Machine Learning

Summary by Ari Feiglin (ari.feiglin@gmail.com)

Contents

1 Bayes Decision Theory

1.1 Formalizing the Theory

Let us imagine the following scenario: you, a computer science student, are preparing to leave your house for the first time in a while. Should you or should you not take your umbrella? If you take your umbrella and it doesn't rain then you've inconvenienced yourself, but if you don't and it rains then you'll end up getting wet. You look outside, see that clouds are grey, and decide to take your umbrella.

Here you are trying to decide between some possible actions, each with their own cost. These actions are based on the state of the world, of which you have some set of observations.

The problem can be formalized as follows, you have

(**1**) a set of world states: $\{\omega_i\}_{i\in I}$, which are disjoint and exhaustive: $\Omega = \bigcup_{i\in I} \omega_i$ (where Ω is the entire space).

(**2**) a set of observations: $S_n = \{x_1, \ldots, x_n\}$.

(**3**) a probabilistic model: conditionals $\mathbb{P}(S_n \mid \omega)$ and priors $\mathbb{P}(\omega)$.

(**4**) a set of possible actions $A = \{\alpha_1, \ldots, \alpha_k\}$.

(**5**) a set of cost functions $\Lambda = \{\lambda(\alpha_k \mid \omega_j)\}_{j,k}$.

So in our example above, we have two world states: raining and not raining, our observation is that the clouds are grey, we have some prior belief about the probabilities of each world state as well as the probability of observing our observation given a world state, our actions are to take and to not take an umbrella, each has an associated cost.

So suppose we've made an observation x, we'd like to compute the new probability of a world state ω given this observation. This can be done via Bayes's law:

$$\mathbb{P}(\omega \mid x) = \frac{\mathbb{P}(x \mid \omega)}{\mathbb{P}(x)} \cdot \mathbb{P}(\omega)$$

$\mathbb{P}(\omega \mid x)$ is called the *posterior*. But $\mathbb{P}(x)$ is not known, so how do we compute it? Using total probability:

$$\mathbb{P}(x) = \sum_{i\in I} \mathbb{P}(x \mid \omega_i)\, \mathbb{P}(\omega_i)$$

As we make more and more observations, we can employ Baye's law over and over to refine the posterior.

1.2 Minimizing Risk

Our goal in Bayes decision theory is to define a strategy $\alpha(S_n)$ which determines which action α to take so that it minimizes our expected costs, given observations S_n.

Definition 1.2.1

Given an action α, we define its **conditional risk** given an observation x to be

$$R(\alpha \mid x) = \sum_{i\in I} \lambda(\alpha \mid \omega_i)\, \mathbb{P}(\omega_i \mid x)$$

Where the posterior $\mathbb{P}(\omega_i \mid x)$ is computed as above using Bayes's law. If we define the random variable $\lambda(\alpha)$ to be $\lambda(\alpha \mid \omega)$ under the state ω, then this is just $\mathbb{E}[\lambda(\alpha) \mid x]$.

Example 1.2.2

Suppose our actions $\alpha_1, \ldots, \alpha_k$ are to guess the world state $\omega_1, \ldots, \omega_k$. The cost we pay is 1 if we are wrong and 0 if we are correct, meaning $\lambda(\alpha_k \mid \omega_j) = 1 - \delta_{kj}$ where $\delta_{kj} = 1$ when $k = j$ and 0 otherwise.

Then the conditional risk is

$$R(\alpha_k \mid x) = \sum_{j=1}^{k} \lambda(\alpha_k \mid \omega_j)\,\mathbb{P}(\omega_j \mid x) = \sum_{j \neq k} \mathbb{P}(\omega_j \mid x) = 1 - \mathbb{P}(\omega_k \mid x)$$

So we minimize the conditional risk when we take α_k such that the posterior $\mathbb{P}(\omega_k \mid x)$ is maximal.

