

156 – Machine Learning

University of California, Los Angeles

Duc Vu

Summer 2021

This is math 156 – Machine Learning, an introductory course on mathematical models for pattern recognition and machine learning. It's instructed by Professor Zosso, and we meet weekly on MWTh from 9:00 am to 10:50 am. The textbook used for the class is *Pattern Recognition and Machine Learning* by Bishop. You can find the other course notes through my [blog site](#). Any error appeared in this note is my responsibility and please [email](#) me if you happen to notice it.

Contents

1	Lec 1: Jun 21, 2021	2
1.1	Introduction & Probability Review	2
1.2	Gaussian Distribution	4
2	Lec 2: Jun 23, 2021	6
2.1	Gaussian Distribution (Cont'd)	6
2.2	Non-parametric Probability Density Function (Estimation)	7
3	Lec 3: Jun 24, 2021	9
3.1	Principal Component Analysis	9
3.2	High-Dimensional PCA	11
3.3	Probabilistic PCA	11
3.4	Maximum Likelihood PCA	13
4	Lec 4: Jun 30, 2021	14
4.1	Kernel PCA	14
4.2	Linear Basis Function Models	16
5	Lec 5: Jul 1, 2021	19
5.1	Overfitting	19
5.2	Regularized Least Squares	19
5.3	Bayesian Linear Regression	20

§1 | Lec 1: Jun 21, 2021

§1.1 Introduction & Probability Review

According to Wikipedia, **Machine Learning** is a scientific discipline that deals with the construction and study of algorithms that can learn from data.

$$\text{Input(data)} \rightarrow \boxed{\text{Model}} \rightarrow \text{Output(Predictions/Decisions)}$$

From §1.2 of the book, let's review a bit on probability.

- Discrete random variable X , value $\{x_i\}$

$$\text{prob}(X = x_i) = p(x_i) = \frac{n_i}{N}$$

and

$$\sum_i \text{prob}(X = x_i) = \sum_i p(x_i)$$

For multiple random variables, $X, Y \in \{x_i\} \times \{y_j\}$

1. $\text{prob}(X = x_i, Y = y_j) = \frac{n_{ij}}{N} = p(x_i, y_j)$ – joint probability
2. $\text{prob}(X = x_i) = \sum_j \text{prob}(X = x_i, Y = y_j)$ – marginal probability
3. $\text{prob}(X = x_i | Y = y_j)$ = conditional

$$\underbrace{p(x_i | y_j)}_{\text{conditional}} \cdot \underbrace{p(y_j)}_{\text{marginal}} = \underbrace{p(x_i, y_j)}_{\text{joint}}$$

\Rightarrow product rule

Bayes' Rule:

$$p(y|x) = \frac{p(x|y) \cdot p(y)}{p(x)}$$

- Continuous random variable $X \in \mathbb{R}$

$$\text{prob}(X = x_i) = 0 \text{ in general}$$

So we consider probability densities instead where

$$p(x) \geq 0$$

s.t. $p(x)$ can be greater than 1. In addition,

$$\int_{-\infty}^{\infty} p(x) = 1$$

Within a neighborhood $a \leq b$, we have

$$\text{prob}(a \leq x \leq b) = \int_a^b p(x) dx$$

Sum rule:

$$\int \underbrace{p(x, y)}_{\text{joint pdf}} dy = \underbrace{p(x)}_{\text{marginal pdf}}$$

Product rule:

$$p(x, y) = p(y|x)p(x) = p(x|y)p(y)$$

Bayes' Rule:

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

Expectations & Covariances

Expectations:

Definition 1.1 — Expectation is defined as

$$\mathbb{E}[f] := \sum_i p(x_i) f(x_i)$$

$$\text{or} := \int_{\mathbb{R}} p(x) f(x) dx$$

“Average value of a function $f : \mathbb{R} \rightarrow \mathbb{R}$ under a probability distribution $p(x)$ ”

In practice, we need to estimate p from data.

