbf{\Lambda}_{aa} & \mathbf{\Lambda}_{ab} \\ \mathbf{\Lambda}_{ba} & \mathbf{\Lambda}_{bb} \end{array}\right]^{-1}\right)$$
$$\sim \mathcal{N}\left(\left[\begin{array}{c} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{array}\right], \left[\begin{array}{cc} \mathbf{\Sigma}_{aa} & \mathbf{\Sigma}_{ab} \\ \mathbf{\Sigma}_{ba} & \mathbf{\Sigma}_{bb} \end{array}\right]\right)$$

Conditional:

$$p(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\boldsymbol{\mu}_a - \mathbf{\Lambda}_{aa}^{-1}\mathbf{\Lambda}_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b), \mathbf{\Lambda}_{aa}^{-1})$$
$$p(\mathbf{x}_a \mid \mathbf{x}_b) = \mathcal{N}(\boldsymbol{\mu}_a + \mathbf{\Sigma}_{ab}\mathbf{\Sigma}_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b), \mathbf{\Sigma}_{aa} - \mathbf{\Sigma}_{ab}\mathbf{\Sigma}_{bb}^{-1}\mathbf{\Sigma}_{ba})$$

TF Tips

- tf.argmin(input, axis=None, name=None, dimension=None), tf.argmax(input, axis=None, name=None, dimension=None) can find the indices of the biggest/smallest elements in the tensor.
- tf.nn.embedding_lookup(params, ids, partition_strategy='mod', name=None, validate_indices=True, max_norm=None) can look up ids in a list of tensors.
- matplotlib is a package for generating plots similar to Matlab. Pyplot is a useful subpackage under matplotlib.