Advanced Machine Learning

Cristian Perez Jensen

December 31, 2024

Note that these are not the official lecture notes of the course, but only notes written by a student of the course. As such, there might be mistakes. The source code can be found at `github.com/cristianpjensen/eth-cs-notes`. If you find a mistake, please create an issue or open a pull request.

Contents

List of symbols

$\doteq$	Equality by definition
$\approx$	Approximate equality
$\propto$	Proportional to
$\mathbb{N}$	Set of natural numbers
$\mathbb{R}$	Set of real numbers
$i : j$	Set of natural numbers between i and j. *I.e.*, $\{i, i+1, \dots, j\}$
$f : A \to B$	Function f that maps elements of set A to elements of set B
$\mathbb{1}\{\text{predicate}\}$	Indicator function (1 if predicate is true, otherwise 0)

$\boldsymbol{v} \in \mathbb{R}^n$	n-dimensional vector
$\boldsymbol{M} \in \mathbb{R}^{m \times n}$	$m \times n$ matrix
$\boldsymbol{M}^\top$	Transpose of matrix $\boldsymbol{M}$
$\boldsymbol{M}^{-1}$	Inverse of matrix $\boldsymbol{M}$
$\det(\boldsymbol{M})$	Determinant of $\boldsymbol{M}$

$\frac{\mathrm{d}}{\mathrm{d}x} f(x)$	Ordinary derivative of $f(x)$ w.r.t. x at point $x \in \mathbb{R}$
$\frac{\partial}{\partial x} f(\boldsymbol{x})$	Partial derivative of $f(\boldsymbol{x})$ w.r.t. x at point $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}) \in \mathbb{R}^n$	Gradient of $f : \mathbb{R}^n \to \mathbb{R}$ at point $\boldsymbol{x} \in \mathbb{R}^n$
$\boldsymbol{\nabla}_{\boldsymbol{x}}^2 f(\boldsymbol{x}) \in \mathbb{R}^{n \times n}$	Hessian of $f : \mathbb{R}^n \to \mathbb{R}$ at point $\boldsymbol{x} \in \mathbb{R}^n$

1 Paradigms of data science

Let $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ be i.i.d. samples, generated by an unknown distribution P. Assume that this distribution is in a distribution family,

$$\mathcal{H} = \{p(\cdot \mid \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Theta\}.$$

The goal is to learn the parameters $\boldsymbol{\theta}$ that fit the data $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ best.

Frequentism. In frequentism, the maximum likelihood estimator (MLE) parameters maximize the following,

$$\begin{aligned}\boldsymbol{\theta}^\star &\in \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \log p(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \mid \boldsymbol{\theta}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n} \log p(\boldsymbol{x}_i \mid \boldsymbol{\theta}).\end{aligned}$$

Bayesianism. Bayesianism assumes that there is a prior over distributions. The maximum a posteriori (MAP) parameters maximize the following,

$$\begin{aligned}\boldsymbol{\theta}^\star &\in \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \log p(\boldsymbol{\theta} \mid \boldsymbol{X}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \log p(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \mid \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta}) && \text{Bayes' rule.} \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \log p(\boldsymbol{\theta}) + \sum_{i=1}^{n} \log p(\boldsymbol{x}_i \mid \boldsymbol{\theta}).\end{aligned}$$

In practice, the prior acts as a regularization term.

Statistical learning. Now, assume that we have labeled samples $\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$, where y is the target variable. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ be a loss function. For a predictor function $f : \mathcal{X} \to \mathcal{Y}$, we define its risk as the expected loss,

$$\mathcal{R}(f) \doteq \mathbb{E}_{X,Y}[\ell(y, f(\boldsymbol{x}))].$$

In statistical learning, we want to find a function that minimizes the risk. However, since the distribution over X, Y is unknown, we cannot compute $\mathcal{R}(f)$ directly. Instead, we use the empirical risk as a surrogate,

$$\hat{\mathcal{R}}(f) \doteq \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f(\boldsymbol{x}_i)).$$

The goal is to obtain the empirical risk minimizer,

$$f^\star \in \operatorname*{argmin}_{f\in\mathcal{F}} \hat{\mathcal{R}}(f),$$

where $\mathcal{F}$ is a family of functions that we assume f belongs to.

2 *Anomaly detection*

In anomaly detection, we are given a sample of objects $\mathcal{X} \subseteq \mathbb{R}^d$ with a normal class $\mathcal{N} \in \mathcal{X}$—the data points that we wish to keep. We wish to construct a function $\phi : \mathcal{X} \to \{0, 1\}$, such that

$$\phi(\boldsymbol{x}) = 1 \iff \boldsymbol{x} \notin \mathcal{N}.$$

Formally, an anomaly is an unlikely event. Hence, the strategy is to fit a model of a parametric family of distributions to the data $\mathcal{X}$,

$$\mathcal{H} = \{p(\cdot \mid \boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Theta\}.$$

Then, we define the anomaly score of $\boldsymbol{x}$ as a low probability $p(\boldsymbol{x} \mid \boldsymbol{\theta}^\star)$ according to the optimal model in this hypothesis class.

Anomaly detection in a high-dimensional space is hard, because the normal class can be very complex. The idea is to project $\mathcal{X}$ down to a lower dimensionality and perform anomaly detection there—hopefully the projected version of the normal class $\Pi(\mathcal{N})$ is less complex. In order to find the optimal linear projection, we will use principal component analysis (PCA).

Furthermore, it has been observed that linear projections of high-dimensional distributions onto low-dimensional spaces resemble Gaussian distributions. Hence, after performing PCA, we will fit a Gaussian mixture model (GMM) to the projected data.

Principal component analysis. The goal of PCA is to linearly project $\mathbb{R}^d$ to $\mathbb{R}^{d^-}$ such that the maximum amount of variance of the data is preserved.[1] Consider the base case $d^- = 1$. Let $\boldsymbol{u} \in \mathbb{R}^d$ with $\|\boldsymbol{u}\| = 1$, we project onto $\boldsymbol{u}$ by inner product,

$$\boldsymbol{x} \mapsto \boldsymbol{u}^\top \boldsymbol{x}.$$

The sample mean of the reduced dataset is computed by

$$\frac{1}{n} \sum_{i=1}^{n} \boldsymbol{u}^\top \boldsymbol{x}_i = \boldsymbol{u}^\top \bar{\boldsymbol{x}},$$

where $\boldsymbol{x}$ is the sample mean of the original dataset. Further, the sample variance of the reduced dataset is

$$\begin{aligned}\frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{u}^\top \boldsymbol{x}_i - \boldsymbol{u}^\top \bar{\boldsymbol{x}}\right)^2 &= \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{u}^\top (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^\top \boldsymbol{u} \\ &= \boldsymbol{u}^\top \left(\frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x} - \bar{\boldsymbol{x}})(\boldsymbol{x} - \bar{\boldsymbol{x}})^\top\right) \boldsymbol{u} \\ &= \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u},\end{aligned}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the dataset. Since we want the projection that preserves the maximum variance, we have the following objective,

$$\boldsymbol{u}^\star \in \operatorname*{argmax}_{\|\boldsymbol{u}\|=1} \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u}.$$

[1] Components with larger variance are more informative.

