1 Basics of Statistical Learning

1.1 Introduction

In order to provide rational decision making, we focus on four key aspects:

1. Precise predictions
2. Data-driven predictions
3. Cost conscious predictions
4. We must take into account the random nature of our data

Modern-day data science combines mathematics, data, risk-analysis, and statistics to meet the above criteria. This statement is justified in the following subsections.

1.2 Linear Models for Regression

The aim of linear modelling is to predict the value of a continuous target variable $y \in \mathbb{R}$, given a d-dimensional vector of input variables $\mathbf{x} \in \mathbb{R}^d$. In practice we observe a dataset of pairs of inputs and outputs $D := \{(\mathbf{x}_i, y_i)\}_{i=1}^n$.

We aim to minimize the sum of squared errors $\sum_{i=1}^n (y_i - f(\mathbf{x}_i; w))^2$, where $f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}_1, \mathbf{x} \rangle + w_0$. Note that in the following $\mathbf{y}$ is a n-dimensional row vector, $\mathbf{w} = [\mathbf{w}_1, w_0]^{\mathrm{T}}$ is a $(d+1)$-dimensional column vector, and the data matrix $\mathbf{X} \in \mathbb{R}^{(d+1)\times n}$. We choose our model parameters $\mathbf{w}_{LS}$ such that

$$\mathbf{w}_{LS} := \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^n (y_i - f(\mathbf{x}_i; w))^2$$

Minimizing the above function gives us $\mathbf{w}_{LS} = (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{y}^{\mathrm{T}}$. The proof can be found in Appendix A.1.

The analytical solution for $\mathbf{w}_{LS}$ relies upon the invertibility of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$. In the case that $n < d$ (we have more parameters than observations), the matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ does not have full rank and is therefore non-invertible. This is shown in Appendix A.1.1.

Clearly least squares linear modelling is data-driven, and considers costs via the squared error. In order to justify that least squares linear models take into account the random nature of our data, we deduce $\mathbf{w}_{LS}$ by creating a model from a probabilistic point of view.

We begin by expressing the randomness of our target variable y with a probability distribution. Given $\mathbf{x}$, we say that $y|\mathbf{x} \sim N(f(\mathbf{x}; \mathbf{w}), \sigma^2)$. This is for a single $(\mathbf{x}, y)$ pair. In practice, our dataset consists of n observations. We further assume that $(\mathbf{x}_i, y_i)$ are independent and identically distributed (IID). With this assumption, we have

$$\mathbb{P}(y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}, \sigma) = \prod_{i=1}^n \mathbb{P}(y_i | \mathbf{x}_i; \mathbf{w}, \sigma)$$

The proof can be found in Appendix A.1.2.

Given this probabilistic model, we want to tune our parameters $\mathbf{w}$ and σ in a data-driven way. To do this we use **Maximum Likelihood Estimation** (MLE). The underlying idea is as follows: A model with parameters that assign a high probability to observing our data is more likely than a model with parameters that assign a relatively low probability to observing our data.

For a model with parameters $\boldsymbol{\theta}$ and observed data $\mathbf{y}$, we define the likelihood to be $L(\boldsymbol{\theta}) = \mathbb{P}(\mathbf{y}; \boldsymbol{\theta})$ We often choose to work with the log-likelihood $l(\boldsymbol{\theta}) = \log(L(\boldsymbol{\theta}))$. The parameter which maximizes the likelihood is $\hat{\boldsymbol{\theta}} := \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, l(\boldsymbol{\theta})$.

Given our probability model of y given $\mathbf{x}$, and a dataset D_0 we can perform MLE to obtain $\mathbf{w}_{\text{MLE}} := \underset{\mathbf{w}}{\text{argmin}} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i; \mathbf{w}))^2$. This derivation can be found in Appendix A.1.3. We see that the parameter choice determined from our probabilistic model, $\mathbf{w}_{\text{MLE}}$, is equal to the least squares parameter choice $\mathbf{w}_{\text{LS}}$. Hence, LS considers the random nature of our data.

We can also compute the variance of the conditional distribution $y|\mathbf{x}$, σ, by MLE. This gives $\sigma^2_{\text{MLE}} = \frac{1}{n}[y - f(\mathbf{x}; \mathbf{w}_{\text{MLE}})]^2$. Thus the probabilistic view allows us to also determine the uncertainty of our predictions, satisfying another criterion of rational decision making.

1.3 LS with Feature Transforms

Linear LS assumes that the relationship between y and $\mathbf{x}$ is linear, and so linear LS only fits straight lines.

We can easily extend the current LS setup to include non-linear functions of our features. Let $\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^b$ be a feature transform. For example, $\boldsymbol{\phi}(x) := [x, x^2, x^3, \ldots, x^b]$ is a polynomial transform. In this case,

$$\mathbf{w}_{\text{LS}} := \underset{\mathbf{w}}{\text{argmin}} \sum_{i \in D_0} (y_i - g(\mathbf{x}_i; \mathbf{w}))^2, \text{ for } g(\mathbf{x}; \mathbf{w}) := \langle \mathbf{w}_1, \boldsymbol{\phi}(\mathbf{x}) \rangle + w_0, \mathbf{w} := [\mathbf{w}_1, w_0]^{\mathrm{T}}$$

The solution is $\mathbf{w}_{\text{LS}} = (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\mathrm{T}})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\mathrm{T}}$. Similarly to before, $\boldsymbol{\phi}(\mathbf{X}) \in \mathbb{R}^{(b+1)\times n}$. When $\boldsymbol{\phi}(\mathbf{X})$ is a symmetric matrix ($\boldsymbol{\phi}(\mathbf{X}) = \boldsymbol{\phi}(\mathbf{X})^{\mathrm{T}}$), $\mathbf{w}_{\text{LS}}$ can be written $\mathbf{w}_{\text{LS}} = \boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\mathrm{T}}$. This is shown in Appendix A.1.4.

It is worth noting that introducing complex feature transforms can increase the likelihood of *overfitting*. This can be dealt with via regularization techniques. We may also encounter the *curse of dimensionality*.

1.4 Overfitting and the Curse of Dimensionality

Suppose $\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^b$. That is, we have b parameters. The variable b is a measure of our model's complexity. Overfitting occurs when we increase our model's complexity (represented by b in the above), and it can no longer **generalize** to new data. This is a key concept in statistical decision making.

When training a model with a training set and test set, we notice:

1. As we **increase** the number of parameters in our model, the training error continues to **decrease**. Our regression $f(\mathbf{x}; \mathbf{w}_{\text{LS}})$ fits the training set better and better as b increases.
2. The testing error **decreases and then increases** again. Our regression $f(\mathbf{x}; \mathbf{w}_{\text{LS}})$ does not generalize to the unseen test data when b is too large. This is known as the **bias-variance tradeoff**.

We require a method to efficiently select the number of parameters to use in our model. We could split our data into training, validation, and test sets, and tune the number of parameters by minimizing the validation error, but this may be an inefficient use of our data. To overcome this, we use **cross-validation**.

1.4.1 Cross-Validation

Cross-validation (CV) is a well-known model selection technique. It is best summarized as a series of steps:

1. Split dataset D into $k+1$ disjoint sets $D_0, \ldots, D_k$.
2. For $i = 0$ to k,
   (a) Fit $f^{(i)}_{\text{LS}}(b)$ on all subsets except D_i for all b,
   (b) Compute the error $\text{Err}(D_i, f^{(i)}_{\text{LS}}(b))$, for all b.
3. Select the b that minimizes $\sum_i \text{Err}^{(i)}/(k+1)$.

In the above, k is a parameter to be chosen. It can be as high as $n-1$, which is known as leave-one-out CV. The computational cost of CV is very high and its effectiveness relies upon the IID assumptions of our dataset. In some cases such as time series, randomly splitting the data may not ensure that the IID assumption holds.

1.4.2 Regularization

We often want to utilize the flexibility of $f(\mathbf{x};\mathbf{w})$ offered by a large value of b. In this case, we use regularization.

$$\mathbf{w}_{LS-R} := \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i\in D}[y_i - f(\mathbf{x}_i;\mathbf{w})]^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

The second term (the regularization term) discourages $\mathbf{w}$ from taking large values. This version uses the L_2-norm which shrinks parameters towards zero. We could use another norm such as the L_1-norm which performs parameter selection by making some parameter values equal to zero.

The value of λ is a choice. As **$\boldsymbol{\lambda}$ increases**, **overfitting decreases** and at large values we may underfit the data, as our model loses its flexibility.

For the regularization term $\lambda\mathbf{w}^{\mathrm{T}}\mathbf{w}$, we have $\mathbf{w}_{\text{LS-R}} := (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\mathrm{T}} + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{X})y^{\mathrm{T}}$. Clearly as λ increases, the values of $\mathbf{w}_{\text{LS-R}}$ become smaller, corresponding to the fitted curve $f(\mathbf{x};\mathbf{w}_{\text{LS-R}})$ becoming flatter and smoother. As above, we may also use cross-validation to choose λ.

We can motivate the use of regularization with three different approaches:

1. Regularized least squares (Frequentist) — Similar to previous sections, we minimize the loss function.
2. Maximum A Posteriori (semi-Bayesian) — We find the parameters $\mathbf{w}$ which maximize the posterior density $p(\mathbf{w}|D)$. This only gives a point estimate for $\mathbf{w}$.
3. Fully probabilistic approach (full Bayesian) — We make predictions by calculating $p(\hat{y}|\mathbf{x}, D)$ as a marginalized probability.

Details of the three different approaches and proof of the equivalence of their parameters can be found in Appendix A.1.5.

1.4.3 The Curse of Dimensionality

Suppose we have one input variable, $x \in \mathbb{R}$. We may use feature transforms to expand our set of features, $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^b$. We now have b parameters in our model.

What if $\mathbf{x} \in \mathbb{R}^d$? We will require $\boldsymbol{\phi}(\mathbf{x}) := [\mathbf{h}(x^{(1)}), \ldots, \mathbf{h}(x^{(d)})]$, $\mathbf{h}(t) := [t^1, t^2, \ldots, t^b] \in \mathbb{R}^b$. $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{db}$. This includes polynomials of each feature up to order b. If we include pairwise cross-dimensional polynomials we have $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^{db+\binom{d}{2}}$. Consider including terms all the way up to d-plets. We know that $\binom{d}{1} + \binom{d}{2} + \binom{d}{3} + \cdots + \binom{d}{d} = 2^d$. That is, **the number of parameters can grow exponentially with dimensionality d**. Thus in order to maintain invertibility of $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ (we require $n > d$) we require that **n grows exponentially with d**. This problem is known as the **curse of dimensionality**. It prevents us from solving high-dimensional problems.

1.5 Binary Classification

Some problems involve making a discrete decision. For example, given a patient's medical data such as body weight, height, and age, we may want to determine whether this person should be a patient or not. We have the following setup:

1. **Input**: $\mathbf{x} \in \mathbb{R}^d$.

2. **Output**: a class label, $y \in \{+1, -1\}$.
3. **Task**: Given $\mathbf{x}$ make a prediction y whilst minimizing errors/cost.

When working with a regression problem, we want to learn a function $f(\mathbf{x})$ that predicts y. However, with a classification problem we want to find a **decision boundary** $f(\mathbf{x}) = 0$, which splits the space of $\mathbf{x}$ into two areas corresponding to the two class labels.