Example 1.2.3

Suppose we have two world states ω_1, ω_2 and two actions α_1, α_2. Then our costs form a 2×2 matrix: $\lambda_{kj} = \lambda(\alpha_k \mid \omega_j)$. And by definition

$$R(\alpha_i \mid x) = \lambda_{i1}\,\mathbb{P}(\omega_1 \mid x) + \lambda_{i2}\,\mathbb{P}(\omega_2 \mid x)$$

We choose α_1 if $R(\alpha_1 \mid x) < R(\alpha_2 \mid x)$, which is equivalent to

$$(\lambda_{12} - \lambda_{22})\,\mathbb{P}(\omega_2 \mid x) < (\lambda_{21} - \lambda_{11})\,\mathbb{P}(\omega_1 \mid x)$$

which is equivalent to

$$\iff \frac{\mathbb{P}(\omega_1 \mid x)}{\mathbb{P}(\omega_2 \mid x)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \iff \frac{\mathbb{P}(x \mid \omega_1)}{\mathbb{P}(x \mid \omega_2)} > \frac{\mathbb{P}(\omega_2)}{\mathbb{P}(\omega_1)} \cdot \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}}$$

The left-hand side is called the **likelihood ratio** and the right-hand side is called the **decision boundary**.

If we have many observations $x_1, \ldots, x_n$ then this becomes

$$\underbrace{\frac{\mathbb{P}(x_1, \ldots, x_n \mid \omega_1)}{\mathbb{P}(x_1, \ldots, x_n \mid \omega_2)}}_{\text{Likelihood ratio}} > \underbrace{\frac{\mathbb{P}(\omega_2)}{\mathbb{P}(\omega_1)} \cdot \frac{\lambda_{22} - \lambda_{12}}{\lambda_{11} - \lambda_{21}}}_{\text{Decision boundary}} = \Theta$$

If the observations $x_1, \ldots, x_n$ are independent then this becomes

$$\frac{\prod_i \mathbb{P}(x_i \mid \omega_1)}{\prod_i \mathbb{P}(x_i \mid \omega_2)} > \Theta$$

taking the log of both sides gives

$$\sum_i \log \frac{\mathbb{P}(x_i \mid \omega_1)}{\mathbb{P}(x_i \mid \omega_2)} > \log \Theta = \Theta'$$

the left-hand side is called the **log-likelihood ratio**.

1.3 Parameter Estimation

Suppose we know the distribution of some random variable X up to some parameter θ, i.e. we know the function $\mathbb{P}(X \mid \theta)$. For example, X is the number of heads in n coin tosses where the coin has a bias of θ, then $X \mid \theta \sim \text{Bin}(n, \theta)$.

We use Bayes decision theory to estimate θ. Suppose we have a prior $\mathbb{P}(\theta)$, we use this to estimate θ. Our actions will be choosing some prediction of θ, $\hat{\theta}$. Define a cost function $\lambda(\hat{\theta}, \theta)$. Then given a sequence of observations $S_n = \{x_1, \ldots, x_n\}$, we want to minimize the expected cost

$$\mathbb{E}[\lambda(\hat{\theta}) \mid S_n] = \int \lambda(\hat{\theta}, \theta)\,\mathbb{P}(\theta \mid S_n)\,d\theta$$

We then define the *Bayes estimator* to be the prediction θ^* which minimizes this:

$$\theta^* = \text{argmin}_{\hat{\theta}}\,\mathbb{E}[\lambda(\hat{\theta}) \mid S_n]$$

To find θ^* we differentiate $\mathbb{E}[\lambda(\hat{\theta}) \mid S_n]$ and compare it with zero. Hand-waving away all the technical details because this is computer science, we can swap the order of differentiation and integration and so

$$\frac{d}{d\hat{\theta}} \mathbb{E}[\lambda(\hat{\theta}) \mid S_n] = \frac{d}{d\hat{\theta}} \int \lambda(\hat{\theta}, \theta) \, \mathbb{P}(\theta \mid S_n) \, d\theta = \int \frac{d}{d\hat{\theta}} \lambda(\hat{\theta}, \theta) \, \mathbb{P}(\theta \mid S_n) \, d\theta$$

Example 1.3.1 (Square Loss)

Suppose we use a **square loss** cost function: $\lambda(\hat{\theta}, \theta) = (\theta - \hat{\theta})^2$. Then the Bayes estimator is

$$\mathbb{E}[\lambda(\hat{\theta})] = \int \lambda(\hat{\theta}, \theta) \, \mathbb{P}(\theta \mid S_n) \, d\theta = \int (\theta - \hat{\theta})^2 \, \mathbb{P}(\theta \mid S_n) \, d\theta$$