The Lagrangian of this problem is

$$\mathcal{L}(\boldsymbol{u}; \lambda) = \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u} + \lambda \left(1 - \|\boldsymbol{u}\|^2\right)$$

with gradient

$$\frac{\partial \mathcal{L}(\boldsymbol{u}; \lambda)}{\partial \boldsymbol{u}} = 2\boldsymbol{\Sigma}\boldsymbol{u} - 2\lambda\boldsymbol{u} \overset{!}{=} 0.$$

So, $\boldsymbol{u}$ must satisfy $\boldsymbol{\Sigma}\boldsymbol{u} = \lambda\boldsymbol{u}$—$\boldsymbol{u}$ is an eigenvector of $\boldsymbol{\Sigma}$. It is easy to see that this must be the principal eigenvector by rewriting the objective,

$$\begin{aligned} \boldsymbol{u}^\star &\in \operatorname*{argmax}_{\|\boldsymbol{u}\|=1} \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u} \\ &= \operatorname*{argmax}_{\substack{\|\boldsymbol{u}\|=1 \\ (\boldsymbol{u},\lambda)\in \operatorname{eig}(\boldsymbol{\Sigma})}} \lambda \|\boldsymbol{u}\|^2 \\ &= \operatorname*{argmax}_{\substack{\boldsymbol{u}\in\mathbb{R}^d \\ (\boldsymbol{u},\lambda)\in \operatorname{eig}(\boldsymbol{\Sigma})}} \lambda \\ &= \boldsymbol{u}_1. \end{aligned}$$

For $d^- > 1$, the remaining principal components can be computed with a similar idea. Iteratively, we factor out the previous principal components and do as above on the transformed dataset. For example, to get the second principal component, we first factor out the first principal component,

$$\mathcal{X}_1 \doteq \{\boldsymbol{x} - \operatorname{proj}_{\boldsymbol{u}_1}(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}\} = \{\boldsymbol{x} - \boldsymbol{u}_1^\top \boldsymbol{x} \cdot \boldsymbol{u}_1 \mid \boldsymbol{x} \in \mathcal{X}\}.$$

Then, we do the same as above.

Gaussian mixture model. The probability density function (PDF) of a Gaussian mixture model with k components is formalized as a convex combination of Gaussians,

$$p(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{j=1}^{k} \pi_j \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j).$$

The parameters of this model are

$$\boldsymbol{\theta} = \{\pi_j, \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \mid j \in [k]\},$$

where $\sum_{j=1}^{k} \pi_j = 1$ and $\{\boldsymbol{\Sigma}_j \mid j \in [k]\}$ are positive definite. We fit the parameters of this model by maximizing the log-likelihood,

$$\begin{aligned} \boldsymbol{\theta}^\star &\in \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \log p(\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}; \boldsymbol{\theta}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n} \log p(\boldsymbol{x}_i; \boldsymbol{\theta}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \sum_{i=1}^{n} \log \sum_{j=1}^{k} \pi_j \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j). \end{aligned}$$

```
Initialize $\boldsymbol{\theta}_0$
for $t \in [T]$ do
    $q^\star \in \operatorname{argmin}_q E(q, \boldsymbol{\theta}_{t-1})$
    $\boldsymbol{\theta}_t \in \operatorname{argmax}_{\boldsymbol{\theta}} M(q^\star, \boldsymbol{\theta})$
end for
return $\boldsymbol{\theta}_T$
```

Algorithm 1. The expectation-maximization algorithm, where

$$M(q, \boldsymbol{\theta}) \doteq \mathbb{E}_{z \sim q}\left[\log \frac{p(\boldsymbol{X}, z; \boldsymbol{\theta})}{q(z)}\right]$$
$$E(q, \boldsymbol{\theta}) \doteq \mathbb{E}_{z \sim q}\left[\log \frac{q(z)}{p(z \mid \boldsymbol{X}; \boldsymbol{\theta})}\right].$$

Note that the above is intractable, so we would like to decompose it into tractable terms that can be computed. Let's assume that we know from which latent variable each data point was generated, then we can compute the MLE of the extended dataset $\{(\boldsymbol{x}_i, z_i) \mid i \in [n]\}$ as

$$\begin{aligned}
\log p(\boldsymbol{X}, \boldsymbol{z}; \boldsymbol{\theta}) &= \sum_{i=1}^{n} \log p(\boldsymbol{x}_i, z_i) \\
&= \sum_{i=1}^{n} \log(\pi_{z_i} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i})) \\
&= \sum_{i=1}^{n} \log \pi_{z_i} + \sum_{i=1}^{n} \log \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{z_i}, \boldsymbol{\Sigma}_{z_i}),
\end{aligned}$$

which is tractable to maximize. Let q be a distribution over $[k]$, then we can rewrite the log-likelihood into tractable terms,

$$\begin{aligned}
\log p(\boldsymbol{X}; \boldsymbol{\theta}) &= \mathbb{E}_{z \sim q}[\log p(\boldsymbol{X}; \boldsymbol{\theta})] \\
&= \mathbb{E}_{z \sim q}\left[\log\left(\frac{p(\boldsymbol{X}, z; \boldsymbol{\theta})}{p(z \mid \boldsymbol{X}; \boldsymbol{\theta})}\right)\right] \\
&= \mathbb{E}_{z \sim q}\left[\log\left(\frac{p(\boldsymbol{X}, z; \boldsymbol{\theta})}{p(z \mid \boldsymbol{X}; \boldsymbol{\theta})} \frac{q(z)}{q(z)}\right)\right] \\
&= \underbrace{\mathbb{E}_{z \sim q}\left[\log \frac{p(\boldsymbol{X}, z; \boldsymbol{\theta})}{q(z)}\right]}_{\doteq M(q, \boldsymbol{\theta})} + \underbrace{\mathbb{E}_{z \sim q}\left[\log \frac{q(z)}{p(z \mid \boldsymbol{X}; \boldsymbol{\theta})}\right]}_{\doteq E(q, \boldsymbol{\theta})}.
\end{aligned}$$

These terms have the following two properties,

$$\begin{aligned}
&\log p(\boldsymbol{X}; \boldsymbol{\theta}) \geq M(q, \boldsymbol{\theta}), \quad \forall q, \boldsymbol{\theta} \\
&\log p(\boldsymbol{X}; \boldsymbol{\theta}) = M(q^\star, \boldsymbol{\theta}), \quad q^\star = p(\cdot \mid \boldsymbol{X}; \boldsymbol{\theta}), \quad \forall \boldsymbol{\theta}.
\end{aligned}$$

Hence, we can use $M(q^\star, \boldsymbol{\theta})$ as an approximation of $\log p(\boldsymbol{X}; \boldsymbol{\theta})$ around $\boldsymbol{\theta}$.

Theorem 2.1 (EM algorithm convergence)**.** Using the expectation-maximization algorithm, $\{\log p(\boldsymbol{x}; \boldsymbol{\theta}_t)\}_{t=0}^{T}$ is non-decreasing.

Proof. Given a data point $\boldsymbol{x}$ and current parameters $\boldsymbol{\theta}$, we have the following update,

$$\boldsymbol{\theta}' \in \operatorname*{argmax}_{\boldsymbol{\theta} \in \Theta} M(q^\star, \boldsymbol{\theta}).$$

Hence, we have

$$\log p(\boldsymbol{x}) = M(q^{\star}, \boldsymbol{\theta}) \leq M(q^{\star}, \boldsymbol{\theta}') \leq \log p(\boldsymbol{x}; \boldsymbol{\theta}').$$

Thus, $\{\log p(\boldsymbol{x}; \boldsymbol{\theta}_t)\}_{t=0}^{T}$ is non-decreasing. ■

Summary. In conclusion, given a set of data points $\mathcal{X}$ with normal points $\mathcal{N} \subseteq \mathcal{X}$, we train an anomaly detector as follows,

1. Fit a projector $\pi : \mathbb{R}^d \to \mathbb{R}^{d^-}$ using PCA;
2. Fit a probability density function $p(\cdot \mid \boldsymbol{\theta})$ with k components to $\{\pi(\boldsymbol{x}) \mid \boldsymbol{x} \in \mathcal{X}\}$ using the EM algorithm;
3. For a point $\boldsymbol{x} \in \mathcal{X}$, its "anomaly score" is computed by $-\log p(\pi(\boldsymbol{x}); \boldsymbol{\theta})$.