1.5.1 Bayes' Optimal Classifier

When predicting class labels, we want to minimize misclassification. That is, we want to minimize

$$\mathbb{P}(\mathbf{x} \text{ is FP or FN}|f) = \int_{R_+} p(\mathbf{x}, y = \text{“}-1\text{”})d\mathbf{x} + \int_{R_-} p(\mathbf{x}, y = \text{“}+1\text{”})d\mathbf{x}.$$

This is minimized when $f(\mathbf{x}) = p(\mathbf{x}, y = \text{“}+1\text{”}) - p(\mathbf{x}, y = \text{“}-1\text{”})$. A proof is given in Appendix A.2.1. This $f(\mathbf{x})$ is referred to as **Bayes' optimal classifier**. This is only an idealized optimal classifier as in practice we do not have access to $p(\mathbf{x}, y)$, we only have access to data from each distribution.

1.5.2 Risk Minimization

We have not accounted for the fact that making wrong decisions may have different costs. We can encode the cost of making errors for different classes into a **loss matrix**. To make an effective decision we must minimize the **expected loss of making a wrong decision**.

Suppose the true output is y and the decision is denoted y_0, then the optimal decision is given by

$$\underset{y_0}{\text{argmin}}\, \mathbb{E}_{p(y|\mathbf{x})}[L(y, y_0)|\mathbf{x}].$$

The value of L is determined by the loss matrix. Once again, we do not have $p(y|\mathbf{x})$ but we can infer it from $p(y|\mathbf{x}, D)$. Thus, we look for $\text{argmin}_{y_0} \mathbb{E}_{p(y|\mathbf{x},D)}[L(y, y_0)|\mathbf{x}]$.

In *classification tasks* there are two approaches to obtaining $p(y|\mathbf{x}, D)$:

1. A straightforward (**discriminative**) approach — We infer $p(y|\mathbf{x}, D)$ using D.
2. An indirect (**generative**) approach — $p(y|\mathbf{x}, D) \propto p(\mathbf{x}|y, D)p(y)$.
   (a) Infer $p(\mathbf{x}|y, D)$ using D.
   (b) $p(y)$ can be determined by the proportion of pos/neg. samples.

The inference of $p(y|\mathbf{x}, D)$, and $p(\mathbf{x}|y, D)$ can be done via MLE, MAP, etc. The discriminative approach only differentiates predictions of y. The generative approach also allows us to 'generate' new inputs $\mathbf{x}$ given an output y. However, learning $p(\mathbf{x}|y)$ with a high-dimensional $\mathbf{x}$ can be very hard. Due to this, if your task is only classification then the discriminative approach is best.

We can also **reject decision making** when $\max\{p(y = \text{“}+1\text{”}|\mathbf{x}), p(y = \text{“}-1\text{”}|\mathbf{x})\}$ is lower than a specified threshold.

1.5.3 Connection to Regression

The output of a regression is a continuous variable, so we cannot use a loss matrix. As seen in Section 1.2, we also minimize the expected loss as $\hat{y} := \text{argmin}_{y_0} \mathbb{E}_{p(y|\mathbf{x})}[L(y, y_0)|\mathbf{x}] = \mathbb{E}_{p(y|\mathbf{x})}[y]$.

It is unlikely we will have $p(y|\mathbf{x})$, but we can infer $p(y|\mathbf{x}, D)$ via MLE, MAP, etc. Thus $\hat{y} \approx \mathbb{E}_{p(y|\mathbf{x},D)}[y]$. This corresponds to looking for the **mean of the inferred predictive distribution**.

2 Probability Theory

2.1 Multivariate Normal Distribution

Definition (Multivariate Normal Distribution)**.** A random vector $\mathbf{X} \in \mathbb{R}^n$ is said to have a **multivariate normal distribution** with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ if its probability density function is given by

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

This is written as $\mathbf{X} \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

MVN random variables are multi-dimensional generalizations of univariate normal distributions. We can see this using:

- $\boldsymbol{\Sigma}^{-1} = \mathbf{U}\mathbf{D}\mathbf{U}^{\mathrm{T}}$, the eigendecomposition of $\boldsymbol{\Sigma}^{-1}$.
- $\mathbf{y} = \mathbf{U}^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu})$, a coordinate transform of the $\mathbf{x}$ variables.
- Using the above, $p(\mathbf{y}) = \prod_i N_y(0, d_i)$, where $1/d_i$ is the ith diagonal element of $\mathbf{D}$

Hence, under a coordinate transform, a multivariate normal random vector $\mathbf{X} \in \mathbb{R}^n$ is the product of n univariate normal random variables. This is a useful fact for generating MVN r.v's using normal r.v's.

Definition (Mahalanobis Distance)**.** An observation $\mathbf{x} \in \mathbb{R}^n$ from a distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ has **Mahalanobis distance**

$$D(\mathbf{x}) = \sqrt{(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

The Mahalanobis distance measures the distance between $\mathbf{x}$ and $\boldsymbol{\mu}$, rotated by $\mathbf{U}$. This distance is **chi-squared distributed** for any $\mathbf{x}$ which is MVN distributed. That is, for $\mathbf{x} \in \mathbb{R}^d$, the Mahalanobis distance $D_M(\mathbf{x}) \sim \chi^2_d$. Hence, this distance **can be used to define a confidence region** for our $\mathbf{x}$.

Moments of MVNs:

- $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$
- $\mathbb{E}[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\mu} + \boldsymbol{\Sigma}$

2.1.1 Partitioned MVNs

Given a partitioned MVN distribution such as

$$\mathbf{x}_a, \mathbf{x}_b \sim N_{\mathbf{x}_a, \mathbf{x}_b}\left(\begin{bmatrix}\boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{aa} & \boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba} & \boldsymbol{\Sigma}_{bb}\end{bmatrix}\right),$$

we want to be able to write down the conditional distribution $\mathbf{x}_a|\mathbf{x}_b$, and the marginal distribution of $\mathbf{x}_a$.

We can show that:

1. $p(\mathbf{x}_a|\mathbf{x}_b) = N_{\mathbf{x}_a}(\boldsymbol{\mu} - \boldsymbol{\Theta}_{aa}^{-1}\boldsymbol{\Theta}_{ab}\mathbf{x}_b) + \boldsymbol{\Theta}_{aa}^{-1}\boldsymbol{\Theta}_{ab}\boldsymbol{\mu}_b, \boldsymbol{\Theta}_{aa}^{-1})$
2. $p(\mathbf{x}_a) = N_{\mathbf{x}_a}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_{aa})$

The second part above tells us that the marginal of a joint MVN has mean and variance that is the same as the mean and variance of the partitioned MVN.

2.1.2 Gaussian Linear Model

For a Gaussian linear model we have:

Prior:	$p(\mathbf{x})$	$=$	$N_{\mathbf{x}}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$
Likelihood:	$p(\mathbf{y}, \mathbf{x})$	$=$	$N_y(\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1})$
Marginal:	$p(\mathbf{y})$	$=$	$N_{\mathbf{y}}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}})$
Posterior:	$p(\mathbf{x}\|\mathbf{y})$	$=$	$N_{\mathbf{x}}(\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = (\boldsymbol{\Lambda} + \mathbf{A}^{\mathrm{T}}\mathbf{L}\mathbf{A})^{-1}$

2.1.3 Likelihood and MLE for MVN

Given a dataset $D := \{\mathbf{x}_i\}_{i=1}^n$, the likelihood function with a MVN density can be written as:

$$\begin{aligned} L(\boldsymbol{\mu}, \boldsymbol{\Sigma}, D) &= \sum_{i=1} \log N_{\mathbf{X}_i}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ &= \text{const} - \frac{n}{2}\log|\boldsymbol{\Sigma}| - \frac{\text{tr}(\bar{\mathbf{X}}\bar{\mathbf{X}}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1})}{2} \end{aligned}$$

where $\bar{\mathbf{X}} = [(\mathbf{x}_1 - \boldsymbol{\mu}), \ldots, (\mathbf{x}_n - \boldsymbol{\mu})] \in \mathbb{R}^{d \times n}$ is the 'centralized' dataset.

To get the MLEs we compute $\max\limits_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} L(\boldsymbol{\mu}, \boldsymbol{\Sigma}, D) = \max\limits_{\boldsymbol{\Sigma}} \max\limits_{\boldsymbol{\mu}} L(\boldsymbol{\mu}, \boldsymbol{\Sigma}, D)$. We get:

- $\boldsymbol{\mu}_{\text{MLE}} := \frac{1}{n}\sum_{i=1}^n \mathbf{x}_i$, then
- $\boldsymbol{\Sigma}_{\text{MLE}} := \bar{\mathbf{X}}_{\text{MLE}}\bar{\mathbf{X}}_{\text{MLE}}{}^{\mathrm{T}}$

3 Linear Methods for Regression

3.1 Bias-Variance Decomposition

We saw in Section 1.4 that as our model $f(\mathbf{x}; \mathbf{w}_{\text{LS}})$ becomes more complex and flexible, it becomes unable to generalize to an unobserved dataset. In this section we produce a mathematical explanation of overfitting. We will explain overfitting via a frequentist analysis called the **bias-variance tradeoff**.

We can easily compute the training error of $\mathbf{w}_{\text{LS}}$ on a training set D, $E(D, \mathbf{w}_{\text{LS}})$, however we are not interested in the error over a specific dataset D. We are interested in the expected error over all possible datasets. To compute this we take the expectation with respect to the dataset D:

$$\begin{aligned}
\mathbb{E}_D[E(D, \mathbf{w}_{\text{LS}})] &= \mathbb{E}_D\left[\sum_{i \in D}[y_i - f(\mathbf{x}_i; \mathbf{w}_{\text{LS}})]^2\right] \\
&= \sum_{i=1}^{n} \underbrace{\mathbb{E}_D\left[[y_i - f(\mathbf{x}_i; \mathbf{w}_{\text{LS}})]^2 | \mathbf{x}_i\right]}_{\text{Expected loss}} \\
\mathbb{E}_D\left[[y_i - f_{\text{LS}}]^2 | \mathbf{x}_i\right] &= \mathbb{E}_\epsilon\left[[y_i - f_{\text{LS}}]^2 | \mathbf{x}_i\right] \\
&= \underbrace{\text{var}_\epsilon[\epsilon]}_{\text{irreducible error}} + \underbrace{[g(\mathbf{x}_i) - \mathbb{E}_\epsilon[f_{\text{LS}} | \mathbf{x}_i]]^2}_{\text{bias}^2} + \underbrace{\text{var}_\epsilon[f_{\text{LS}} | \mathbf{x}_i]}_{\text{variance}}
\end{aligned}$$

We notice the following:

- The first term measures the randomness of the data generating process.
- The second term measures how close our prediction is to the true data. As our model's complexity increases the bias shrinks towards zero.
- The third term measures the variance of our estimator. As our model's complexity increases and it becomes more 'wiggly', the variance increases.

The **bias-variance tradeoff**:

- As we increase b, f_{LS} becomes **more complex** and can fit a more complex underlying function, so the **bias decreases**.
- As we increase b, the flexibility afforded to f_{LS} allows it to fit to the noise in our dataset, so the **variance increases**.

We must balance the bias and variance in order to minimize the expected loss. This provides us with the **minimum expected error**.

Definition (In-sample Error)**.** The collective error over the entire dataset.

$$\text{In-sample error} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_\epsilon[(y - f_{\text{LS}})^2 | \mathbf{x}_i]$$

The in-sample error considers the average error given if we were able to repeatedly resample y for every bbx_i in our dataset. In practice we do not know the latent generating function $g(\mathbf{x})$, nor do we know the true distribution of ϵ. This makes it impossible to get new samples of y over our existing $\mathbf{x}_i$. Hence, we use the out-sample error.