Differentiating gives that we want

$$\int \theta \, \mathbb{P}(\theta \mid S_n) \, d\theta = \hat{\theta} \int \mathbb{P}(\theta \mid S_n) \, d\theta = \hat{\theta}$$

(since $\int \mathbb{P}(\theta \mid S_n) = 1$.) So we get that our estimator is

$$\hat{\theta}_{SE} = \int \theta \, \mathbb{P}(\theta \mid S_n) \, d\theta = \mathbb{E}[\theta \mid S_n]$$

Example 1.3.2 (Maximum Aposteriori)

Suppose we use a **zero-one** cost function: $\lambda(\hat{\theta}, \theta) = 1 - \delta_{\hat{\theta}\theta}$. We have already showed that to minimize the cost, we must maximize $\operatorname{argmax} \mathbb{P}(\theta \mid S_n)$. By Bayes, this is just $\operatorname{argmax} \mathbb{P}(S_n \mid \theta) \frac{\mathbb{P}(\theta)}{\mathbb{P}(S_n)} = \operatorname{argmax} \mathbb{P}(S_n \mid \theta) \, \mathbb{P}(\theta)$. Since the observations in S_n are independent, we see that this is then just equal to $\operatorname{argmax} \prod_i \mathbb{P}(x_i \mid \theta) \, \mathbb{P}(\theta)$. Finally we take the log, which is monotonic, to get that the estimator

$$\hat{\theta}_{MAP} = \operatorname{argmax} \sum_i \log \mathbb{P}(x_i \mid \theta) + \log \mathbb{P}(\theta)$$

as n grows, the last term becomes negligible and this just becomes the maximum log likelihood function.

Exercise 1.3.3

Compute the MAP estimator for $X \sim \text{Exp}(\theta)$, i.e. $f_X(x) = \theta \exp(-\theta x)$, with the prior $\mathbb{P}(\theta) = \exp(-\theta)$. We want to compute

$$\sum_i \log \mathbb{P}(x_i \mid \theta) + \log \mathbb{P}(\theta) = \sum_i \log\big(\theta \exp(-\theta x_i)\big) + \log \exp(-\theta) = n \log \theta - \theta \sum_i x_i - \theta$$

Differentiating with respect to θ gives

$$\frac{n}{\theta} - \sum_i x_i - 1 = 0 \implies \hat{\theta}_{MAP} = \frac{n}{\sum_i x_i + 1}$$

Here the prior behaves similarly to a sample, it can be viewed as a "pseudo-sample".

Example 1.3.4 (Maximum Likelihood)

We know that

$$\hat{\theta}_{MAP} = \operatorname{argmax} \sum_i \log \mathbb{P}(x_i \mid \theta) + \log \mathbb{P}(\theta)$$

A common approach is to approximate this by dropping the prior, giving

$$\hat{\theta}_{ML} = \operatorname{argmax} \sum_i \log \mathbb{P}(x_i \mid \theta)$$

Asymptotically this is efficient.

2 PAC Learning

2.1 Definitions

We now define *Probably Approximately Correct (PAC)* Learning.

Definition 2.1.1

Let $\mathcal{X}$ be the **input space**, the set of all possible examples or instances. The set of all **labels** or **target values** is $\mathcal{Y}$. For now, we restrict our view to be binary: $\mathcal{Y} = \{0, 1\}$. A **concept** is a map $c\colon \mathcal{X} \longrightarrow \mathcal{Y}$, or equivalently a subset of $\mathcal{X}$ (the set $\{x \in \mathcal{X} \mid c(x) = 1\}$). A **concept class** is a class of concepts which we would like to learn (approximate) and is denoted $\mathcal{C}$.

The idea of PAC learning is as follows: the learner considers a fixed set of concepts $\mathcal{H}$ called the *hypothesis set*, which may or may not coincide with $\mathcal{C}$. The learner then receives a sequence of samples $S = (x_1, \ldots, x_n)$ which are independent and distribute according to some distribution $\mathcal{D}$. The learner also receives labels $(c(x_1), \ldots, c(x_n))$ according to some concept $c \in \mathcal{C}$ which it is tasked with learning. Using this information the learner attempts to choose a hypothesis $h_S \in \mathcal{H}$ which minimizes the *generalization error* (or *risk*):

Definition 2.1.2

Given a hypothesis $h \in \mathcal{H}$, target concept $c \in \mathcal{C}$, and an underlying distribution $\mathcal{D}$, the **generalized error** (or **risk**) is:

$$R(h) = \mathbb{P}(h(x) \neq c(x) \mid x \sim \mathcal{D}) = \mathbb{E}[\mathbf{1}\{h(x) \neq c(x)\} \mid x \sim \mathcal{D}]$$

i.e. it is the probability that $h(x)$ differs from $c(x)$ when x is chosen randomly with a distribution of $\mathcal{D}$.