3 Density estimation

In this section, we will consider parametric models,

$$\{p(\boldsymbol{x};\boldsymbol{\theta}) \mid \boldsymbol{\theta} \in \Theta\}.$$

The problem we concern ourselves with is finding the best parameter $\boldsymbol{\theta}$. The most common method of finding the best parameters is maximum likelihood estimation (MLE),

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{MLE}} &\in \operatorname*{argmax}_{\boldsymbol{\theta}\in\Theta} \prod_{i=1}^{n} p(\boldsymbol{x}_i;\boldsymbol{\theta}) \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}\in\Theta} -\sum_{i=1}^{n} \log p(\boldsymbol{x}_i;\boldsymbol{\theta}).\end{aligned}$$

The following properties makes the MLE estimator attractive,

1. Consistency—in the limit of n, $\hat{\boldsymbol{\theta}}_{\text{MLE}}$ converges to the true parameter $\boldsymbol{\theta}^{\star}$;
2. Equivariance—if $\hat{\boldsymbol{\theta}}$ is the MLE of $\boldsymbol{\theta}$, then $g(\hat{\boldsymbol{\theta}})$ is the MLE of $g(\boldsymbol{\theta})$ for any function g;
3. Asymptotically normal—in the limit of n, ${}^{\hat{\boldsymbol{\theta}}_{\text{MLE}}-\boldsymbol{\theta}}/_{\sqrt{n}}$ converges to a random variable with distribution $\mathcal{N}(0, \mathcal{I}(\boldsymbol{\theta})^{-1})$, where $\mathcal{I}$ is the Fisher information matrix;
4. Asymptotically efficient—in the limit of n, the MLE estimator has the smallest variance of all unbiased estimators.

We can understand the asymptotic efficiency (property 4) of estimators better by the Rao–Cramér bound, which provides a bound on the variance of the estimator. We will only consider the general scalar case, but it generalizes to the multivariate case.

Theorem 3.1 (Rao–Cramér bound (scalar case)). For any (scalar) unbiased estimator $\hat{\theta} : \mathcal{Y}^n \to \mathbb{R}$, given n data points, of $\theta \in \mathbb{R}$, we have the following bound on its variance,

$$\operatorname{Var}\left[\hat{\theta}(\boldsymbol{y})\right] \geq \frac{\left(\frac{\partial}{\partial\theta} b_{\hat{\theta}} + 1\right)^2}{\mathcal{I}_n(\theta)} + b_{\hat{\theta}}^2,$$

where $\mathcal{I}_n(\theta)$ is the Fisher information,

$$\mathcal{I}_n(\theta) \doteq \mathbb{E}_{\boldsymbol{y}\mid\theta}\left[\left(\frac{\partial}{\partial\theta} \log p(\boldsymbol{y} \mid \theta)\right)^2\right] \overset{\text{iid}}{=} n \cdot \mathcal{I}(\theta),$$

and $b_{\hat{\theta}}$ is the bias of $\hat{\theta}$,

$$b_{\hat{\theta}} \doteq \mathbb{E}_{\boldsymbol{y}\mid\theta}\left[\hat{\theta}(\boldsymbol{y})\right] - \theta.$$

Proof. Let the "score" be defined as follows,

$$\Lambda(\boldsymbol{y},\theta) \doteq \frac{\partial}{\partial\theta}\log p(\boldsymbol{y} \mid \theta) = \frac{1}{p(\boldsymbol{y} \mid \theta)}\frac{\partial}{\partial\theta}p(\boldsymbol{y} \mid \theta).$$

Then we have $\mathcal{I}_n(\theta) = \mathbb{E}_{\boldsymbol{y}|\theta}\left[\Lambda(\boldsymbol{y},\theta)^2\right]$.

The expected score is equal to zero,

$$\begin{aligned}
\mathbb{E}_{\boldsymbol{y}|\theta}[\Lambda(\boldsymbol{y},\theta)] &= \int p(\boldsymbol{y} \mid \theta)\Lambda(\boldsymbol{y},\theta)\mathrm{d}\boldsymbol{y} \\
&= \int \frac{\partial}{\partial\theta}p(\boldsymbol{y} \mid \theta)\mathrm{d}\boldsymbol{y} \\
&= \frac{\partial}{\partial\theta}\int p(\boldsymbol{y} \mid \theta)\mathrm{d}\boldsymbol{y} \\
&= \frac{\partial}{\partial\theta}1 \\
&= 0.
\end{aligned}$$

(Hence, the Fisher information is equivalent to the variance of the score.) Furthermore, the cross-correlation between $\Lambda(\boldsymbol{y},\theta)$ and $\hat{\theta}(\boldsymbol{y})$ can be computed as

$$\begin{aligned}
\mathrm{Cov}_{\boldsymbol{y}|\theta}\left(\Lambda(\boldsymbol{y},\theta),\hat{\theta}(\boldsymbol{y})\right) &= \mathbb{E}_{\boldsymbol{y}|\theta}\left[(\Lambda(\boldsymbol{y},\theta) - \mathbb{E}[\Lambda(\boldsymbol{y},\theta)])\left(\hat{\theta}(\boldsymbol{y}) - \mathbb{E}\left[\hat{\theta}(\boldsymbol{y})\right]\right)\right] \\
&= \mathbb{E}_{\boldsymbol{y}|\theta}\left[\Lambda(\boldsymbol{y},\theta)\hat{\theta}(\boldsymbol{y})\right] - \mathbb{E}_{\boldsymbol{y}|\theta}[\Lambda(\boldsymbol{y},\theta)]\mathbb{E}_{\boldsymbol{y}|\theta}\left[\hat{\theta}(\boldsymbol{y})\right] \\
&= \mathbb{E}_{\boldsymbol{y}|\theta}\left[\Lambda(\boldsymbol{y},\theta)\hat{\theta}(\boldsymbol{y})\right] \\
&= \int p(\boldsymbol{y} \mid \theta)\Lambda(\boldsymbol{y},\theta)\hat{\theta}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\
&= \int \frac{\partial}{\partial\theta}p(\boldsymbol{y} \mid \theta)\hat{\theta}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\
&= \frac{\partial}{\partial\theta}\int p(\boldsymbol{y} \mid \theta)\hat{\theta}(\boldsymbol{y})\mathrm{d}\boldsymbol{y} \\
&= \frac{\partial}{\partial\theta}\left(\mathbb{E}_{\boldsymbol{y}|\theta}\left[\hat{\theta}(\boldsymbol{y})\right] - \theta\right) + 1 \\
&= \frac{\partial}{\partial\theta}b_{\hat{\theta}} + 1.
\end{aligned}$$