Definition (Out-sample Error)**.**

$$\text{Out-sample error} = \mathbb{E}_{\mathbf{x}}\mathbb{E}_{\epsilon}[(y - f_{\text{LS}})^2|\mathbf{x}] = \mathbb{E}_{p(y,\mathbf{x})}[(y - f_{\text{LS}})^2]$$

This is the error over the entire distribution of $\mathbf{x}$.

If we have a set of unseen datapoints $D_1 := \{(y_i', x_i')\}_{i=1}^{n'}$,if D and D_1 are IID then by the law of large numbers we have

$$\frac{1}{n'} \sum_{(y',\mathbf{x}')\in D_1} (y' - f_{\text{LS}}(\mathbf{x}'))^2 \to \mathbb{E}_{p(y,\mathbf{x})}[(y - f_{\text{LS}})^2].$$

This justifies the use of $E(D_1, \mathbf{w}_{\text{LS}})$ (the **testing error**) to evaluate the predictions given by f_{LS}.

In approximating the out-sample error via the testing error, we can assess the degree of overfitting of our model. This requires us to reserve some data for testing, which may be considered a 'waste' of data. CV may help us avoid this loss of information but it has a large cost. We may also not be able to compute the out-sample error as our dataset is not IID (e.g. time-series data).

3.2 Feature Transforms and Kernel Methods

3.2.1 Linear Basis Expansions

In Section 1.3 we looked at feature transformations such as the polynomial transform $\boldsymbol{\phi}(x) := [x, x^2, \ldots, x^b]$. For a periodic response y we may consider feature transforms such as the trigonometric transform

$$\boldsymbol{\phi}(x) := [1, \sin(x), \cos(x), \sin(2x), \cos(2x), \ldots, \sin(bx), \cos(bx)] \in \mathbb{R}^{2b+1}.$$

Idea: It is very unlikely that our latent data generating mechanism $g(\mathbf{x})$ is linear in $\mathbf{x}$. We need to expand the space spanned by our feature variables to provide the estimator $f(\mathbf{x}; \mathbf{w})$ with more flexibility.

$$g(\mathbf{x}) \approx f(\mathbf{x}; \mathbf{w}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle = \sum_i w^{(i)} \phi^{(i)}(\mathbf{x})$$

This is called a **linear basis expansion** of $\mathbf{x}$ — we are expanding the space spanned by our input variables. The $\phi^{(i)}$ are called **basis functions**.

Some choices of basis functions include:

- The polynomial basis: $\boldsymbol{\phi}^{(i)}(x) := [1, x, x^2, \ldots, x^b]$,
- Non-linear transformations of single inputs: $\phi^{(i)}(\mathbf{x}) := \log(x_j)$ or $\phi^{(i)}(\mathbf{x}) := \sqrt{x_j}$ for some $x_j \in \mathbf{x}$,
- The **radial basis function** (RBF): $\phi^{(i)}(\mathbf{x}) := \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}_i\|^2}{\sigma^2}\right)$, where
  - $\sigma > 0$ is called the bandwidth and is determined *before* fitting. A common heuristic is to set σ to be the median of all pairwise distances of $\mathbf{x}$ in the dataset.
  - $\mathbf{x}_i$ are called the RBF centroids and are randomly chosen from $\mathbf{x}$ in the dataset
  - If $g(\mathbf{x})$ has a wide support, $f(\mathbf{x}; \mathbf{w})$ must be supported almost everywhere. The number of RBFs we need to cover a space grows **exponentially** with the space's dimension. Hence, we require $b = O(c^d)$. Here we encounter the curse of dimensionality.

3.2.2 Kernel Methods

Motivation: In the above, $\boldsymbol{\phi}(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}^{s(b)}$ transforms an input $\mathbf{x}$ to a feature space. $f(\mathbf{x}; \mathbf{w})$ is an inner product in this feature space. By increasing b we can increase the dimension of the feature space, and thus increase the flexibility of our estimator f. By using kernel methods we can expand the feature space to be infinitely dimensional, greatly improving the flexibility of f.

When we use kernel methods, we must define an inner product / kernel function $k(\cdot,\cdot)$. We can then write the prediction function as

$$f(\mathbf{x};\mathbf{w}_{LS-R}) := \mathbf{k}(\mathbf{K}+\lambda I)^{-1}\mathbf{y}^{\mathrm{T}},$$

where $k^{(i)} = k(\mathbf{x},\mathbf{x}_i) = \langle \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}_i)\rangle$, $\mathbf{k} \in \mathbb{R}^n$. $\mathbf{K}_{i,j} = k(\mathbf{x}_i,\mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j)\rangle$.

Note: $\boldsymbol{\phi}(\mathbf{x})$ only appears in the inner product $k(\cdot,\cdot)$. Hence, even if we cannot write down $\boldsymbol{\phi}(\mathbf{x})$ explicitly we can still compute $f(\mathbf{x};\mathbf{w}_{LS-R})$. There are many choices of k that correspond to inner products of powerful and potentially infinite dimensional feature transforms $\boldsymbol{\phi}(\mathbf{x})$.

If an explicit $\boldsymbol{\phi}(\mathbf{x})$ can be derived from k we say that k induces $\boldsymbol{\phi}(\mathbf{x})$.

Choices of k:

- Linear kernel function: $k(\mathbf{x}_i,\mathbf{x}_j) := \langle \mathbf{x}_i,\mathbf{x}_j\rangle$. Implicit feature transform: $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$.
- Polynomial kernel function (degree b): $k(\mathbf{x}_i,\mathbf{x}_j) := (\langle \mathbf{x}_i,\mathbf{x}_j\rangle + 1)^b$. Implicit feature transform: $\boldsymbol{\phi}(\mathbf{x}) = [1,\mathbf{x},\ldots,\mathbf{x}^b]$.
- The **radial basis kernel** (RBF kernel): $k(\mathbf{x}_i,\mathbf{x}_j) := \exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_i\|^2}{\sigma^2}\right)$. The feature transform $\boldsymbol{\phi}(\mathbf{x})$ induced by k is **infinitely dimensional**.

Our choice of k may depend on the domain/task, and the type of data we have access to. The RBF kernel is a good suggestion for $\mathbf{x} \in \mathbb{R}^d$.

Computational cost: The cost of computing $\mathbf{K}$ is $O(n^2)$, and the cost of computing $(\mathbf{K}+\lambda I)^{-1}$ is $O(n^3)$. Note that these **costs do not depend on** b. These costs are of course demanding for a large n.

3.3 Probabilistic Model Selection

Motivation: When choosing a model we want to minimize the expected squared loss. This is not a simple procedure, as we cannot generate new datasets easily. We could instead minimize the out-sample (testing) error, but this may not be optimal (non IID data, waste of data, etc.). To avoid this problem, we consider model selection from a probabilistic point of view.

Suppose we have a set of models $\mathcal{M} = \{m_1,\ldots,m_K\}$. We define the model prior $p(m)$ and by Bayes' rule we have $p(m|D) \propto p(D|m)p(m)$. This expresses the **preference of models** given D. In order to make a prediction we marginalize over all models in $\mathcal{M}$:

$$p(\hat{y}|D) = \sum_{m\in\mathcal{M}} p(\hat{y}|m,D)p(m|D)$$

This is a weighted sum of the predictions produced by different models. This contrasts greatly to the frequentist approach, for which

$$\hat{m} := \operatorname*{argmin}_m \sum_{i=1}^{n} \mathbb{E}_D\{[y - f(\mathbf{x}_i;\mathbf{w},m)]^2|\mathbf{x}_i\}$$

A key takeaway is that: **Frequentists minimize, and Bayesians marginalize**.

In probabilistic model selection, we select the most probable model given by $p(m|D)$ in order to approximate the $p(\hat{y}|D)$ given above. We need to compute $p(m|D)$. By Bayes' rule we have

$$p(m|D) \propto \underbrace{p(D|m)}_{\text{model evidence}} \underbrace{p(m)}_{\text{prior}}$$

We can rewrite the model evidence as $p(D|m) = \int p(D|\mathbf{w},m)p(\mathbf{w}|m)d\mathbf{w}$, where the vector $\mathbf{w}$ parametrizes the model m. We select the model m which **maximizes this quantity**.

3.3.1 Model evidence

In the following we introduce assumptions which allow us analyze the properties of the model evidence, and the properties of models which maximize this quantity.

We make the following assumptions:

- $p(\mathbf{w}|D, m)$ plateaus at $\mathbf{w}_{\text{MAP}}$. (Note that $p(\mathbf{w}|D, m) \propto p(D|\mathbf{w}, m)p(\mathbf{w}|m)$),
- The prior is flat $p(w|m) = \frac{1}{\Delta\text{prior}}$.

Then we can write $p(D|m) = \int p(D|w, m)p(w|m)dw \approx p(D|w_{\text{MAP}}, m) \cdot \frac{\Delta\text{posterior}}{\Delta\text{prior}}$. Clearly, for a model with b parameters $\log p(D|m) \approx \log p(D|\mathbf{w}_{\text{MAP}}, m) + b \log \frac{\Delta\text{posterior}}{\Delta\text{prior}}$.

We assume that the posterior is sharper than the prior (this is the case most of the time), $\frac{\Delta\text{posterior}}{\Delta\text{prior}} < 1$, and thus the second term in $\log p(D|m)$ is negative. Recall that we want to maximize $\log p(D|m)$.

Key take away: In maximizing $\log p(D|m)$, we see that:

- Models with a **wider posterior** are preferred. Models which fit the dataset too well give us less confidence.
- We prefer models with a **lower value of** b. Models with high complexity are penalized.

The model evidence prefers a model that maximizes $p(D|\mathbf{w}_{\text{MAP}}, m)$, whilst balancing the model complexity b, and the variance of the posterior.

3.3.2 Tuning hyperparameters

When implementing a model there are often hyperparameters which we tune in order to minimize the expected squared error. With the probabilistic approach we use **marginalized likelihood maximization**.

Once again, we want to calculate the predictive distribution

$$\begin{aligned} p(\hat{y}|D) &= \int p(\hat{y}|D, \alpha)p(\alpha|D)d\alpha \\ &= \int\int p(\hat{y}|D, \mathbf{w}, \alpha)p(\mathbf{w}|D, \alpha)p(\alpha|D)d\mathbf{w}d\alpha \end{aligned}$$

This integral with respect to α may be intractable and so we make an assumption that will simplify the problem. We assume that the distribution $p(\alpha|D)$ is 'pointy' at $\hat{\alpha}$. Then $p(\hat{y}|D) \approx \int p(\hat{y}|D, \mathbf{w}, \hat{\alpha})p(\mathbf{w}|D, \hat{\alpha})d\mathbf{w}$.

We need to find the $\hat{\alpha}$ at which $p(\alpha|D)$ is maximized. We see that

$$p(\alpha|D) \propto p(D|\alpha)p(\alpha) = \int p(D|\mathbf{w}, \alpha)p(\mathbf{w}|\alpha)p(\alpha)d\mathbf{w}.$$

In order to maximize this, we assume that the prior $p(\alpha)$ is flat, and simply choose

$$\hat{\alpha} := \underset{\alpha}{\text{argmax}} \int p(D|\mathbf{w}, \alpha)p(\mathbf{w}|\alpha)d\mathbf{w}$$

We see that we tune our hyperparameters by finding the values which **maximize the model evidence**.