But the generalized error cannot be known to the learner, as it knows not the target concept nor the underlying distribution. So instead the learner minimizes the *empirical error* (or *risk*):

Definition 2.1.3

Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x_1, \ldots, x_n)$, define the **empirical error** (or **risk**) to be:

$$\widehat{R}_S(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{h(x_i) \neq c(x_i)\}$$

Notice that

$$\begin{aligned}
\mathbb{E}[\widehat{R}_S(h) \mid S \sim \mathcal{D}^n] &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[\mathbf{1}\{h(x_i) \neq c(x_i)\} \mid S \sim \mathcal{D}^n] \\
&= \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}[\mathbf{1}\{h(x) \neq c(x)\} \mid x \sim \mathcal{D}] = \frac{1}{n} \sum_{i=1}^{n} R(h) = R(h)
\end{aligned}$$

We now formally define what PAC learning is. Let n be a number such that the size of every $x \in \mathcal{X}$ can be represented in $O(n)$ space, for $c \in \mathcal{C}$ let *size* c be the maximal computational cost of c. We focus on algorithms $\mathcal{A}$ which take as input a sample S and return a hypothesis h_S.

Definition 2.1.4

A concept class $\mathcal{C}$ is **PAC-learnable** if there exists an algorithm $\mathcal{A}$ and a polynomial function $poly(\bullet, \bullet, \bullet, \bullet)$ such that for all $\varepsilon, \delta > 0$, distribution $\mathcal{D}$ on $\mathcal{X}$ and target concept $c \in \mathcal{C}$, for every sample size $n \geq poly(1/\varepsilon, 1/\delta, n, size\, c)$,

$$\mathbb{P}(R(h_S) \leq \varepsilon \mid S \sim \mathcal{D}^n) \geq 1 - \delta$$

If $\mathcal{A}$ runs in $poly(1/\varepsilon, 1/\delta, n, size\, c)$ time then $\mathcal{C}$ is **efficiently PAC-learnable**. An algorithm $\mathcal{A}$, if one exists, is called a **PAC-learning algorithm** for $\mathcal{C}$.

The intuition is as follows: a concept class $\mathcal{C}$ is PAC-learnable if there exists an algorithm $\mathcal{A}$ where given a sample size at least polynomial in $1/\varepsilon$ and $1/\delta$, it returns a hypothesis with an error bound by ε at least $1-\delta$ of the time. Note that if the running time is polynomial in $1/\varepsilon$ and $1/\delta$, then assuming the total input is read by the algorithm, the input must too be polynomial in $1/\varepsilon$ and $1/\delta$.

2.2 The Finite Case

Given a concept $c \in \mathcal{C}$ and a sample $S = (x_1, \ldots, x_n)$, call an hypothesis $h \in \mathcal{H}$ *consistent* if $h(x_i) = c(x_i)$ for all $1 \leq i \leq n$. Equivalently, $\widehat{R}_S(h) = 0$. We will assume that our hypotheses are consistent, and so we can always assume that our target concept is in $\mathcal{H}$.

Theorem 2.2.1

Let $\mathcal{H}$ be a finite set of hypotheses, and $\mathcal{A}$ an algorithm such that for any target concept $c \in \mathcal{H}$, $\mathcal{A}$ returns a consistent hypothesis h_S for any input sample S (where $S \sim \mathcal{D}^n$). Then for every every $\varepsilon, \delta > 0$, the inequality $\mathbb{P}(R(h_S) \leq \varepsilon \mid S \sim \mathcal{D}^n) \geq 1-\delta$ holds if

$$n \geq \frac{1}{\varepsilon} \cdot \log \frac{|\mathcal{H}|}{\delta}$$

Proof: let $\varepsilon > 0$, and define $\mathcal{H}_\varepsilon = \{h \in \mathcal{H} \mid R(h) > \varepsilon\}$. The probability that $h \in \mathcal{H}_\varepsilon$ is consistent is