Using the Cauchy-Schwarz inequality, we have the following bound,

$$\begin{aligned}
\mathrm{Cov}_{\boldsymbol{y}|\theta}\left(\Lambda(\boldsymbol{y},\theta),\hat{\theta}(\boldsymbol{y})\right)^2 &= \left(\mathbb{E}_{\boldsymbol{y}|\theta}\left[\Lambda(\boldsymbol{y},\theta)\left(\hat{\theta}(\boldsymbol{y}) - \mathbb{E}\left[\hat{\theta}(\boldsymbol{y})\right]\right)\right]\right)^2 \\
\left(\frac{\partial}{\partial\theta}b_{\hat{\theta}} + 1\right)^2 &\leq \mathbb{E}_{\boldsymbol{y}|\theta}\left[\Lambda(\boldsymbol{y},\theta)^2\right] \cdot \mathbb{E}_{\boldsymbol{y}|\theta}\left[\left(\hat{\theta}(\boldsymbol{y}) - \mathbb{E}\left[\hat{\theta}(\boldsymbol{y})\right]\right)^2\right] \\
&= \mathcal{I}_n(\theta) \cdot \mathbb{E}_{\boldsymbol{y}|\theta}\left[\left(\hat{\theta}(\boldsymbol{y}) - \theta - \mathbb{E}\left[\hat{\theta}(\boldsymbol{y}) + \theta\right]\right)^2\right] \\
&= \mathcal{I}_n(\theta) \cdot \left(\mathbb{E}_{\boldsymbol{y}|\theta}\left[\left(\hat{\theta}(\boldsymbol{y}) - \theta\right)^2\right] + \left(\mathbb{E}_{\boldsymbol{y}|\theta}\left[\hat{\theta}\right] - \theta\right)^2 \right. \\
&\qquad \left. - 2\mathbb{E}_{\boldsymbol{y}|\theta}\left[\hat{\theta}(\boldsymbol{y}) - \theta\right]^2\right) \\
&\leq \mathcal{I}_n(\theta) \cdot \left(\mathbb{E}_{\boldsymbol{y}|\theta}\left[\left(\hat{\theta}(\boldsymbol{y}) - \theta\right)^2\right] - b_{\hat{\theta}}^2\right).
\end{aligned}$$

Re-arranging yields

$$\mathbb{E}_{\boldsymbol{y}|\theta}\left[\left(\hat{\theta}(\boldsymbol{y}) - \theta\right)^2\right] \geq \frac{\left(\frac{\partial}{\partial\theta}b_{\hat{\theta}} + 1\right)^2}{\mathcal{I}_n(\theta)} + b_{\hat{\theta}}^2.$$

■

Note the trade-off between variance and bias. If $\frac{\partial}{\partial\theta} b_{\hat{\theta}} < 0$, then the variance might be smaller than the variance of the best unbiased estimator.

Corollary. Let $\hat{\theta}$ be an unbiased estimator—*i.e.*, $b_{\hat{\theta}} = 0$—then

$$\mathrm{Var}\left[\hat{\theta}(\boldsymbol{y})\right] \geq \frac{1}{\mathcal{I}_n(\theta)}.$$

Lemma 3.2. The maximum likelihood estimator $\hat{\theta}_{\mathrm{ML}}$ is asymptotically efficient,

$$\lim_{n\to\infty} \mathrm{Var}\left[\hat{\theta}_{\mathrm{ML}}\right] = \frac{1}{\mathcal{I}_n(\theta)}.$$

However, the MLE estimator is not necessarily efficient for finite samples.

4 Regression

In regression, we try to estimate a function $f : \mathbb{R}^d \to \mathbb{R}$ that best fits a given dataset $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n \subseteq \mathbb{R}^d \times \mathbb{R}$. *I.e.*, we try to minimize the following loss function,

$$\ell(\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=1}^{n}\|f(\boldsymbol{x}_i) - y_i\|^2.$$

4.1 Linear regression

In linear regression, we assume the underlying data to be linear with a fixed noise,

$$Y \mid \boldsymbol{X} = \boldsymbol{x} \sim \mathcal{N}\left(\boldsymbol{\beta}_\star^\top \boldsymbol{x}, \sigma^2\right)$$

for some ground truth $\boldsymbol{\beta}_\star$. We parametrize f as a linear function,

$$f(\boldsymbol{x}; \boldsymbol{\beta}) = \boldsymbol{\beta}^\top \boldsymbol{x}.$$

Under these assumptions, the minimizer of the loss function is analytically tractable—the ordinary least squares estimator (OLSE),

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y},$$

where $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ is the design matrix with respective outputs $\boldsymbol{y} \in \mathbb{R}^n$.

In practice, it is important to remove outliers, because linear models can be heavily influenced by them. Also, the data should be standardized, such that all features are on the same scale, because differences in scale can make matrix inversion unstable.

TODO: Curse of dimensionality [Sur and Candès, 2019], Figures 4.1 and 4.2.

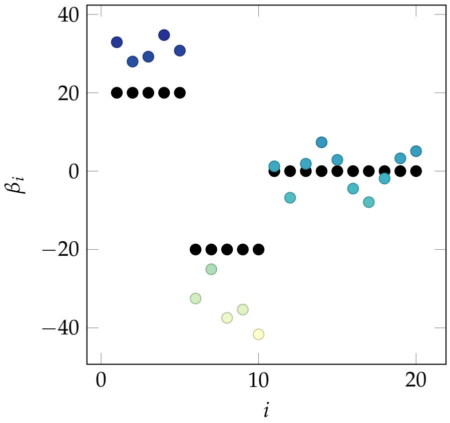

Figure 4.1. $\boldsymbol{\beta}^\star$ is shown as black marks and $\hat{\boldsymbol{\beta}}$ is indicated by the marks. As can be seen, $\hat{\boldsymbol{\beta}}$ is overestimated for indices where $\beta_i^\star \neq 0$.

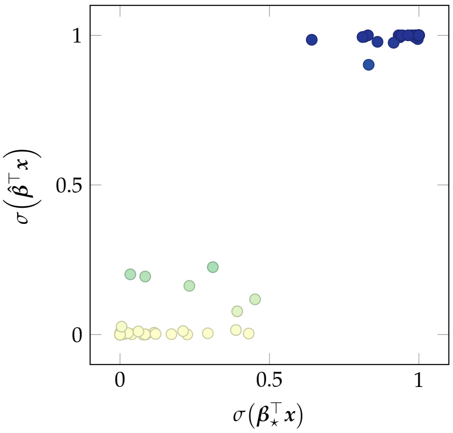

Figure 4.2. True *vs.* predicted probability—the estimator is overconfident in its predictions.

Further, if features are collinear, then the model might only learn the correlation with the target variable of one of them and discard the other. In addition, some singular values will equal zero. This makes matrix inversion unstable. This can easily be shown by considering the singular value decomposition $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$, then

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} \\
&= \left(\boldsymbol{V}\boldsymbol{D}^\top \boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top\right)^{-1} \boldsymbol{V}\boldsymbol{D}^\top \boldsymbol{U}^\top \boldsymbol{y} \\
&= \left(\boldsymbol{V}\boldsymbol{D}^\top \boldsymbol{D}\boldsymbol{V}^\top\right)^{-1} \boldsymbol{V}\boldsymbol{D}^\top \boldsymbol{U}^\top \boldsymbol{y} \\
&= \boldsymbol{V}\left(\boldsymbol{D}^{-1}\right)^\top \boldsymbol{D}^{-1}\boldsymbol{V}^\top \boldsymbol{V}\boldsymbol{D}^\top \boldsymbol{U}^\top \boldsymbol{y} \\
&= \boldsymbol{V}\left(\boldsymbol{D}^{-1}\right)^\top \boldsymbol{D}^{-1}\boldsymbol{D}^\top \boldsymbol{U}^\top \boldsymbol{y} \\
&= \boldsymbol{V}\boldsymbol{D}^{-1}\boldsymbol{U}^\top \boldsymbol{y}.
\end{aligned}$$

The inversion of $\boldsymbol{D}$ is unstable if the singular values are small. The solution to this is to remove collinear features and/or to add regularization.