4 Linear Methods for Classification

4.1 Multi-Class Classification

Recall from Section 1.5 the problem proposed in binary classification: We have an input $\{\mathbf{x}_i\}_{i=1}^n$ and output $y \in \{-1, +1\}$. To solve this classification problem we find a decision boundary $f(\mathbf{x}) = 0$. If $f(\mathbf{x}) \geq 0$ we predict $+1$, if $f(\mathbf{x}) \leq 0$ we predict -1. This has a very clear geometric interpretation. The space spanned by our data is split into two regions R_+ and R_-.

The **multi-class classification problem** occurs when our target variable $y \in \{1, \ldots, K\}$. In this case we cannot use a single decision boundary $f(\mathbf{x})$ to predict y.

- We estimate K functions $\{f_k(\mathbf{x}; \mathbf{w}_k)\}_{k=1}^K$.
- given an input $\mathbf{x}$ we predict $\hat{k}$ if $f_{\hat{k}}(\mathbf{x}; \mathbf{w}_k) > f_j(\mathbf{x}; \mathbf{w}_j)$, $\forall j$.

Our functions f_k no longer have a geometric interpretation, but they do have a probabilistic interpretation.

4.1.1 Least Square Classifier

For **binary classification**:

- $\mathbf{w}_{\text{LS}} := \text{argmin}_{\mathbf{w}} \sum_{i \in D} [y_i - f(\mathbf{x}_i; \mathbf{w})]^2$, where $y_i \in \{-1, +1\}$.
- We predict $\hat{y} := sign(f(\mathbf{x}; \mathbf{w}_{\text{LS}}))$.

Note: We can also use feature transforms $\boldsymbol{\phi}$ for f (polynomial, trigonometric, RBF, etc.). In this case, $f(\mathbf{x}; \mathbf{w}) := \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}) \rangle$. This means that even if our data is not separable in the original space (i.e. if classes' respective convex hulls are non-disjoint), we can expand the feature space such that it becomes separable. **Problem**: The square loss imposed by a LS classifier does not make sense in classification tasks. Outliers which are far from the decision boundary have too much influence and can shift the boundary. LS classification also lacks a probabilistic interpretation.

4.1.2 Fisher Discriminant Analysis

We can think of the inner product $\langle \mathbf{w}, \mathbf{x} \rangle$ as embedding $\mathbf{x}$ onto a one-directional line along the direction $\mathbf{w}$. Fisher Discriminant Analysis seeks to find a $\mathbf{w}$ such that observations within the same class are close, and observations in different classes are maximally separated. This is best understood via an illustration. Our embedding is $\mathbf{w}^{\text{T}} \mathbf{x}$. The embedded center for class k is $\hat{\mu} := \frac{1}{n_k} \sum_{i, y_i = k} \mathbf{w}^{\text{T}} \mathbf{x}_i$. Then the within-class scatterness of class k is $s_{w,k} = \sum_{i, y_i = k} (\mathbf{w}^{\text{T}} \mathbf{x}_i - \hat{\mu}_k)^2$. The embedded dataset center is $\hat{\mu} := \frac{1}{n} \sum_{i=1}^n \mathbf{w}^{\text{T}} \mathbf{x}_i$. The between-class scatterness can then be defined as $s_{b,k} := n_k (\hat{\mu}_k - \mu)^2$. We maximize between-class scatterness, and minimize within-class scatterness, i.e. $\max_{\mathbf{w}} \sum_k s_{b,k} / \sum_k s_{w,k}$. Note that FDA does not learn a decision boundary.

4.2 Probabilistic Classifiers

We want to formulate the classification problem under a probabilistic framework. This will allow us to develop new classifiers justified from a probabilistic approach.

We aim to minimize the expected loss: $\hat{y} := \text{argmin}_{y_0} \mathbb{E}_{p(y|\mathbf{x})}[L(y, y_0) \mid \mathbf{x}]$. In order to compute this we need $p(y \mid \mathbf{x})$. Similar to the regression problem in Section 1.2, we have two options:

- **Discriminative**: Infer $p(y \mid \mathbf{x})$ directly.
- **Generative**: Infer $p(y \mid \mathbf{x}) \propto p(\mathbf{x} \mid y) p(y)$. (Infer $p(\mathbf{x} \mid y)$).

4.2.1 Generative Classifiers

To infer $p(\mathbf{x} \mid y)$, we need to specify a model for our $\mathbf{x}$. In doing this we will place quite stringent assumptions on our **class density** $p(\mathbf{x} \mid y)$.

Multivariate Normal Distribution — for continuous data.

If $\mathbf{x}$ is continuous, MVN is a natural choice for the conditional distribution of $\mathbf{x}|y$. We model $\mathbf{x}|y = k; \mathbf{w} \sim N_{\mathbf{x}}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$. We shall assume that our data $\mathbf{x}_i, y_i$ are IID for all i, and that our classes share covariance matrix $\boldsymbol{\Sigma}$. Then we have likelihood

$$p(D \mid \mathbf{w}) = \prod_{i \in D} p(\mathbf{x}_i, y_i \mid \mathbf{w}) = \prod_{i \in D} p(\mathbf{x}_i \mid y_i; \mathbf{w}) p(y_i) = \prod_{i \in D} N_{\mathbf{x}}(\boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}) p(y_i)$$

We can then compute the maximum likelihood estimates: $\hat{\boldsymbol{\mu}}_{1,\ldots,K}, \hat{\boldsymbol{\Sigma}} := \operatorname{argmax}_{\boldsymbol{\mu}_{1,\ldots,K}, \boldsymbol{\Sigma}} \sum_{i \in D} \log N_{\mathbf{x}}(\boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}) p(y_i)$. To compute this we

1. Plug in estimates for $p(y_i)$. We use $\frac{n_k}{n}$ for class k.
2. Compute the MLE for $\boldsymbol{\mu}$. $\hat{\boldsymbol{\mu}} := \frac{1}{n_k} \sum_{i \in D, y_i = k} \mathbf{x}_i$.
3. Plug in $\hat{\boldsymbol{\mu}}$ to compute $\hat{\boldsymbol{\Sigma}}$. $\hat{\boldsymbol{\Sigma}} := \sum_{k=1,\ldots,K} \frac{n_k}{n} \underbrace{\frac{1}{n_k} \sum_{i \in D, y_i = k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^{\mathrm{T}}}_{\text{MLE of the covariance of individual classes}}$.

When using a shared covariance matrix in the MVN model, the decision boundary is *piecewise-linear*. In order to make a **prediction**, we predict $\hat{y} := \operatorname{argmax}_y p(y \mid \mathbf{x}; \hat{\mathbf{w}}) \propto p(\mathbf{x} \mid y; \hat{\mathbf{w}}) p(y)$. The **decision boundary** is then $\{\mathbf{x} \mid p(y = k \mid \mathbf{x}; \hat{\mathbf{w}}) = p(y = k' \mid \mathbf{x}; \hat{\mathbf{w}}), \forall k \neq k'\}$.

If we assume that each class k has its own covariance matrix $\boldsymbol{\Sigma}_k$. Then the decision boundary is **no longer linear**.

Naïve Bayes — for discrete data.

Suppose we have a discrete input variable $\mathbf{x} := [x^{(1)}, \ldots, x^{(d)}]$. Once again we are interested in specifying $p(\mathbf{x} \mid y)$. To do this we make the **naïve assumption** of conditional independence. Then we can write $p(\mathbf{x} \mid y = k) = \prod_{i \in D} p(x_i, \mid y = k)$, where $y_i \in \{1, \ldots, K\}$. Hopefully we can then easily estimate $p(x_y \mid y)$ and $p(y)$. Given an observation $\mathbf{x}_0$, we can then make a prediction $\hat{y} := \operatorname{argmax}_y p(\mathbf{x} = \mathbf{x}_0 \mid y) p(y)$. Note that we could also make this naïve assumption and formulate the prediction under a MLE framework.

4.2.2 Discriminative Classifiers

In this case our task is to directly infer $p(y \mid \mathbf{x})$. We will avoid placing assumptions on $p(\mathbf{x} \mid y)$.

Logistic Regression
We see that when $y \in \{-1, +1\}$, we can use Bayes rule and the assumption that $p(\mathbf{x} \mid y) p(y) > 0 \ \forall \mathbf{x}, y$ to write

$$p(y \mid \mathbf{x}) = \frac{1}{1 + \frac{p(\mathbf{x}|y=-1)p(y=-1)}{p(\mathbf{x}|y=1)p(y=1)}}$$

Hence our problem is reduced to learning models for the **density ratio** $\frac{p(\mathbf{x}|y=-1)}{p(\mathbf{x}|y=1)}$.

Note that we are placing less restrictions on our model, as assumptions on $p(\mathbf{x} \mid y)$ imply assumptions on $\frac{p(\mathbf{x}|y=-1)}{p(\mathbf{x}|y=1)}$. However, the converse does not hold.

We model the log ratio $\log \frac{p(\mathbf{x}|y=-1)p(y=-1)}{p(\mathbf{x}|y=1)p(y=1)}$ as $f(\mathbf{x}; \mathbf{w})$. Given this model, we can write

$$p(y = 1 \mid \mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(f(\mathbf{x}; \mathbf{w}))}.$$

We will use the typical $f(\mathbf{x};\mathbf{w}) = \langle \mathbf{x}, \mathbf{w}\rangle + w_0$. The model for $p(y \mid \mathbf{x};\mathbf{w}) := \sigma(f(\mathbf{x};\mathbf{w})$ is a linear function of our covariates wrapped in a (potentially) non-linear transform. This concept of modelling an outcome via plugging a linear function of covariates into a non-linear activation/link function is known as **generalized linear regression**.

It is easy to see that $p(y = -1 \mid \mathbf{x};\mathbf{w}) = \frac{1}{1+\exp(-f(\mathbf{x};\mathbf{w}))}$. Thus we can write $p(y \mid \mathbf{x};\mathbf{w}) = \sigma(f(\mathbf{x};\mathbf{w}) \cdot y)$.

We have likelihood $p(D \mid \mathbf{x}) = \prod_{i\in D} p(y_i \mid \mathbf{x}_i;\mathbf{w})$. This is the same as the regression task. Then we can compute the maximum likelihood estimate! We have $\mathbf{w}_{\text{MLE}} = \text{argmax}_{\mathbf{w}} \log \prod_{i\in D} p(y_i \mid \mathbf{x}_i;\mathbf{w}) = \text{argmax}_{\mathbf{w}} \sum_{i\in D} \log \sigma(f(\mathbf{x};\mathbf{w})y)$. **Note**: Logistic regression is far less influenced by outliers, relative to the least squares classifier.

Adaptations:

- We can use feature transforms on $\mathbf{x}$ to achieve non-linear classifiers.
- We can place priors on our parameters $\mathbf{w}$. Then $\mathbf{w}_{\text{MAP}} = \text{argmax}_{\mathbf{w}} \sum_{i\in D} \log \sigma(f(\mathbf{x};\mathbf{w})y) + \log p(\mathbf{w})$.
- We can use the full probabilistic approach.