$$\text{Sampling Approximation: } \mathbb{E}[f] \approx \frac{1}{N} \sum_{n=1}^N f(x_n)$$

Definition 1.2 — Marginal expectation is defined as

$$\mathbb{E}_x[f](y) := \sum_x p(x) f(x, y)$$

Conditional expectation:

$$\mathbb{E}_x[f|y] := \sum_x p(x|y) f(x)$$

Covariances:

Definition 1.3 — Variance is defined as

$$\text{var}[f] := \mathbb{E} \left[(f(x) - \mathbb{E}[f])^2 \right]$$

$$= \mathbb{E}[f^2] - \mathbb{E}[f]^2$$

Covariance (random variables) is defined as

$$\text{cov}[x, y] := \mathbb{E}[(x - \mathbb{E}[x])(y - \mathbb{E}[y])]$$

$$= \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]$$

For vectors $\vec{x}, \vec{y} \in \mathbb{R}^D$, the covariance matrix is

$$\mathbb{E} \left[(\vec{x} - \mathbb{E}[\vec{x}]) (\vec{y} - \mathbb{E}[\vec{y}])^\top \right]$$

Question 1.1. How does this fit in within the context of machine learning?

In machine learning, there are usually two approaches to find the “optimal prediction”

- Frequentist approach: maximize likelihood

$$\max_w p(D|w)$$

- Bayesian approach: maximize posterior

$$\text{posterior through Bayes': } p(w|D) = \frac{p(D|w) \cdot p(w)}{p(D)}$$

s.t.

$$\max_w p(w|D) \sim p(D|w) \cdot p(w)$$

where D represents data, and w is parameters.

Gaussian noise model:

$$p(t_n|x_n, w, \beta) = N\left(t_n|y(x_n, w), \frac{1}{\beta}\right)$$

Given training data $\{(x, t)\}$, we can determine optimal parameters w, β by

1. Frequentist: maximize likelihood

$$p(t|x, w, \beta) \stackrel{\text{i.i.d.}}{=} \prod_{n=1}^N N(t_n|y(x_n, w), \beta^{-1})$$

2. include a prior: $p(w|\alpha) = N(w|0, \alpha^{-1})$

$$\implies \text{posterior: } p(w|x, t, \alpha, \beta) \propto p(t|x, w, \beta) p(w|\alpha)$$

Then, we can estimate

$$\min_w \left\{ \frac{\beta}{2} \sum_{n=1}^N (y(x_n, w) - t_n)^2 + \frac{\alpha}{2} w^\top w \right\}$$

3. Fully Bayesian: not just point estimates \implies predictive distribution

$$p(t_i|x_i, x, t) = \int \underbrace{p(t_i|x_i, w)}_{\text{model}} \underbrace{p(w|x, t)}_{\text{posterior}} dw$$

§1.2 Gaussian Distribution

Definition 1.4 (Gaussian Distribution) — The 1-D Gaussian distribution is defined as

$$N(x|\mu, \sigma^2) := \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean and σ^2 is the variance.

For D -dimensional,

$$N(\vec{x}|\vec{\mu}, \Sigma) := \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{x}-\vec{\mu})^\top \Sigma^{-1}(\vec{x}-\vec{\mu})}$$

where Σ is the covariance matrix and $|\Sigma|$ is the determinant of Σ .

Consider $x \in \mathbb{R}^D$, $x \sim N$. Assume

$$x = \begin{bmatrix} x_a \\ x_b \end{bmatrix}$$

where x_a is unknown and x_b is given component.

$$x \sim N\left(\begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}\right)$$

Note that

$$\Sigma = \Sigma^\top$$

Also, we define the precision matrix Λ as

$$\begin{aligned}\Lambda &:= \Sigma^{-1} \\ &= \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}\end{aligned}$$

Unfortunately, $\Lambda_{aa} \neq \Sigma_{aa}^{-1}$ and similar result applies for b .

Question 1.2. What can we say about $p(x_a|x_b)$?