Consider the mean-squared error loss. The risk (expected loss) can be decomposed as follows into bias, variance, and noise terms,

$$\mathbb{E}\left[(f(\boldsymbol{x})-y)^2\right] = \underbrace{(\mathbb{E}[f(\boldsymbol{x})]-\mathbb{E}[y])^2}_{\text{squared bias}} + \underbrace{\mathrm{Var}[f(\boldsymbol{x})]}_{\text{variance}} + \underbrace{\mathbb{E}\left[(\mathbb{E}[y]-y)^2\right]}_{\text{noise}}.$$

The OLSE is the minimum variance unbiased estimator.[2] However, this does not mean it is the best, because introducing some bias may considerably decrease the variance. Bayesianism adds bias by introducing a prior—priors often induce a regularizing term.[3]

4.2 *Polynomial regression*

In polynomial regression, we construct a feature function of all possible polynomials, *e.g.*,

$$\phi([x_1,x_2]) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, \ldots].$$

We then perform linear regression in this space,

$$\psi(\boldsymbol{x};\boldsymbol{\beta}) = \langle \boldsymbol{\beta}, \phi(\boldsymbol{x})\rangle.$$

The problem is that this space is infinitely dimensional, and the inner product is ill-defined in an infinitely dimensional space.[4] We can solve this by transforming the vector such that inner products cannot diverge by introducing a data-dependent scalar,

$$\phi(\boldsymbol{x}) = \alpha(\boldsymbol{x}) \cdot \tilde{\phi}(\boldsymbol{x}), \quad \alpha(\boldsymbol{x}) > 0.$$

$\tilde{\phi}$ contains all polynomials in this equation.

An inner product w.r.t. this feature function is a valid inner product. Such transformations can have a closed form for the inner product, called kernelization. Commonly, the radial basis function (RBF) kernel is used,

$$\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle = \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{x}'\|^2}{2\sigma^2}\right), \quad \sigma \in \mathbb{R}.$$

Let $\boldsymbol{\Phi} \in \mathbb{R}^{n\times\infty}$ contain all feature vectors, then the OLSE for polynomial regression can be computed by

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^\top \boldsymbol{y}, \quad \boldsymbol{\Phi} \in \mathbb{R}^{n\times\infty}.$$

However, $\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$ cannot be computed, because it is $\infty \times \infty$-dimensional. We can fix this by observing that $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \in \mathbb{R}^{n\times n}$ and rewriting the OLSE as

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{y} \\
&= \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^\top\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}\boldsymbol{y} \\
&= \left(\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}\boldsymbol{y} \\
&= \boldsymbol{\Phi}^\top\left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\right)^{-1}\boldsymbol{y}.
\end{aligned}$$

[2] This is proven by the Gauss-Markov theorem.

[3] *E.g.*, a Gaussian prior induces an ℓ_2-norm regularizing term.

[4] This is due to $\langle\phi(\boldsymbol{x}),\phi(\boldsymbol{x}')\rangle = \infty$ for some $\boldsymbol{x},\boldsymbol{x}'$, *e.g.*, $\boldsymbol{x} = \boldsymbol{x}' = [1,1]$.

Let $\boldsymbol{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$, then it only contains kernel evaluations—$k_{ij} = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle$. The next problem is that $\boldsymbol{\Phi}^\top$ is still infinitely dimensional. However, this is not a problem, since when we want to make a prediction, we do the following,

$$\begin{aligned} \psi(\hat{x}) &= \langle \phi(\hat{x}), \hat{\boldsymbol{\beta}} \rangle \\ &= \phi(\hat{x})^\top \boldsymbol{\Phi}^\top \boldsymbol{K}^{-1} \boldsymbol{y}. \end{aligned}$$

Let $\boldsymbol{k}(\hat{x}) = \phi(\hat{x})^\top \boldsymbol{\Phi}^\top$, then it only contains kernel evaluations with the new point—$k_i(\hat{x}) = \langle \phi(\hat{x}), \phi(\boldsymbol{x}_i) \rangle$. In conclusion, we can make predictions by

$$\psi(\hat{x}) = \boldsymbol{k}(\hat{x}) \boldsymbol{K}^{-1} \boldsymbol{y}, \quad \boldsymbol{k}(\hat{x}) \in \mathbb{R}^{1 \times n}, \boldsymbol{K} \in \mathbb{R}^{n \times n}, \boldsymbol{y} \in \mathbb{R}^n.$$

However, the problem with this approach is that it takes $O(n^3)$ to make a prediction—it scales in the amount of data points.

4.3 *Regularization*

TODO: Ridge regression—ℓ_2-norm.

TODO: LASSO—ℓ_1-norm induces sparsity. Better interpretability properties.

TODO: Figure showing coefficient weights under different regularization factors.

5 Causality

In general, models do not distinguish between causal and non-causal factors in the feature space. Therefore, they might identify non-causal factors as highly correlating with the output variable. *E.g.*, when classifying images as either depicting a cow or a camel, the model might identify the background as an important feature, because cows are usually in grass and camels are usually in the desert. However, if this classifier were to see a cow with a different background, it would fail.

The following are examples of causal fallacies, where one might conclude that X causes Y,

- Reverse causality, where Y actually predicts X and not the other way around;
- Third-cause fallacy, where there is a confounding factor Z that influences both X and Y;
- Bidirectional causation, where X influence each other;

A domain shift happens when the test samples are drawn from a different distribution than the training samples. *E.g.*, COVID-19 detection models trained on a western population might not perform well on an eastern population.

Shortcut learning happens when there is a spurious correlation between causal and non-causal features in the training dataset. The resulting estimator abuses the non-causal features to generalize in the testing dataset. The solution to this is to encode the features such that they do not depend on the environment.

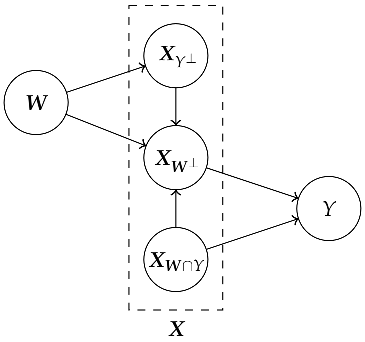
$\boldsymbol{W}$
$\boldsymbol{X}_{Y^\perp}$
$\boldsymbol{X}_{\boldsymbol{W}^\perp}$
$\boldsymbol{X}_{\boldsymbol{W}\cap Y}$
Y
$\boldsymbol{X}$

Figure 5.1. Causal graph.

5.1 Counterfactual invariance

Let $\boldsymbol{X}$ be the feature vector representing the object and let Y be a target variable of interest, and let f be the function that estimates Y from $\boldsymbol{X}$. Further, let $\boldsymbol{W}$ describe features that influence $\boldsymbol{X}$, but should not influence the estimator f. Let a counterfactual be denoted as $\boldsymbol{X}(\boldsymbol{w})$, which is the feature vector we would have obtained if we would have had $\boldsymbol{W} = \boldsymbol{w}$, after the fact. Then, the estimator f is counterfactually invariant if

$$f(\boldsymbol{X}(\boldsymbol{w})) = f(\boldsymbol{X}(\boldsymbol{w}')), \quad \forall \boldsymbol{w}, \boldsymbol{w}' \in \text{range}(\boldsymbol{W}).$$

In words, f is not influenced by the value of $\boldsymbol{W}$. One way of obtaining a counterfactually invariant estimator is by extending the training dataset to contain enough counterfactuals. This can, for example, be achieved by data augmentation. However, this is not always possible.