Multi-class logistic regression:

We can easily extend binary logistic regression to multi-class logistic regression. We have $p(y = k \mid \mathbf{x}) = \frac{p(\mathbf{x}|y=k)p(y=k)}{\sum_{k'} p(\mathbf{x}|y=k')p(y=k')}$. Then we model the log ratio $\log \frac{p(\mathbf{x}|y=k)p(y=k)}{p(\mathbf{x}|y=k')p(y=k')}$ as $f(\mathbf{x};\mathbf{w}_k) - f(\mathbf{x};\mathbf{w}_{k'})$. We simply estimate $\mathbf{w}$ via MLE, use a softmax activation function, and predict $\hat{y} = \text{argmax}_y\, p(y \mid \mathbf{x};\mathbf{w}_{\text{MLE}})$

4.3 Support Vector Machines

Support Vector Machines (SVM) allow us to find optimal separating hyperplanes when our data is nonseperable. SVMs provide more reliable decision boundaries than methods such as the perceptron classifier.

Problem: When using methods such as the perceptron classifier, we may find that our classifier performs well on the training data but does not generalize to unseen data. This occurs when our decision boundary is close to our training data and thus if we were to gather more data, some data points may cross the decision boundary due to the data's variability.

Solution: If our decision boundary is defined by $f(\mathbf{x};\mathbf{w}) = 0$, we aim to satisfy the following conditions:

1. $\forall i, y_i = +,\ f(\mathbf{x}_i;\mathbf{w}) \geq 0$.
2. $\forall i, y_i = -,\ f(\mathbf{x}_i;\mathbf{w}) \leq 0$.
3. The error margin of $f(\mathbf{x};\mathbf{w})$ is as thick as possible.

For $f(\mathbf{x};\mathbf{w}) = \langle \mathbf{w}', \mathbf{x}\rangle + w_0$, the thickness of the margin is $\frac{1}{\|\mathbf{w}'\|}$. It is important to note that in many cases, our data is not linearly separable and so we must allow our classifier f to make errors. For a point $\mathbf{x}_i$ within the margin we define the distance between the margin and the point to be ϵ_i. We call this setup the **soft-margin classifier**. Our optimization problem becomes:

$$\text{Minimize } \|\mathbf{w}'\|^2, \text{ Subject to } \forall i, y_i f(\mathbf{x}_i;\mathbf{w}) + \epsilon_i \geq 1, \epsilon_i \geq 0$$

This is a constrained minimization problem. If our classifier $f(\mathbf{x};\mathbf{w})$ is a linear model, then the soft-margin classifier is a **convex** minimization problem. Hence, **every local minimum is a global minimum**. To solve this constrained minimization problem, we use the **Lagrangian Dual** and the KKT conditions.

4.3.1 The Lagrange Dual and the KKT Conditions

Consider a constrained optimization problem of the form:

$$\begin{aligned} \text{minimize} \quad f_0(\mathbf{x}) \quad \text{subject to} \quad & f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p, \end{aligned}$$

with variable $\mathbf{x} \in \mathbb{R}^n$. Lagrangian duality allows us to take these constraints into account by augmenting the objective function via a weighted sum of the constraint functions. This allows us to transform the constrained problem to an unconstrained problem. We define the Lagrangian function to be

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} \nu_i h_i(\mathbf{x}),$$

where $\boldsymbol{\lambda}$, $\boldsymbol{\nu}$ are the Lagrange multiplier vectors which are also referred to as the dual variables. The Lagrange dual function is the minimum value of the Lagrangian over $\boldsymbol{x}$: $l(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\mathbf{x}} L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$. Note that this is a concave function, even if the original problem is not convex. For each pair ($\boldsymbol{\lambda}$, $\boldsymbol{\nu}$) the dual function provides lower bounds on the optimal solution to the original problem. We may wonder what the best lower bound on the optimal solution is. This produces the *Lagrange dual optimization problem*:

$$\text{maximize} \quad g(\boldsymbol{\lambda}, \boldsymbol{\nu}) \quad \text{subject to} \quad \lambda_i \geq 0, \quad i = 1, \ldots, m.$$

The original problem is often referred to as the primal problem. The dual problem is a convex optimization problem as the objective function is concave and the constraints are convex. This holds whether the primal problem is convex or not.

KKT optimal conditions:
Assume that the functions $f_0, \ldots, f_m, h_1, \ldots, h_p$ are differentiable. Let $\mathbf{x}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ be primal and dual optimal points with zero duality gap. Since $\mathbf{x}^*$ minimizes $L(\boldsymbol{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$, we have

$$\nabla f_0(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\mathbf{x}^*) = \mathbf{0},$$

$$\begin{aligned} f_i(\mathbf{x}^*) &\leq 0, \quad i = 1, \ldots, m \\ h_i(\mathbf{x}^*) &= 0, \quad i = 1, \ldots, p \\ \lambda_i^* &\geq 0 \quad i = 1, \ldots, m \\ \lambda_i^* f_i(\mathbf{x}^*) &= 0, \quad i = 1, \ldots, m \end{aligned}$$

$$\nabla f_0(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(\mathbf{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\mathbf{x}^*) = \mathbf{0}$$

these are the Karush-Kuhn-Tucker (KKT) conditions. These are necessary conditions, but if our optimization is a convex one then they are sufficient conditions.

Forming the dual optimization problem for soft-margin SVM: The Lagrange dual is $l(\boldsymbol{\lambda}) := \min_{\mathbf{w}', \boldsymbol{\epsilon}} \|\mathbf{w}'\|^2 + \sum_i \epsilon_i - \lambda_i[y_i(\langle \mathbf{w}', \mathbf{x} \rangle + w_0) + \epsilon_i - 1] - \lambda_i' \epsilon_i$. Differentiating w.r.t $\mathbf{w}'$ and $\boldsymbol{\epsilon}$ gives optimality conditions $\mathbf{w}' = \sum_i \lambda_i y_i \boldsymbol{x}_i / 2$, $\lambda + \lambda' = 1$, and $\sum_i \lambda_i y_i = 0$. We can plug this $\mathbf{w}'$ into $l(\boldsymbol{\lambda})$. This gives $l(\boldsymbol{\lambda}) = -\widetilde{\boldsymbol{\lambda}}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \widetilde{\boldsymbol{\lambda}} / 4 + \langle \boldsymbol{\lambda}, \mathbf{1} \rangle$, where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, and $\widetilde{\boldsymbol{\lambda}} := [\lambda_1 \cdot y_1, \ldots, \lambda_n \cdot y_n]^{\mathrm{T}}$. Our Lagrange dual optimization problem is then: maximize $\widetilde{\boldsymbol{\lambda}}^{\mathrm{T}} \mathbf{X}^{\mathrm{T}} \mathbf{X} \widetilde{\boldsymbol{\lambda}} / 4 + \langle \boldsymbol{\lambda}, \mathbf{1} \rangle$ subject to $0 < \lambda_i < 1$, $\sum_i \lambda_i y_i = 0$. We can solve this for $\hat{\boldsymbol{\lambda}}$, which then gives us $\hat{\mathbf{w}}'$. We can plug $\mathbf{w}'$ into the Lagrange dual problem and use the KKT conditions to obtain w_0. Note that $\mathbf{X}$ only appears in $\mathbf{X}^{\mathrm{T}} \mathbf{X}$ and so we can use kernels! Also note that the dual optimization problem is quadratic in n, and is thus slow when n is large.

5 Probabilistic Graphical Models

5.1 Independence of Random Variables

Let X and Y be random variables. If X is independent of Y, then:

- We can write $X \perp Y$,
- Factorization: $p(X, Y) = p(X)p(Y)$,
- $P(X \mid Y) = P(X)$, and $P(Y \mid X) = p(Y)$. (There is no direct information exchange between X and Y).

Let X, Y, and Z be random variables. If X is independent of Y given Z, then:

- We can write $X \perp Y \mid Z$,
- Factorization: $p(X, Y \mid Z) = p(X \mid Z)p(Y \mid Z)$ and $p(X, Y \mid Z) \propto g_1(X, Z)g_2(Y, Z)$,
- $P(X \mid Y, Z) = P(X \mid Z)$, and $P(Y \mid X) = p(Y \mid Z)$. ($Y$ does not provide any information which changes the probability of X given Z, and there is no direct information exchange between X and Y).

Key take away: (Conditional) independence tells us how information is exchanged between random variables.

- $X \perp Y \iff$ No information is exchanged between X and Y.
- $X \perp Y \mid Z \iff$ No information is directly exchanged between X and Y.

5.2 Markov Networks

5.2.1 Representing (Conditional) Independence with Graphs

Given a set of random variables, it will be cumbersome to list all of the (conditional) independencies. A **graphical representation** provides an intuitive and efficient way to describe such dependencies.

Given a graph with vertex set V and edge set E, we define the graph to be $G := (V, E)$. For random variables $X, Y, Z \subseteq V$, we say that X and Y are conditionally independent given Z if X and Y are completely 'blocked' by Z on the graph G. We see that $X \perp Y \mid Z$ is represented by the graph G.

Given the graph in Figure 1 we can immediately read off the (conditional) independence encoded by the graph:

- $D \perp A \mid C$
- $D \perp B \mid C$
- $D \perp A \mid C, B$
- $D \perp A, B \mid C$
- $D \perp B \mid C, A$

Clearly, we can easily **represent dependencies with a graph**.

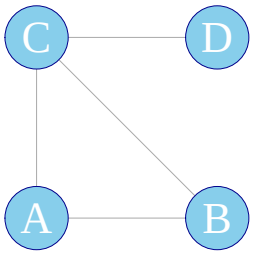

Figure 1: A graph.

5.2.2 Representing Probability Distribution Factorization with Graphs

Given a graph $G := (V, E)$ we say that the probability distribution $p(\mathbf{X})$ factorizes over G if

$$p(\mathbf{X}) \propto \prod_{c \in C} g_c(\mathbf{X}^{(c)}),$$

where C is the set of all cliques (fully connected subgraph) in G, and g_c is a function defined on $\mathbf{X}^{(c)}$ which is the subset of X restricted on C.

Given the graph in Figure 1 we can write down the probability distribution factorization:

$$p(A, B, C, D) \propto g_1(A, B, C) \cdot g_2(C, D)$$

Equivalency of factorization and conditional independence over G:

- If p factorizes over G, then p satisfies all conditional dependence represented by G.
- If p satisfies all conditional independence represented by G, then p factorizes over G.

Definition (Markov Networks)**.** A set of random variables $\mathbf{X}$ which use an undirected graph to represent their conditional independence is called an **undirected graphical model**, or **Markov network**.