$$\begin{aligned}\mathbb{P}(\widehat{R}_S(h) = 0) &= \mathbb{P}(h(x_1) = c(x_1), \ldots, h(x_n) = c(x_n) \mid x_i \sim \mathcal{D}) \\ &= \prod_{i=1}^{n} \mathbb{P}(h(x_i) = c(x_i) \mid x_i \sim \mathcal{D}) = \mathbb{P}(h(x) = c(x) \mid x \sim \mathcal{D})^n = \big(1 - \mathbb{P}(h(x) \neq c(x) \mid x \sim \mathcal{D})\big)^n\end{aligned}$$

this is since the samples are independent and distributively equal. Now recall that by definition, we have that $\mathbb{P}(h(x) \neq c(x) \mid x \sim \mathcal{D}) = R(h)$ which is greater than ε by definition, so

$$\mathbb{P}(\widehat{R}_S(h) = 0) \leq (1-\varepsilon)^n$$

Thus the probability that a "bad" hypothesis tricks us by being consistent is bound by $(1-\varepsilon)^n$. Now, we want to show that the probability that there exists a bad consistent hypothesis is bound by δ. This means that with probability at least $1-\delta$, there exist no bad consistent hypotheses and so h_S is necessarily a "good" hypothesis with $R(h_S) \leq \varepsilon$. The probability of there existing a consistent bad hypothesis is

$$\mathbb{P}(\exists h \in \mathcal{H}_\varepsilon.\ \widehat{R}_S(h) = 0) \leq \sum_{h \in \mathcal{H}_\varepsilon} \mathbb{P}(\widehat{R}_S(h) = 0) \leq \sum_{h \in \mathcal{H}_\varepsilon} (1-\varepsilon)^n \leq |\mathcal{H}|(1-\varepsilon)^n$$

We also know that $(1-\varepsilon)^n \leq e^{-n\varepsilon}$, so this probability is bound by $|\mathcal{H}|e^{-n\varepsilon}$. If $n \geq \frac{1}{\varepsilon} \cdot \log \frac{|\mathcal{H}|}{\delta}$ then this is bound by δ, as required. ■

Since $\frac{1}{\varepsilon} \cdot \log \frac{|\mathcal{H}|}{\delta}$ is bound by some polynomial in $\frac{1}{\varepsilon}, \frac{1}{\delta}$, this means that when $\mathcal{H}$ is finite a consistent algorithm $\mathcal{A}$ is PAC-learning. But what if our algorithm isn't consistent, i.e. the target concept is not in $\mathcal{H}$?

Theorem 2.2.2

For every $\varepsilon, \delta > 0$ if $n > \frac{1}{\varepsilon^2} \log \frac{2|\mathcal{H}|}{\delta}$ then

$$\mathbb{P}\left(R(h_S) - \min_{h \in \mathcal{H}} R(h) \leq \varepsilon\right) \geq 1 - \delta$$

Proof: similar to before, but using Hoeffding's inequality $\mathbb{P}(S - \mathbb{E}[S] \geq \varepsilon) \leq \exp(-2\varepsilon^2/n)$. ■

This is a generalization of the previous theorem, since if c is taken from $\mathcal{H}$ then $\min_{h \in \mathcal{H}} R(h) = 0$. Call a concept class $\mathcal{C}$ which satisfies this probability bound *Agonistically PAC-learnable*:

Definition 2.2.3

A concept class $\mathcal{C}$ is **agonistically PAC-learnable** if there exists an **agnostic PAC-learner** $\mathcal{A}$ such that for every $\varepsilon, \delta > 0$ distribution $\mathcal{D}$ and $n \geq poly(1/\varepsilon, 1/\delta, n, size\, c)$ for some $poly$,

$$\mathbb{P}\left(R(h_S) - \min_{h \in \mathcal{H}} R(h) \leq \varepsilon \right) \geq 1 - \delta$$

So assuming the hypothesis class is finite, every concept class is PAC-learnable in the agnostic sense.