Let $\boldsymbol{X}_{\boldsymbol{A}}$ be the parts of $\boldsymbol{X}$ causally influenced by $\boldsymbol{A}$, and let $\boldsymbol{A}^\perp$ be the set of variables independent of $\boldsymbol{A}$. Then, f is counterfactually invariant if and only if f only depends on $\boldsymbol{X}_{\boldsymbol{W}^\perp}$.

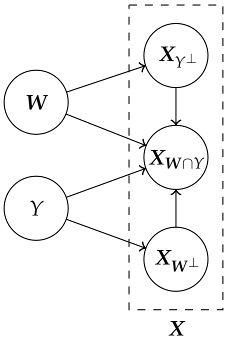
$\boldsymbol{W}$
Y
$\boldsymbol{X}_{Y^\perp}$
$\boldsymbol{X}_{\boldsymbol{W}\cap Y}$
$\boldsymbol{X}_{\boldsymbol{W}^\perp}$
$\boldsymbol{X}$

Figure 5.2. Anti-causal graph.

Theorem 5.1 (Necessary conditions for counterfactual invariance [Veitch et al., 2021]). If f is a counterfactually invariant predictor, then

- In the anti-causal scenario, $f(\boldsymbol{X}) \perp \boldsymbol{W} \mid Y$;
- In the causal scenario without selection (but possibly confounded), $f(\boldsymbol{X}) \perp \boldsymbol{W}$;
- In the causal scenario without confoundedness (but possibly with selection), $f(\boldsymbol{X}) \perp \boldsymbol{W} \mid Y$, as long as $\boldsymbol{X} \perp Y \mid \boldsymbol{X}_{\boldsymbol{W}^{\perp}}, \boldsymbol{W}$.

Proof. This can be proven by using d-separation. ■

6 Gaussian processes

Let the inputs be $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, the outputs be $\boldsymbol{y} \in \mathbb{R}^{n}$, and the weight matrix be $\boldsymbol{\beta} \in \mathbb{R}^{d}$, then linear regression models the generative process as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}\left(\boldsymbol{0}, \sigma^2 \boldsymbol{I}\right).$$

This is equivalent to defining a Gaussian over the predictions,

$$\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\beta} \sim \mathcal{N}\left(\boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I}\right).$$

BLR (***B**ayesian **L**inear **R**egression*) extends linear regression by defining a prior over the regression coefficients,

$$\boldsymbol{\beta} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Lambda}^{-1}\right), \quad \boldsymbol{\Lambda} \in \mathbb{R}^{d\times d}.$$

The posterior can be analytically computed as

$$\boldsymbol{\beta} \mid \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}),$$

where

$$\boldsymbol{\mu} = \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}, \quad \boldsymbol{\Sigma} = \sigma^2\left(\boldsymbol{X}^\top \boldsymbol{X} + \sigma^2 \boldsymbol{\Lambda}\right)^{-1}.$$

Proof.

$$\begin{aligned}
p(\boldsymbol{\beta} \mid \boldsymbol{X}, \boldsymbol{y}) &\propto p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\beta}) \cdot p(\boldsymbol{\beta}) && \text{Bayes' rule.}\\
&= \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{X}\boldsymbol{\beta}, \sigma^2 \boldsymbol{I}\right) \cdot \mathcal{N}\left(\boldsymbol{\beta}; \boldsymbol{0}, \boldsymbol{\Lambda}^{-1}\right)\\
&\propto \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^\top \boldsymbol{\Lambda}\boldsymbol{\beta}\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}\left(\|\boldsymbol{y}\|^2 + \|\boldsymbol{X}\boldsymbol{\beta}\|^2 - 2\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{\beta}\right) + \boldsymbol{\beta}^\top \boldsymbol{\Lambda}\boldsymbol{\beta}\right)\right) && \text{Cosine theorem.}\\
&= \exp\left(-\frac{1}{2}\left(\frac{1}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{y} + \frac{1}{\sigma^2}\boldsymbol{\beta}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\beta} - \frac{2}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\beta}^\top \boldsymbol{\Lambda}\boldsymbol{\beta}\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\boldsymbol{\beta}^\top\left(\frac{1}{\sigma^2}\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{\Lambda}\right)\boldsymbol{\beta} - \frac{2}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{\beta} + \frac{1}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{y}\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\boldsymbol{\beta}^\top \boldsymbol{\Sigma}^{-1}\boldsymbol{\beta} - \frac{1}{\sigma^2}\boldsymbol{\beta}^\top \boldsymbol{X}^\top \boldsymbol{y} - \frac{1}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{\beta} + \frac{1}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{y}\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\boldsymbol{\beta}^\top \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\beta} - \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}\right) - \frac{1}{\sigma^2}\boldsymbol{y}^\top \boldsymbol{X}\left(\boldsymbol{\beta} - \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}\right)\right)\right)\\
&= \exp\left(-\frac{1}{2}\left(\left(\boldsymbol{\beta} - \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}\right)\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{\beta} - \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}\right)\right)\right)\\
&\propto \mathcal{N}\left(\boldsymbol{\beta}; \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{X}^\top \boldsymbol{y}, \boldsymbol{\Sigma}\right).
\end{aligned}$$

■

In general, we do not know the true weights that generated the data points. But, we can now define a joint distribution over output variables with unknown weights,

$$\boldsymbol{y} \mid \boldsymbol{X} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{X}\boldsymbol{\Lambda}^{-1}\boldsymbol{X}^\top + \sigma^2 \boldsymbol{I}\right).$$

Proof. As we saw earlier, we compute outputs as follows,

$$y = X\beta + \epsilon, \quad \epsilon \sim \mathcal{N}\Big(\mathbf{0}, \sigma^2 I\Big).$$

However, now the weights are unknown, so we make use of its prior to compute the distribution over y,

$$\begin{aligned}
\mathbb{E}[y] &= \mathbb{E}[X\beta + \epsilon] \\
&= X\mathbb{E}[\beta] + \mathbb{E}[\epsilon] \\
&= \mathbf{0}. \\
\mathrm{Cov}[y] &= \mathbb{E}\Big[(X\beta + \epsilon)(X\beta + \epsilon)^\top\Big] \\
&= \mathbb{E}\Big[X\beta\beta^\top X^\top\Big] + \mathbb{E}\Big[X\beta\epsilon^\top\Big] + \mathbb{E}\Big[\epsilon\beta^\top X^\top\Big] + \mathbb{E}\Big[\epsilon\epsilon^\top\Big] \\
&= X\mathbb{E}\Big[\beta\beta^\top\Big]X^\top + X\mathbb{E}[\beta]\mathbb{E}\Big[\epsilon^\top\Big] + \mathbb{E}[\epsilon]\mathbb{E}\Big[\beta^\top\Big]X^\top + \sigma^2 I \\
&= X\Lambda^{-1}X^\top + \sigma^2 I.
\end{aligned}$$

β and ϵ are independent.

■

GPs (***G**aussian **P**rocesses*) generalize BLR by observing that we can kernelize the covariance matrix,

$$k(x_i, x_j) = x_i^\top \Lambda^{-1} x_j.$$

We could instead use any other kernel function to model other functions.