5.2.3 Gaussian Markov Networks

Let $\mathbf{x} \in \mathbb{R}^d$ be multivariate Gaussian distributed, $\mathbf{x} \sim N(\mathbf{0}, \mathbf{\Sigma})$. Then

$$\begin{aligned}
p(\mathbf{x}) &\propto \exp\left(-\frac{\mathbf{x}(\mathbf{\Sigma})^{-1}\mathbf{x}^{\mathrm{T}}}{2}\right), \\
&\propto \exp\left(-\frac{\sum_{u,v}\mathbf{x}^{(u)}\mathbf{\Theta}^{(u,v)}\mathbf{x}^{(v)}}{2}\right), \qquad \text{where } \mathbf{\Theta} := (\mathbf{\Sigma})^{-1}, \\
&\propto \prod_{u,v;\mathbf{\Theta}^{(u,v)}\neq 0} \exp\left(-\mathbf{x}^{(u)}\mathbf{\Theta}^{(u,v)}\mathbf{x}^{(v)}\right), \\
&\propto \prod_{u,v;\mathbf{\Theta}^{(u,v)}\neq 0} g_{u,v}(\mathbf{x}^{(u)}, \mathbf{x}^{(v)}).
\end{aligned}$$

We see that $p(\mathbf{x})$ factorizes over G! The graph G is defined by the adjacency matrix $\mathbf{A}$ which takes values

$$\mathbf{A}^{(u,v)} := \begin{cases} 0, & \text{if } \mathbf{\Theta}^{(u,v)} = 0 \\ 1, & \text{if } \mathbf{\Theta}^{(u,v)} \neq 0 \end{cases}$$

Note that $\boldsymbol{A}$ is a symmetric matrix and hence G is an undirected graph. Given a graph G that encodes the conditional independence of a Gaussian random variable, we can infer the sparsity of $\mathbf{\Theta}$ using the above definition of $\mathbf{A}$. This means that **if we know the dependencies of a set of random variables, then we can easily write down a Gaussian model**! We can take this one step further: if we do not know the dependency relationships, we can infer them from data. Given a dataset D we can fit $\widehat{\mathbf{\Theta}}$ using MLE. $\widehat{\mathbf{\Theta}} = \operatorname{argmax}_{\mathbf{\Theta}} \log p(D; \mathbf{\Theta})$. The sparsity of $\widehat{\mathbf{\Theta}}$ produces a graph which then produces a factorization of $p(\mathbf{X})$. This graph and corresponding factorization tells us about the dependency of our variables.

5.2.4 Graphical Lasso

When learning the dependencies between covariates using data, we will obtain *many* non-zero entries in $\mathbf{\Theta}$. To avoid this we introduce a regularization method for graphical models. Given a dataset $D := \{\mathbf{x}_i\}_{i=1}^n$, $\mathbf{x} \in \mathbb{R}^d$. We can construct the Gaussian likelihood $p(D \mid \mathbf{\Theta}) = \prod_i N_{\mathbf{x}_i}(\mathbf{0}, \mathbf{\Theta}^{-1})$. We proceed with MLE with an additional L_1 regularization term.

$$\widehat{\mathbf{\Theta}} := \underset{\mathbf{\Theta}}{\operatorname{argmax}} \log p(D \mid \mathbf{\Theta}) + \lambda\|\mathbf{\Theta}\|_1 = \underset{\mathbf{\Theta}}{\operatorname{argmin}}\, tr(\mathbf{S}\mathbf{\Theta}) - \log\det\mathbf{\Theta} + \lambda\|\mathbf{\Theta}\|_1,$$

where $\mathbf{S}$ is the sample covariance, $\|\mathbf{\Theta}\|_1 = \sum_{i,j}|\mathbf{\Theta}^{(i,j)}|$. We can then construct a graph using the sparsity of $\mathbf{\Theta}$.

5.2.5 Conditional Markov Networks

Often, we are interested in the conditional distribution. In regression and classification we are often tasked with predicting a response variable Y given covariates $\mathbf{X}$. We need to factorize the conditional distribution over the graph G.

We say that a conditional distribution $p(Y \mid \mathbf{X})$ factorizes over a graph G with nodes $V = X \cup Y$, if

- $P(Y \mid \mathbf{X}) = \frac{1}{N(\mathbf{X})} \prod_{c \in C} g_c(V_c)$, where $C := \{c \text{ is a clique in } G \mid V_c \not\subseteq \mathbf{X}\}$,
- $N(\mathbf{X}) := \int \prod_{c \in C} g_c(V_c) dY$.

$N(\mathbf{X})$ is a normalizing constant normalizing the conditional distribution over the domain of the random variable Y. By the definition of the set C, we see that $p(Y \mid \mathbf{X})$ does not include factors defined on subsets of the conditioning variable $\mathbf{X}$.

Example: Consider the graph given in Figure 1.

$$
\begin{aligned}
p(C \mid A, B, D) &= \frac{1}{N(A, B, D)} g_1(A, B, C) \cdot g_2(C, D) \\
N(C) &= \int g_1(A, B, C) \cdot g_2(C, D) dC \\
p(D \mid A, B, C) &= \frac{1}{N(A, B, C)} g_1(C, D) \\
N(A, B, C) &= \int g_1(C, D) dD = g_1(C)
\end{aligned}
$$

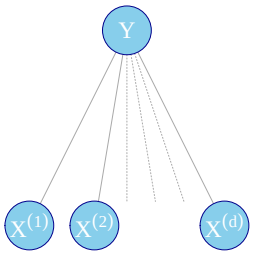

Figure 2: A graph.

We can reconstruct logistic regression using the above formulation of conditional probabilities.

5.2.6 Classification with Conditional Markov Networks

Suppose we have $Y \in \{-1, 1\}, \mathbf{X} \in \mathbb{R}^d$, and a simple undirected Markov network as shown in Figure 2. We are interested in the conditional distribution $p(Y \mid \mathbf{X})$. Using the factorization rule specified above we can write $p(Y \mid \mathbf{X}) = \frac{1}{N(\mathbf{X})} \prod_i g_i(Y, \mathbf{x}^{(i)})$, where $N(\mathbf{X}) = \sum_{Y \in \{-1,1\}} \prod_i g_i(Y, \mathbf{X}^{(i)})$. We can construct a model of the conditional likelihood $p(Y \mid \mathbf{X})$ by defining g_i. Let $g_i(Y = y, \mathbf{X}^{(i)} = x^{(i)}; \beta_i, \beta_0) := \exp(y(\beta_i \cdot x^{(i)} + \beta_0))$. Then

$$
\begin{aligned}
p(y \mid \mathbf{x}; \boldsymbol{\beta}, \beta_0) &= \frac{1}{N(\mathbf{x})} \prod_i \exp(y(\beta^{(i)} \cdot x^{(i)} + \beta_0)) \\
&= \frac{1}{N(\mathbf{x})} \exp(y(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + d\beta_0)) \\
N(X; \boldsymbol{\beta}, \beta_0) &= \sum_{y \in \{-1,1\}} \exp(y(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + d\beta_0)) \\
&= \exp(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + d\beta_0) + \exp(-\langle \boldsymbol{\beta}, \mathbf{x} \rangle - d\beta_0)
\end{aligned}
$$

We can easily show that **this is logistic regression** as it was introduced previously! Simply consider the case where $y = 1$. We see that

$$
\begin{aligned}
p(y=1 \mid \mathbf{x}; \boldsymbol{\beta}, \beta_0) &= \frac{\exp(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + d\beta_0)}{\exp(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + d\beta_0) + \exp(-\langle \boldsymbol{\beta}, \mathbf{x} \rangle - d\beta_0)} \\
&= \frac{1}{1 + \frac{\exp(-\langle \boldsymbol{\beta}, \mathbf{x} \rangle - d\beta_0)}{\exp(\langle \boldsymbol{\beta}, \mathbf{x} \rangle + d\beta_0)}} = \frac{1}{1 + \exp(-2\langle \boldsymbol{\beta}, \mathbf{x} \rangle - 2d\beta_0)} \\
&= \frac{1}{1 + \exp(f(\mathbf{x}; \boldsymbol{\beta}, \beta_0))},
\end{aligned}
$$

where $f(\mathbf{x}; \boldsymbol{\beta}, \beta_0) = -\langle \mathbf{2}\boldsymbol{\beta}, \mathbf{x} \rangle - 2d\beta_0$. If we wanted to make these exactly equivalent we could set $\mathbf{w} = -2\boldsymbol{\beta}$ $w_0 = -2d\beta_0$. Then $f(\mathbf{x}; \boldsymbol{\beta}, \beta_0) = \langle \mathbf{w}, \mathbf{x} \rangle + w_0$ as in Section 4.2.2.

5.3 Bayesian Networks

Markov networks are **undirected** graphical models encoding the conditional independence of our random variables $\mathbf{X}$ and the factorization or the probability distribution $p(\mathbf{X})$. In this section we introduce an approach which uses **directed** graphical models to do the same job. In some cases a directed graphical model may better represent our random variables (X may depend on Y but Y may not depend on X).

We use a Directed Acyclic Graph (DAG) as the graphical representation. We have graph $G := (V, E)$, where V is the vertex set and E is the directed edge set. As the name implies, G is a directed graph which is also acyclic. If there exists a directed edge from A to B, then we say that A is a parent of B, and B is a child of A. We also say that B is a descendant of A. **Example**: In Figure 3 we see that C is a child of D, and D is the parent of C, we can also say that A is a descendant of D.

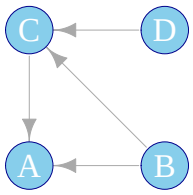

Figure 3: A DAG.

5.3.1 Representing Probability Distribution Factorization with a DAG

A DAG can be used to represent the factorization of a probability distribution. We say that the probability distribution $p(\mathbf{X})$ factorizes over a DAG G if $p(\mathbf{X}) = \prod_{v \in V} p(\mathbf{X}_v \mid \mathbf{X}_{\text{parent}(\mathbf{X}_v)})$. **Example**: For the DAG specified in Figure 3 we see that

$$P(A, B, C, D) = P(A \mid B, C)p(B)p(C \mid B, D)p(D)$$

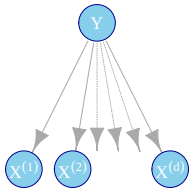

Figure 4: A DAG.

5.3.2 Representing Conditional Independence with a DAG

For a DAG G, we say that a random variable $\mathbf{X}_v$ is independent of $\mathbf{X}_{\text{non-desc}(X_v)}$ given $\mathbf{X}_{\text{parent}(X_v)}$. That is, X_v is conditionally independent of its non-descendants given its parents. **Example**: For the DAG specified in Figure 3 we can easily see that $A \perp D \mid C, D$, and $B \perp D$.

Equivalency: Just as is the case with a Markov network, there is an equivalency between representing conditional independence and the factorization of a probability distribution over a DAG.

Definition (Bayesian Network)**.** A set of random variables $\mathbf{X}$ which use a DAG to represent their conditional independence is called a **directed graphical model**, or **Bayesian network**.

5.3.3 Classification with Bayesian Networks

Suppose $Y \in \{-1, 1\}$ and $\mathbf{X} \in \mathbb{R}^d$. We are interested in the conditional probability $P(Y \mid \mathbf{X}))$, where the probability distribution can be factorized over the DAG in Figure 4. Then we can immediately write

$$p(Y \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid Y)}{p(\mathbf{X})p(Y)} = \frac{\prod_i p(\mathbf{X}^{(i)} \mid Y)p(Y)}{p(\mathbf{X})}$$

This is how Naïve Bayes is derived!

6 Advanced Topics in Machine Learning

6.1 Calibration in Predictive Machine Learning

6.1.1 Calibrated Classification

Definition (Calibrated)**.** Supposed we have $f : X \to S_Y$ with $S_Y = \{[s_1, \ldots, s_K] \mid \sum_{j=1}^{K} s_j = 1, s_j \geq 0\}$. Denoting $S = f(X)$ as the random variable of model predictions, f is said to be *calibrated* if and only if $\forall s \in S_Y, \forall y \in Y$, the following equality holds:

$$P(Y = j \mid S = [s_1, \ldots, s_K]) = s_j$$

Definition (Calibrated Quantile Regression)**.** Suppose we have a pair of jointly distributed random variables $(X, Y) \in \mathbb{R}^2$ and a quantile regression model $g : X \to [0, 1]$. Denoting $G_\tau = g(X, \tau)$ as the random variable of the $\tau-$quantile predictions, g is said to be quantile calibrated iff $\forall \tau \in [0, 1]$, the following holds:

$$P(Y \leq G_\tau) = \tau$$

6.1.2 Post-hoc calibration

When a binary classifier is not calibrated, a calibrator $g : [0, 1] \to [0, 1]$ can be applied to improve the level of calibration.