Use product rule:

$$p(x_a|x_b) \cdot p(x_b) = p(x_a, x_b)$$

where $p(x_b)$ is a constant w.r.t. x_a

$$\implies p(x_a|x_b) \propto p(x_a, x_b)$$

Let's look at quadratic form in exponential only.

$$\begin{aligned}-\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu) &= -\frac{1}{2}(x_a - \mu_a)^\top \Lambda_{aa}(x_a - \mu_a) - \frac{1}{2}(x_a - \mu_a)^\top \Lambda_{ab}(x_b - \mu_b) \\ &\quad - \frac{1}{2}(x_b - \mu_b)^\top \Lambda_{ba}(x_a - \mu_a) - \frac{1}{2}(x_b - \mu_b)^\top \Lambda_{bb}(x_b - \mu_b)\end{aligned}$$

Also,

$$\text{other side} = -\frac{1}{2}x_a^\top \Sigma_{a|b}^{-1}x_a + x_a^\top \Sigma_{a|b}^{-1}\mu_{a|b} + \text{const}$$

- Quadratic terms need to match

$$\begin{aligned}-\frac{1}{2}x_a^\top \Sigma_{a|b}^{-1}x_a &= -\frac{1}{2}x_a^\top \Lambda_{aa}x_a \\ \implies \Sigma_{a|b}^{-1} &= \Lambda_{aa}\end{aligned}$$

- Linear terms in x_a

$$\begin{aligned}x_a^\top \Sigma_{a|b}^{-1}\mu_{a|b} &= x_a^\top \Lambda_{aa}\mu_{a|b} \\ \Lambda_{aa}\mu_{a|b} &= \Lambda_{aa}\mu_a - \Lambda_{ab}(x_b - \mu_b) \\ \implies \mu_{a|b} &= \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b)\end{aligned}$$

Note that

$$\begin{aligned}\Lambda_{aa} &= (\Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba})^{-1} \\ \Lambda_{ab} &= -\Lambda_{aa}\Sigma_{ab}\Sigma_{bb}^{-1}\end{aligned}$$

Thus,

$$\begin{cases} \mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b) \\ \Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \end{cases}$$

§ 2 | Lec 2: Jun 23, 2021

§ 2.1 Gaussian Distribution (Cont'd)

Let's start with a set of observations:

$$X = \{\vec{x}_1, \dots, \vec{x}_N\} \quad N \text{ data points where each } \vec{x}_n \in \mathbb{R}^D$$

and each $\vec{x}_n \sim N(\mu, \Sigma)$. As usual, there are two approach to this.

- Maximum likelihood: given the data, what μ, Σ are most probable/likely?

$$\max_{\mu, \Sigma} p(X|\mu, \Sigma)$$

Model assumption: \vec{x}_n are i.i.d (independently, identically distributed). From i.i.d, we have

$$\begin{aligned} p(X|\mu, \Sigma) &= \prod_{n=1}^N p(\vec{x}_n|\mu, \Sigma) \\ &= \prod_{n=1}^N N(\vec{x}_n|\mu, \Sigma) \end{aligned}$$

This is tricky to do, so let's minimize the negative log likelihood

$$\begin{aligned} \min_{\mu, \Sigma} -\ln p(X|\mu, \Sigma) &= -\ln \prod_{n=1}^N \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x_n - \mu)^\top \Sigma^{-1}(x_n - \mu)} \\ &= \cancel{-N \ln \frac{1}{(2\pi)^{\frac{D}{2}}}} - N \ln \frac{1}{|\Sigma|^{\frac{1}{2}}} + \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^\top \Sigma^{-1}(x_n - \mu) \\ &= \frac{N}{2} \ln |\Sigma| + \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^\top \Sigma^{-1}(x_n - \mu) + C \end{aligned}$$

As the domain is unbounded (unconstrained optimization problem) and objective function is convex, so to find optimal μ , we set $\frac{d}{d\mu} = 0$. Then

$$\begin{aligned} \frac{1}{2} \sum_{n=1}^N \Sigma^{-1}(x_n - \mu) &= 0 \\ \sum_{n=1}^N \Sigma^{-1}x_n &= N\Sigma^{-1}\mu \\ \implies \mu &= \frac{1}{N} \sum_{n=1}^N x_n \end{aligned}$$