6.1 Kernels

Kernel functions specify the similarity between any two data points. They encode assumptions about the function that is to be learned.

Definition 6.1 (Kernel function)**.** A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ over $\mathcal{X} \subset \mathbb{R}$ must satisfy the following properties,

$$\begin{gathered}
k(x, x') = k(x', x) \\
\int k(x, x') f(x) f(x') \mathrm{d}x \mathrm{d}x' \geq 0, \quad \forall f \in L_2.
\end{gathered}$$

Symmetry.

Positive semi-definiteness.

Definition 6.2 (Stationary and isotropic)**.** A kernel $k(x, x')$ is stationary if it only depends on $x - x'$. Further, it is isotropic if it only depends on $\|x - x'\|_2$.

The following are common kernels,

$$\begin{aligned}
k(x, x') &= x^\top x' && && \text{Linear kernel.} \\
k(x, x') &= \Big(x^\top x' + 1\Big)^p, && p \in \mathbb{N} && \text{Polynomial kernel.} \\
k(x, x') &= \exp\left(-\frac{\|x - x'\|_2^2}{\ell^2}\right), && \ell \in \mathbb{R} && \text{RBF (\textit{\textbf{R}adial \textbf{B}asis \textbf{F}unction}) kernel.} \\
k(x, x') &= \tanh\Big(\kappa x^\top x'\Big) - b, && \kappa, b \in \mathbb{R} && \text{Sigmoid kernel.}
\end{aligned}$$

Different kernels have different invariance properties, such as invariance to rotation or translation. In order to learn invariances from data, many samples are needed. So, it might be better to encode them if we know them a priori.

Given two kernel functions k_1, k_2 defined on the same data space and positive scalar $c > 0$, the following are also valid kernels,

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{x}') &= k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}, \boldsymbol{x}') &= k_1(\boldsymbol{x}, \boldsymbol{x}') \cdot k_2(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}, \boldsymbol{x}') &= c \cdot k_1(\boldsymbol{x}, \boldsymbol{x}') \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \exp(k_1(\boldsymbol{x}, \boldsymbol{x}')). \end{aligned}$$

In practice, the kernels are often composed together and hyperparameters are determined by performance on a held out validation dataset.

6.2 *Prediction*

As we saw earlier in the case of BLR, the marginal is jointly Gaussian,

$$\begin{bmatrix} \boldsymbol{y} \\ y^\star \end{bmatrix} \mid \boldsymbol{X}, \boldsymbol{x}^\star \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} + \sigma^2 \boldsymbol{I} & \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}^\star) \\ \boldsymbol{k}(\boldsymbol{x}^\star, \boldsymbol{X}) & k(\boldsymbol{x}^\star, \boldsymbol{x}^\star) \end{bmatrix}\right).$$

Theorem 6.3 (Conditional Gaussian distribution)**.** Given

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}\right).$$

Then, the conditional Gaussian is given by

$$x_2 \mid x_1 = z \sim \mathcal{N}\left(\mu_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(z - \mu_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\right).$$

Using the above theorem, we can easily compute the predictive distribution,

$$y^\star \mid \boldsymbol{x}^\star, \boldsymbol{X}, \boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{k}^\top \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}, c - \boldsymbol{k}^\top \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}\right),$$

where

$$\boldsymbol{K} = \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}), \quad \boldsymbol{k} = \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}^\star), \quad c = k(\boldsymbol{x}^\star, \boldsymbol{x}^\star).$$

Now, we can compute a prediction along with its uncertainty. However, the problem with this approach is that it has $\mathcal{O}(n^3)$ runtime.

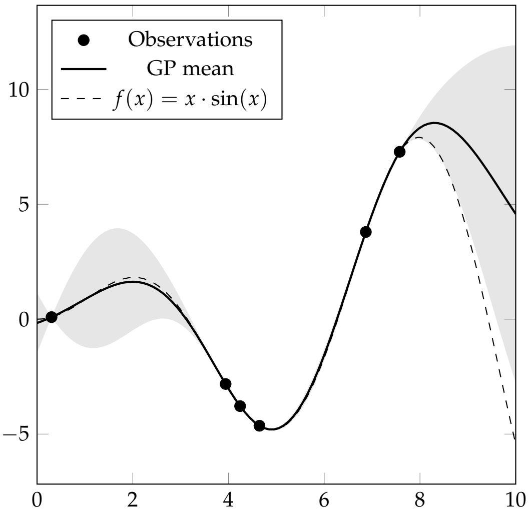

Figure 6.1. Fitted Gaussian process (RBF kernel, $\ell = 1.86$) with its 95% confidence interval.

7 Uncertainty quantification

TODO

7.1 Statistical model validation

TODO

7.2 Bayesian neural networks

TODO

7.3 Transductive active learning

TODO

8 Convex optimization

TODO

8.1 Support vector machine

TODO

9 Ensembles

TODO

9.1 Bagging

TODO

9.2 Random forests

TODO

9.3 AdaBoost

TODO

10 Stable diffusion

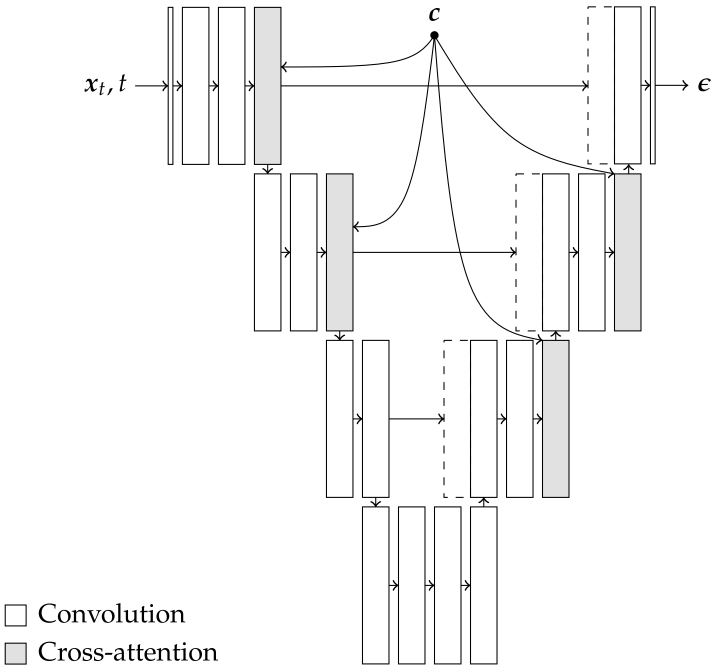

Figure 10.1. Architecture of stable diffusion.

10.1 Diffusion models

An SDE (***S**tochastic **D**ifferential **E**quation*) is a differential equation in which one or more terms is a stochastic process, resulting in a solution that is also stochastic. Typically, the SDE of a diffusion process is of the following form,

$$\mathrm{d}\boldsymbol{X}_t = \boldsymbol{\mu}(\boldsymbol{X}_t, t)\mathrm{d}t + \sigma(\boldsymbol{X}_t, t)\mathrm{d}\boldsymbol{W}_t,$$

where $\boldsymbol{W}_t$ is a Wiener process (or Brownian motion). This equations tells us that the change in $\boldsymbol{X}_t$ is driven by a deterministic factor $\boldsymbol{\mu}(\boldsymbol{X}_t, t)$ and a stochastic factor $\sigma(\boldsymbol{X}_t, t)d\boldsymbol{W}_t$. Note that $\boldsymbol{\mu}(\cdot, \cdot)$ and $\sigma(\cdot, \cdot)$ induce a probability distribution over time, p_t of $\boldsymbol{X}_t$.