Some post-hoc approaches include:

- Empirical binning (binary)
- Isotonic regression (binary)
- Beta calibration (binary)
- Dirichlet calibration (multi-class)

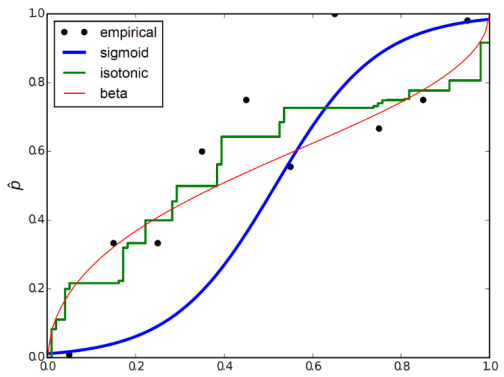

Figure 5: Different post-hoc calibration methods.

Post-hoc quantile calibration: We could use a calibrator $C : [0, 1] \to [0, 1]$ to improve our quantile calibration. However, if a classifier is calibrated and the predicted probability is 0.5 then we can say the target is distributed as Bernoulli(0.5). This does not apply to quantile calibration as proposed above: If a regressor is quantile calibrated then given a predicted mean 0 and standard deviation 1 then we *cannot* say that the target is distributed as Gaussian$(2, 1)$. To overcome this we use **distribution calibration**.

6.1.3 Distribution calibration

Definition (Distribution Calibrated)**.** Suppose we have $f : X \to S_Y$ with $S_Y = \{s \mid s : Y \to [0, \infty), \int_Y s(t)dt = 1\}$. Denoting $S = f(X)$ as the random variable of model predictions, f is said to be distribution-calibrated if and only if $\forall s \in S_Y, \forall y \subseteq Y$, the following equality holds:

$$P(Y = y \mid S = s) = s(y).$$

We should note that being distribution calibrated is a sufficient condition for being quantile calibrated. Models that are well calibrated on a distribution level provide improved uncertainty quantification on the target variable.

Beta calibration: A parametric calibrator can be derived from the beta distribution:

$$\beta_{(a,b,m)}(q) = \frac{1}{1 + e^{-(a \ln q - b \ln(1-q) + m)}}$$

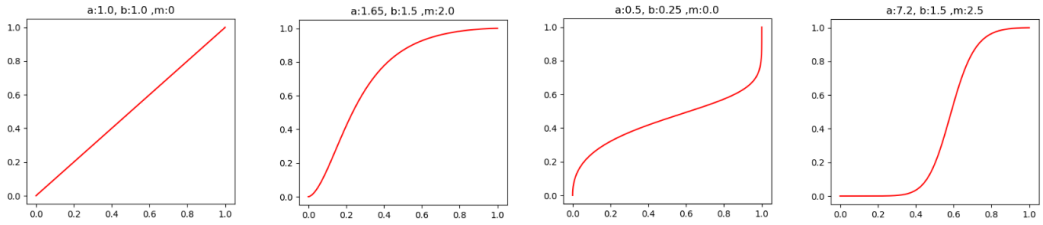

Figure 6: Different forms of the Beta link.

The motivation for the beta calibrator is that it can provide a rich class of calibration maps which is not offered by other approaches. Isotonic calibration is a very powerful non-parametric method but it often overfits when we do not have enough data. Other methods such as calibration via a logistic curve may be reasonably used for normally distributed scores, but they cannot learn the identity function and thus can provide worse probability estimates if applied to an already calibrated classifier.

Distribution calibration is applicable to any conditional density estimator but we focus on a regression setting. Below is the GP-Beta approach which combines multiple output Gaussian processes with beta calibration, for the binary classification and distribution regression.

GP-Beta Model:
We use Beta calibration maps to transform CDFs of the distributions output by the regressor similar to post-hoc calibration methods proposed above. We want ta model which maps any regression output (μ_i, σ_i) to a set of Beta calibration parameters (a_i, b_i, c_i). However for each (μ_i, σ_i) we only observe one target value y_i. To overcome this, we want to observed target values from other observations close to (μ_i, σ_i). This motivated the use of a Gaussian Process: We learn a GP to predict the parameters a, b, c of the Beta calibration map.

We assume there are three latent functions that are jointly distributed with respect to a multi-output GP, corresponding to the parameters a, b, c of the Beta calibration:

$$(w_a, w_b, w_c) \sim GP(0, k, \mathbf{B}),$$

where k is the kernel function on the regression output distributions, and $\mathbf{B}$ is a 3×3 coregionalization matrix modeling the covariance among the outputs. The GP-Beta model uses the univariate Gaussian embedding with a RBF kernel:

$$\mathbf{K}_{i,j} = \frac{\theta}{|\sigma_i + \sigma_j + \theta^2|^{1/2}} \exp\left(-\frac{(\mu_i - \mu_j)^2}{2(\sigma_i + \sigma_j + \theta^2)}\right)$$

Given n training points $(\boldsymbol{\mu}, \boldsymbol{\sigma}) = ((\mu_1, \sigma_1), \ldots, (\mu_n, \sigma_n))$, a Gaussian likelihood on $(w_a^{(i)}, w_b^{(i)}, w_c^{(i)})_{i=1}^n$ can be written as:

$$p\left(\begin{bmatrix}\mathbf{w}_a \\ \mathbf{w}_b \\ \mathbf{w}_c\end{bmatrix} \mid \boldsymbol{\mu}, \boldsymbol{\sigma}\right) = N\left(\begin{bmatrix}\mathbf{w}_a \\ \mathbf{w}_b \\ \mathbf{w}_c\end{bmatrix} \mid \mathbf{0}, \mathbf{B} \otimes \mathbf{K}\right),$$

where $\mathbf{K}$ is the $n \times n$ matrix with elements i, j as described above; it is obtained by applying the kernel function k on $(\boldsymbol{\mu}, \boldsymbol{\sigma})$, $\otimes$ is the Kronecker product, and $\mathbf{w}_a = [w_a^{(1)}, \ldots, w_a^{(n)}]$.

We will write the above as $p(\mathbf{w} \mid \boldsymbol{\mu}.\boldsymbol{\sigma}) = N(\mathbf{w} \mid \mathbf{0}, \mathbf{C})$, where $\mathbf{w} = [\mathbf{w}_1{}^{\mathrm{T}}, \ldots, \mathbf{w}_m{}^{\mathrm{T}}]^{\mathrm{T}}$, $\mathbf{w}_i = [w_a^{(i)}, w_b^{(i)}, w_c^{(i)}]^{\mathrm{T}}$.

For target values $\mathbf{y} = (y_1, \ldots, y_n)$, we can plug in the Beta link:

$$
\begin{aligned}
p(\mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\sigma}) &= \int_{\mathbf{w}} \left(\prod_{i=1}^{n} p(y_i \mid \mathbf{w}_i, \mu_i, \sigma_i) \right) p(\mathbf{w} \mid \boldsymbol{\mu}, \boldsymbol{\sigma}) d\mathbf{w} \\
p(y_i \mid \mathbf{w}_i, \mu_i, \sigma_i) &= \frac{d\beta_{a_i, b_i, c_i}(p(Y \leq t \mid \mu_i, \sigma_i))}{dt} \Bigg|_{t=y_i} \\
a_i &= e^{\gamma_a^{-1} \mathbf{w}_i^{(a)} + \delta_a} \\
b_i &= e^{\gamma_b^{-1} \mathbf{w}_i^{(b)} + \delta_b} \\
c_i &= \gamma_c^{-1} \mathbf{w}_i^{(m)} + \delta_c
\end{aligned}
$$

The exponential function is introduced to ensure the monotonicity of the calibrator, and the hyperparameters γ, δ are introduced to control the model behavior at the prior.

Inference for the GP-Beta model:
The $p(\mathbf{y} \mid \boldsymbol{\mu}, \boldsymbol{\sigma})$ specified above is analytically intractable due to the non-linearity of the link function, and thus optimizing the hyperparameters of the Beta link will be very difficult. Given a test instance (μ_*, σ_*), the density $p(y_i \mid \mathbf{w}_i, \mu_i, \sigma_i)$ is also intractable. We may attempt to overcome these problems by using MCMC, but this is too slow. We cannot use Laplace approximation as it is not compatible with the multi-output GPs. We can use variational inference combined with a Monte-Carlo gradient:

$$
\begin{aligned}
\ln p(\mathbf{y}) &\geq \mathbb{E}_{q(\mathbf{u})}[\ln p(\mathbf{y} \mid \mathbf{u})] - KL[q(\mathbf{u}), N(\mathbf{u})] \\
&\geq \mathbb{E}_{q(\mathbf{w})}[\ln p(\mathbf{y} \mid \mathbf{w})] - KL[q(\mathbf{u}), N(\mathbf{u})]
\end{aligned}
$$

A Basics of Statistical Learning

A.1 Linear regression by minimizing least squares

Proof that the minimizer of $\mathbf{w}_{LS} := \text{argmin}_{\mathbf{w}} \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i; w))^2$ is $\mathbf{w}_{LS} = (fX\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{y}^{\mathrm{T}}$:

$$\begin{aligned}
\mathbf{w}_{LS} &= \underset{\mathbf{w}}{\text{argmin}} \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i; w))^2 \\
&= \underset{\mathbf{w}}{\text{argmin}} \, ||\mathbf{y} - \mathbf{w}^{\mathrm{T}}\mathbf{X}||^2 \\
&= \underset{\mathbf{w}}{\text{argmin}} (\mathbf{y} - \mathbf{w}^{\mathrm{T}}\mathbf{X})(\mathbf{y} - \mathbf{w}^{\mathrm{T}}\mathbf{X})^{\mathrm{T}} \\
&= \underset{\mathbf{w}}{\text{argmin}} \{\mathbf{y}\mathbf{y}^{\mathrm{T}} - 2\mathbf{w}^{\mathrm{T}}\mathbf{X}\mathbf{y}^{\mathrm{T}} + \mathbf{w}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w}\}
\end{aligned} \tag{1}$$

Differentiating with respect to $\mathbf{w}$ and setting equal to zero gives

$$\begin{aligned}
0 &= -2\mathbf{X}\mathbf{y}^{\mathrm{T}} + 2\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{w}_{\mathrm{LS}} \\
\mathbf{w}_{\mathrm{LS}} &= (\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}\mathbf{X}\mathbf{y}^{\mathrm{T}}.
\end{aligned} \tag{2}$$

A.1.1 Least-squares solution may be invertible

Proof that if $d > n$ then the matrix $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is non-invertible.

$$\begin{aligned}
\text{rank}(\mathbf{X}\mathbf{X}^{\mathrm{T}}) &\leq \min(\text{rank}(\mathbf{X}), \text{rank}(\mathbf{X}^{\mathrm{T}})) \\
&= \text{rank}(\mathbf{X}) \\
&\leq \min(d, n) \\
&= n
\end{aligned}$$

A.1.2 Simplification of the joint density

Under assumptions of normality of our errors, and IID data, we have that

$$\mathbb{P}(y_1, \ldots, y_n | \mathbf{x}_1, \ldots, \mathbf{x}_n; \mathbf{w}, \sigma) = \prod_{i=1}^{n} \mathbb{P}(y_i | \mathbf{x}_i; \mathbf{w}, \sigma)$$