2.3 The Infinite Case

Suppose now that $\mathcal{H}$ is infinite, and we don't necessarily have that the target concept is in $\mathcal{H}$. Define

$$h_S := \operatorname*{argmin}_{h \in \mathcal{H}} \widehat{R}_S(h), \qquad h_{\mathcal{D}} := \operatorname*{argmin}_{h \in \mathcal{H}} R_{\mathcal{D}}(h)$$

Then we have

Lemma 2.3.1

$$R_{\mathcal{D}}(h_S) - R_{\mathcal{D}}(h_{\mathcal{D}}) \leq 2 \sup_{h \in \mathcal{H}} |\widehat{R}_S(h) - R_{\mathcal{D}}(h)|$$

Proof: let $\varepsilon = \sup_{h \in \mathcal{H}} |\widehat{R}_S(h) - R_{\mathcal{D}}(h)|$, then

$$\begin{aligned} R_{\mathcal{D}}(h_S) &\leq \widehat{R}_S(h_S) + \varepsilon && \text{since } h_S \in \mathcal{H} \\ &\leq \widehat{R}_S(h_{\mathcal{D}}) + \varepsilon && \text{since } h_S \text{ minimizes } \widehat{R}_S \\ &\leq R_{\mathcal{D}}(h_{\mathcal{D}}) + 2\varepsilon && \text{since } h_{\mathcal{D}} \in \mathcal{H} \end{aligned}$$

■

Definition 2.3.2

A hypothesis class $\mathcal{H}$ **shatters** a finite set $C = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ if for any labels $y_1, \ldots, y_n \in \{0, 1\}$ there exists an hypothesis $h \in \mathcal{H}$ such that $h(x_i) = y_i$ for all i.

In other words, $\mathcal{H}$ shatters C if for every function $f : C \longrightarrow \mathcal{Y}$, there exists an hypothesis h such that $h\big|_C = f$. So if we define $\mathcal{H}_C = \left\{ h\big|_C \;\middle|\; h \in \mathcal{H} \right\}$, then $\mathcal{H}$ shatters C if and only if $|\mathcal{H}_C| = 2^{|C|}$.

Definition 2.3.3

The **VC dimension** of an hypothesis class $\mathcal{H}$, denoted $\text{VCdim}(\mathcal{H})$, is the size of the largest set $C \subseteq \mathcal{X}$ shattered by $\mathcal{H}$.

Theorem 2.3.4

Let $\mathcal{H}$ be an hypothesis class with VC dimension $d < \infty$. Then there exists a constant $c > 0$ such that for every $\varepsilon, \delta \in (0, 1)$ there is a polynomial such that for every $n \geq poly(1/\varepsilon, 1/\delta)$ for which

$$\mathbb{P}\left(\sup_{h \in \mathcal{H}} |\widehat{R}_S(h) - R_{\mathcal{D}}(h)| < c\sqrt{\frac{d}{n}} \;\middle|\; S \sim \mathcal{D}^n \right) \geq 1 - \delta$$

The proof of this is beyond the scope of the class.

3 Linear Regression

3.1 Linear Regression

Suppose our input space $\mathcal{X}$ is a subset of $\mathbb{R}^d$ for some d, and $\mathcal{Y}$ is $\mathbb{R}$. Given a sample $\{(\vec{x}_i, y_i)\}_{1\le i\le n}$, we would like to find the best linear approximation $h\colon \mathbb{R}^d \longrightarrow \mathbb{R}$ which approximates the relation between the inputs and their labels. Thus the hypothesis class is then the set of all linear functions

$$\mathcal{H} = \left\{ f(\vec{x}) = \vec{x}^\top \vec{w} + b \;\middle|\; \vec{w} \in \mathbb{R}^d, b \in \mathbb{R} \right\}$$

For a hypothesis f, let us denote $\hat{y}_i = f(x_i) = \vec{x}_i^\top \vec{w} + b$, notice that

$$\begin{pmatrix} \text{---} & \vec{x}_1 & \text{---} & 1 \\ & \vdots & & \vdots \\ \text{---} & \vec{x}_n & \text{---} & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_d \\ b \end{pmatrix} = \begin{pmatrix} \vec{x}_1^\top \vec{w} + b \\ \vdots \\ \vec{x}_n^\top \vec{w} + b \end{pmatrix}$$

To simplify notation, we can just assume that the final coordinate of each $\vec{x}_i$ is 1 and so we can view each hypothesis f as its vector $\vec{w}$ (where the last coefficient is b). Thus

$$X\vec{w} = \hat{y}$$

Where X is the matrix whose rows are $\vec{x}_i$.