Anderson [1982] showed that the reverse SDE of the diffusion process can be computed as follows,

$$\mathrm{d}\boldsymbol{X}_t = \Big[\boldsymbol{\mu}(\boldsymbol{X}_t, t) - \sigma^2(\boldsymbol{X}_t, t)\boldsymbol{\nabla}_X \log p_t(\boldsymbol{X}_t)\Big]\mathrm{d}t + \sigma(\boldsymbol{X}_t, t)\mathrm{d}\bar{\boldsymbol{W}}_t,$$

where $\bar{\boldsymbol{W}}_t$ is a standard Wiener process when time flows backwards from T to 0, and $\mathrm{d}t$ is an infinitesimal negative timestep.

Using the DDPM scheduler, diffusion models have the following forward process,

$$\boldsymbol{x}_{t+1} = \sqrt{1 - \beta_t}\boldsymbol{x}_t + \sqrt{\beta_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}).$$

Conditioned on $\boldsymbol{x}_t$, we can reconstruct $\boldsymbol{x}_{t-1}$ as follows with a predicted noise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$,

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right) + \sqrt{1-\alpha_t}\boldsymbol{z}, \quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{1}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{\tau=1}^{t} \alpha_t$. Thus, we need to learn this function.

It can be shown that a diffusion model with the DDPM scheduler is an approximation of a discretization of the following SDE,

$$\mathrm{d}\boldsymbol{x}_t = -\frac{1}{2}\beta_t \boldsymbol{x}_t \mathrm{d}t + \sqrt{\beta_t}\mathrm{d}\boldsymbol{w}_t.$$

Drift term $\boldsymbol{\mu}(\boldsymbol{x}_t, t) \doteq -\frac{1}{2}\beta_t \boldsymbol{x}_t$; diffusion term $\sigma(t) \doteq \sqrt{\beta_t}$. $\mathrm{d}\boldsymbol{w}_t$ is a Gaussian with variance dt. We can show that this approximates the diffusion model by discretizing,

$$\begin{aligned}\boldsymbol{x}_{t+1} - \boldsymbol{x}_t &= -\frac{1}{2}\beta_t \boldsymbol{x}_t + \sqrt{\beta_t}\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})\\ \boldsymbol{x}_{t+1} &= \left(1 - \frac{1}{2}\beta_t\right)\boldsymbol{x}_t + \sqrt{\beta_t}\boldsymbol{\epsilon}\\ &\approx \sqrt{1-\beta_t}\boldsymbol{x}_t + \sqrt{\beta_t}\boldsymbol{\epsilon}.\end{aligned}$$

The reverse process is thus given by the following reverse SDE,

$$\mathrm{d}\boldsymbol{x}_t = \left[-\frac{1}{2}\beta_t \boldsymbol{x}_t - \beta_t \boldsymbol{\nabla}_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right]\mathrm{d}t + \sqrt{\beta_t}\mathrm{d}\tilde{\boldsymbol{w}}_t.$$

In practice, we train a diffusion model by randomly sampling $\boldsymbol{x}_0 \sim p_0, t \sim \mathrm{Unif}([T]), \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ and performing a gradient step on the following loss function,

$$\ell = \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{1-\beta_t}\boldsymbol{x}_0 + \sqrt{\beta_t}\boldsymbol{\epsilon}\right)\right\|^2.$$

We can sample by iteratively denoising using Equation (1), starting from $\boldsymbol{x}_T \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

10.2 U-net

U-nets [Ronneberger et al., 2015] are models used for image-to-image translation tasks. In the case of diffusion models, we have such a task, where we get $\boldsymbol{x}_t$ as input and want to predict $\boldsymbol{\epsilon}$. This framework is used to model $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}$. U-nets work by downsampling the input in stages and then upsampling back to the original space in the same stages. At every step of upsampling, the output of the corresponding downsampling step is concatenated to its input. In this way, we get low-level and high-level information.

10.3 Latent diffusion models

Stable diffusion [Rombach et al., 2022] performs diffusion modeling in the latent space of a pretrained VAE [Kingma, 2013]. During training, we thus first map the input image $x_0 \in \mathbb{R}^d$ to its latent encoding,

$$z_0 = \boldsymbol{\mathcal{E}}(x_0), \quad z_0 \in \mathbb{R}^{d'}, \quad d' \ll d.$$

Then, we use the same loss function as above, where we sample a random timestep and noise,

$$\ell = \left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}\left(\sqrt{1-\beta_t}z_0 + \sqrt{\beta_t}\boldsymbol{\epsilon}\right)\right\|^2, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d'}).$$

Then, during inference, we sample a noise vector in the latent space $z_T \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{d'})$ and denoise using Equation (1) to get $\tilde{z}_0$. Lastly, we decode the latent vector back into pixel space,

$$\tilde{\boldsymbol{x}}_0 = \boldsymbol{\mathcal{D}}(\tilde{z}_0).$$

This process requires a well-behaving latent space, so the regularization term that the VAE framework places on the latent space is very important.

10.4 Text embeddings

In order to perform text-to-image generation, we will need a continuous high-dimensional representation of the input text. For this, we use CLIP [Radford et al., 2021]. CLIP trains image and text transformer models to align text-image pairs. Because of this, they contain more semantic embeddings than other methods of training models to obtain text embeddings. Given a text input sequence of size T_c, the CLIP text model returns a sequence of embeddings,

$$\boldsymbol{C} \in \mathbb{R}^{T_c \times d_c}.$$

We denote this matrix by $\boldsymbol{C}$, because we use it for conditioning.

This sequence is contextualized, because CLIP makes use of self-attention.

10.5 Cross-attention

Now the question becomes how to condition a U-net on the input text sequence $\boldsymbol{C} \in \mathbb{R}^{T_c \times d_c}$. Stable diffusion [Rombach et al., 2022] does this by making use of cross-attention blocks, which it places at the end of every downsampling stage of the U-net. It first rearranges the output of the downsampling block into timesteps,

$$\boldsymbol{X} \in \mathbb{R}^{T \times d}.$$

Then, it computes queries from $\boldsymbol{X}$, and keys and values from $\boldsymbol{C}$,

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{X}\boldsymbol{W}_Q, \quad \boldsymbol{W}_Q \in \mathbb{R}^{d_k \times d} \\ \boldsymbol{K} &= \boldsymbol{C}\boldsymbol{W}_K, \quad \boldsymbol{W}_K \in \mathbb{R}^{d_k \times d_c} \\ \boldsymbol{V} &= \boldsymbol{C}\boldsymbol{W}_V, \quad \boldsymbol{W}_V \in \mathbb{R}^{d_v \times d_c}. \end{aligned}$$

It uses these to perform the attention mechanism with a residual connection,

$$\Xi = \boldsymbol{X} + \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V},$$

where Ξ denotes the conditioned representation of $\boldsymbol{X}$. Note that this architecture is agnostic to the type of the condition. As long as we can embed the conditioning variable, we can condition on it in this way—*e.g.*, we can additionally condition on images [Ye et al., 2023].

References

Brian DO Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

Pragya Sur and Emmanuel J Candès. A modern maximum-likelihood theory for high-dimensional logistic regression. *Proceedings of the National Academy of Sciences*, 116(29):14516–14525, 2019.

Victor Veitch, Alexander D'Amour, Steve Yadlowsky, and Jacob Eisenstein. Counterfactual invariance to spurious correlations in text classification. *Advances in neural information processing systems*, 34:16196–16208, 2021.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.