To prove this, we first need to show that for A, B, C, D random variables, (A,B) $\perp\!\!\!\perp$ (C, D) $\implies$ A $\perp\!\!\!\perp$ C. This is done as follows:

$$\begin{aligned}
\mathbb{P}(A, C) &= \int \mathbb{P}(A, B, C, D)\, dB dD \\
&= \int \mathbb{P}(A, B)\, \mathbb{P}(C, D)\, dB dD, && \text{by independence of (A,B) and (C,D)} \\
&= \int \mathbb{P}(A, B)\, dB \int \mathbb{P}(C, D)\, dD \\
&= \mathbb{P}(A)\, \mathbb{P}(B), && \text{by properties of joint probability mass functions}
\end{aligned}$$

Using the above, we can see that if $(\mathbf{x}_i, y_i) \perp\!\!\!\perp (\mathbf{x}_j, y_j)$ for $i \neq j$, then

$$\begin{aligned}
\mathbf{x}_i &\perp\!\!\!\perp \mathbf{x}_j, i \neq j \\
y_i &\perp\!\!\!\perp \mathbf{x}_j, i \neq j \\
y_i &\perp\!\!\!\perp y_j, i \neq j
\end{aligned} \tag{3}$$

Using this, we see that

$$\begin{aligned}\mathbb{P}(y_1,\dots,y_n|\mathbf{x}_1,\dots,\mathbf{x}_n;\mathbf{w},\sigma) &= \prod_{i=1}^{n}\mathbb{P}(y_i|\mathbf{x}_1,\dots,\mathbf{x}_n;\mathbf{w},\sigma), && \text{using } y_i \perp\!\!\!\perp y_j, i\neq j\\ &= \prod_{i=1}^{n}\mathbb{P}(y_i|\mathbf{x}_i;\mathbf{w},\sigma), && \text{using } y_i \perp\!\!\!\perp \mathbf{x}_j, i\neq j\end{aligned}$$

A.1.3 MLE of the probabilistic model

$$\begin{aligned}\mathbf{w}_{\text{MLE}} &:= \underset{\mathbf{w}}{\text{argmax}} \log \prod_{i=1}^{n}\mathbb{P}\left(y_i|\mathbf{x}_i;\mathbf{w},\sigma\right)\\ &= \underset{\mathbf{w}}{\text{argmax}}\left[\sum_{i=1}^{n} -\frac{(y_i - f(\mathbf{x}_i;\mathbf{w}))^2}{2\sigma^2}\right] - n\log\sqrt{2\pi\sigma^2}\\ &= \underset{\mathbf{w}}{\text{argmin}}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\mathbf{w}))^2\end{aligned}$$

A.1.4 Least squares solution with symmetric feature transform matrix

When we use feature transforms, the design matrix $\mathbf{X}$ is replaced by $\boldsymbol{\phi}(\mathbf{X})$, and the least squares solution is $\mathbf{w}_{\text{LS}} = (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}}$.

Proof that when $\boldsymbol{\phi}(\mathbf{X})$ is a symmetric matrix, $\mathbf{w}_{\text{LS}} = \boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}}$:

$$\begin{aligned}\mathbf{w}_{\text{LS}} &= (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}}\\ &= \boldsymbol{\phi}(\mathbf{X})^{-\text{T}}\boldsymbol{\phi}(\mathbf{X})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}}, && \text{using } (AB)^{-1} = B^{-1}A^{-1}\\ &= \boldsymbol{\phi}(\mathbf{X})^{-\text{T}}\\ &= \boldsymbol{\phi}(\mathbf{X})^{-1}\mathbf{y}^{\text{T}}, && \text{using } \boldsymbol{\phi}(\mathbf{X}) = \boldsymbol{\phi}(\mathbf{X})^{-\text{T}}\end{aligned}$$

A.1.5 Regularization

In this section we will derive an equation for the parameters which minimize the least squares loss function with regularization. We will also prove its equivalence to parameter solutions derived via a semi-Bayesian approach and Maximum A Posteriori, and a fully probabilistic approach.

Proof that if the regularization term is $\lambda\mathbf{w}^{\text{T}}\mathbf{w}$ then $\mathbf{w}_{\text{LS-R}} := (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}} + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{X})y^{\text{T}}$:

$$\begin{aligned}\mathbf{w}_{\text{LS-R}} &:= \underset{\mathbf{w}}{\text{argmin}}\sum_{i\in D}[y_i - f(\mathbf{x}_i;\mathbf{w})]^2 + \lambda\mathbf{w}^{\text{T}}\mathbf{w}\\ &= \underset{\mathbf{w}}{\text{argmin}}\{(\mathbf{y} - \mathbf{w}^{\text{T}}\boldsymbol{\phi}(\mathbf{X}))(\mathbf{y} - \mathbf{w}^{\text{T}}\boldsymbol{\phi}(\mathbf{X}))^{\text{T}} + \lambda\mathbf{w}^{\text{T}}\mathbf{w}\}\\ &= \underset{\mathbf{w}}{\text{argmin}}\{\mathbf{y}\mathbf{y}^{\text{T}} - 2\mathbf{w}^{\text{T}}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}} + \mathbf{w}^{\text{T}}\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}}\mathbf{w} + \lambda\mathbf{w}^{\text{T}}\mathbf{w}\}\end{aligned}$$

Differentiating with respect to $\mathbf{w}$ and setting equal to zero gives

$$\begin{aligned}-2\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}} + 2\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}}\mathbf{w}_{\text{LS-R}} + 2\lambda\mathbf{w}_{\text{LS-R}} &= 0\\ (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}} + \lambda\mathbf{I})\mathbf{w}_{\text{LS-R}} &= \boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}}\\ \mathbf{w}_{\text{LS-R}} &= (\boldsymbol{\phi}(\mathbf{X})\boldsymbol{\phi}(\mathbf{X})^{\text{T}} + \lambda\mathbf{I})^{-1}\boldsymbol{\phi}(\mathbf{X})\mathbf{y}^{\text{T}}\end{aligned}$$

We now consider a probabilistic view of the regression problem. We have an inverse problem. We assume our data is generated by some latent, unobserved data generating mechanism. In regression we observe y_i and assume it is generated by $y_i = g(x_i) + \epsilon$. Where ϵ is noise. We want to gain some knowledge about g.

We want to infer the posterior probability distribution $p(g|D)$. We do this using Bayes' rule, and the assumption that g can be represented as a parametric function $f(\mathbf{x}; \mathbf{w})$. Once we determine $\mathbf{w}$, we also know g. Thus, we seek to infer $p(\mathbf{w}|D)$.

$$\begin{aligned} p(\mathbf{w}|D) &= \frac{p(D|\mathbf{w})p(\mathbf{w})}{p(D)} \\ &= \frac{\prod_{i \in D} p(y_i|\mathbf{x}_i, \mathbf{w}, \sigma^2)p(w_i)}{p(D)}, \end{aligned}$$

where $y_i|\mathbf{x_i}, \mathbf{w} \sim N(f(\mathbf{x}_i; \mathbf{w}), \sigma^2)$ and $w_i \sim N(0, \sigma^2_{\mathbf{w}})$. Then to make a prediction we can find a $\mathbf{w}$ that maximises this posterior density. This procedure is called Maximum A Posteriori (MAP).

$$\begin{aligned} \mathbf{w}_{\text{MAP}} &:= \underset{\mathbf{w}}{\text{argmax}} \prod_{i \in D} p(y_i|\mathbf{x}_i, \mathbf{w}, \sigma^2)p(w_i) \\ &= \underset{\mathbf{w}}{\text{argmax}} \prod_{i \in D} \exp\left[-\frac{1}{2\sigma^2}(y_i - f(\mathbf{x}_i; \mathbf{w}))^2\right] \exp\left[-\frac{1}{2\sigma^2_{\mathbf{w}}} w_i^2\right] \\ &= \underset{\mathbf{w}}{\text{argmax}} -\frac{1}{2\sigma^2}(\mathbf{y} - f(\mathbf{x}; \mathbf{w}))(\mathbf{y} - f(\mathbf{x}; \mathbf{w}))^{\mathrm{T}} - \frac{1}{2\sigma^2_{\mathbf{w}}} \mathbf{w}^{\mathrm{T}} \mathbf{w} \\ &= \underset{\mathbf{w}}{\text{argmin}} (\mathbf{y} - f(\mathbf{x}; \mathbf{w}))(\mathbf{y} - f(\mathbf{x}; \mathbf{w}))^{\mathrm{T}} + \frac{1}{2} \mathbf{w}^{\mathrm{T}} \mathbf{w} \end{aligned}$$

In the last line we have used $\lambda = 2\sigma^2/\sigma^2_{\mathbf{w}}$.

A.2 Binary Classification

A.2.1 Minimizing false positive and false negatives

Here we prove that the minimizer of $\mathbb{P}(\mathbf{x} \text{ is FP or FN}|f) = \int_{R_+} p(\mathbf{x}, y = \text{“}-1\text{”})d\mathbf{x} + \int_{R_-} p(\mathbf{x}, y = \text{“}+1\text{”})d\mathbf{x}$. is given by $f(\mathbf{x}) = p(\mathbf{x}, y = \text{“}+1\text{”}) - p(\mathbf{x}, y = \text{“}-1\text{”})$.

$$\begin{aligned} f(\mathbf{x}) &= \underset{f(\mathbf{x})}{\text{argmin}} \left[\int_{R_+} p(\mathbf{x}, y = \text{“}-1\text{”})d\mathbf{x} + \int_{R_-} p(\mathbf{x}, y = \text{“}+1\text{”})d\mathbf{x}\right] \\ &= \underset{f(\mathbf{x})}{\text{argmin}} \left[\int p(\mathbf{x}, y = \text{“}-1\text{”}) \mathbb{1}_{\{f(\mathbf{x}) \geq 0\}} d\mathbf{x} + \int p(\mathbf{x}, y = \text{“}+1\text{”}) \mathbb{1}_{\{f(\mathbf{x}) \leq 0\}} d\mathbf{x}\right] \\ &= \underset{f(\mathbf{x})}{\text{argmin}} \left[\int p(\mathbf{x}, y = \text{“}-1\text{”}) \mathbb{1}_{\{f(\mathbf{x}) \geq 0\}} d\mathbf{x} + \int p(\mathbf{x}, y = \text{“}+1\text{”})(1 - \mathbb{1}_{\{f(\mathbf{x}) \geq 0\}}) d\mathbf{x}\right] \\ &= \underset{f(\mathbf{x})}{\text{argmin}} \left[\int (p(\mathbf{x}, y = \text{“}-1\text{”}) - p(\mathbf{x}, y = \text{“}+1\text{”})) \mathbb{1}_{\{f(\mathbf{x}) \geq 0\}} d\mathbf{x} + \int p(\mathbf{x}, y = \text{“}+1\text{”}) d\mathbf{x}\right] \\ &= \underset{f(\mathbf{x})}{\text{argmin}} \left[\int (p(\mathbf{x}, y = \text{“}-1\text{”}) - p(\mathbf{x}, y = \text{“}+1\text{”})) \mathbb{1}_{\{f(\mathbf{x}) \geq 0\}} d\mathbf{x}\right] \\ &= p(\mathbf{x}, y = \text{“}+1\text{”}) - p(\mathbf{x}, y = \text{“}-1\text{”}) \end{aligned}$$