We want to minimize the difference between $\hat{y}$ and $\vec{y}$. We can do this by taking the mean square error (MSE): $\frac{1}{n}\|\hat{y} - \vec{y}\|^2 = \frac{1}{n}\|X\vec{w} - \vec{y}\|^2$. So we define

$$\mathrm{MSE}(\vec{w}) = \frac{1}{n}\|X\vec{w} - \vec{y}\|^2 = \frac{1}{n}(X\vec{w} - \vec{y})^\top (X\vec{w} - \vec{y})$$

We take the gradient and compare with zero:

$$\nabla \mathrm{MSE}(\vec{w}) = \frac{2}{n} X^\top (X\vec{w} - \vec{y})$$

comparing with zero gives

$$\nabla \mathrm{MSE}(\vec{w}) = 0 \iff X^\top X\vec{w} - X^\top \vec{y} = 0 \iff \vec{w}_{\min} = (X^\top X)^{-1} X^\top \vec{y}$$

This must be a minimum since MSE is quadratic in $\vec{w}$.

So now if we have new data X_{new}, our estimator for the new values will be

$$\hat{y}_{\text{new}} = X_{\text{new}} \vec{w}_{\min} = X_{\text{new}} (X^\top X)^{-1} X^\top \vec{y}$$

Furthermore, notice that the error $\vec{y} - \hat{y}$ is orthogonal to $\hat{y}$:

$$\begin{aligned} (\vec{y} - \hat{y})^\top \hat{y} = (\vec{y} - X\vec{w})^\top \hat{y} &= \vec{y}^\top \hat{y} - \vec{w}^\top X^\top X\vec{w} \\ &= \vec{y}^\top X\vec{w} - \vec{w}^\top X^\top X (X^\top X)^{-1} X^\top \vec{y} \\ &= \vec{y}^\top X\vec{w} - \vec{w}^\top X^\top \vec{y} \\ &= \vec{y}^\top X\vec{w} - (X\vec{w})^\top \vec{y} = 0 \end{aligned}$$

3.2 Polynomial Fitting

Suppose we have n points $\{(x_i, y_i)\}_{1\le i\le n}$ which we would like to approximate using a k-degree polynomial $\hat{y} = p(x) = w_0 x^0 + \cdots + w_k x^k$. We can do so using linear regression: define

$$X = \begin{pmatrix} x_1^0 & x_1^1 & \cdots & x_1^k \\ \vdots & \vdots & & \vdots \\ x_n^0 & x_n^1 & \cdots & x_n^k \end{pmatrix}$$

Then doing linear regression gets us the best vector $\vec{w}$ such that

$$X\vec{w} = \begin{pmatrix} \sum_{i=0}^{k} w_i x_1^i \\ \vdots \\ \sum_{i=0}^{k} w_i x_n^i \end{pmatrix} = \begin{pmatrix} p(x_1) \\ \vdots \\ p(x_n) \end{pmatrix}$$

is closest to $\vec{y}$, as required.

3.3 Ridge Regression

What if $X^\top X$ is not invertible? Which can happen if we have too few samples, i.e. $n < d$. Notice that $X^\top X$ is positive semi-definite: $v^\top X^\top X v = (Xv)^\top (Xv) \geq 0$, and thus all its eigenvalues are nonnegative. Then for any $\lambda > 0$, $(X^\top X + \lambda I)v = \mu v$ if and only if $X^\top X v = (\mu - \lambda)v$ which must mean that $\mu - \lambda \geq 0$ and in particular $\mu > 0$. So all of $X^\top X + \lambda I$'s eigenvalues are positive, and in particular non-zero, meaning it is invertible.

So using this knowledge, let us define the *ridge estimator* to be

$$\widehat{w}_{\text{ridge}} = \underset{\vec{w}}{\text{argmin}} \|X\vec{w} - \vec{y}\|^2 + \lambda \|\vec{w}\|^2$$

Taking the gradient and equating to zero gives that

$$\widehat{w}_{\text{ridge}} = (X^\top X + \lambda I)^{-1} X^\top y$$

which, as explained above, exists no matter what, for any $\lambda > 0$.