- Maximum a posteriori (MAP)

$$\max_{\mu} p(\mu, \Sigma|X) \stackrel{\text{Bayes'}}{\implies} \max_{\mu} p(X|\mu, \Sigma) \cdot p(\mu)$$

e.g., $p(\mu|\mu_0, \Sigma_0) = N(\mu|\mu_0, \Sigma_0)$. We have

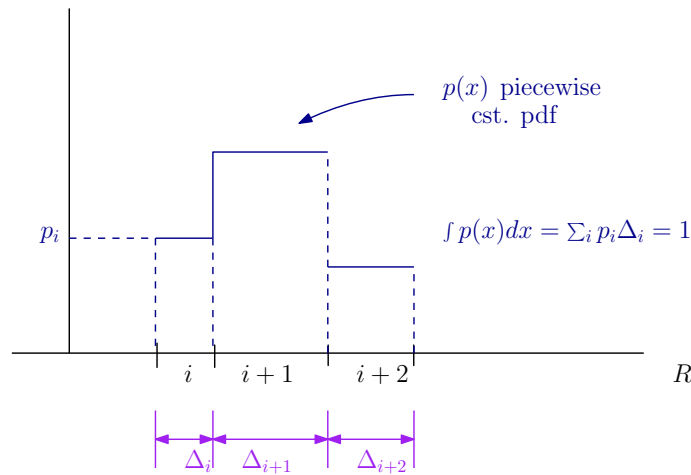
$$\begin{aligned}
 & -\ln p(X|\mu, \Sigma) \cdot p(\mu|\mu_0, \Sigma_0) \\
 & \min_{\mu} \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^\top \Sigma^{-1} (x_n - \mu) + \frac{1}{2} (\mu - \mu_0)^\top \Sigma_0^{-1} (\mu - \mu_0) \\
 & \frac{d}{d\mu} = 0 : \sum_{n=1}^N \Sigma^{-1} (x_n - \mu) + \Sigma_0^{-1} (\mu - \mu_0) = 0 \\
 & \Rightarrow \mu_{\text{MAP}} = (N\Sigma^{-1} + \Sigma_0^{-1})^{-1} (N\Sigma^{-1}\bar{x} + \Sigma_0^{-1}\mu_0)
 \end{aligned}$$

§2.2 Non-parametric Probability Density Function (Estimation)

Let's consider the following

- Histograms
- partition domain of x into distinct bins of width Δ_i
- count number of observations n_i of x falling into bin i
- divide by N, Δ_i to get a pdf.

$$p_i = \frac{n_i}{N\Delta_i} \text{ is density over bin } i$$



We often partition the domain uniformly, i.e., $\Delta_i = \Delta$

Consider a region $R \subseteq \mathbb{R}^D$. The probability of a randomly chosen point will fall into R (according to pdf of $p(x)$) is

$$p = \int_R p(x) dx$$

Collect N samples; a fraction K of which will fall into R . So $K \sim \text{Binomial}(N, p)$

$$\begin{aligned}
 \mathbb{E} \left[\frac{K}{N} \right] &= p \\
 \text{var} \left[\frac{K}{N} \right] &= \frac{p(1-p)}{N} \\
 \text{var} \left[\frac{K}{N} \right] &\xrightarrow{N \rightarrow \infty} 0
 \end{aligned}$$

refer to fig 2.24 in textbook for other cases

For large N , $\frac{K}{N} \approx P \implies K \approx N \cdot P$. Also, we want R big so that there are plenty of points in there. On the other hand, we want R small s.t. $p(x) \sim \text{constant over } R$ where $p = p(x)V$ in which V is the volume of R . Thus,

$$p(x) = \frac{K}{NV}$$

For histogram: we fix V and measure $\frac{K}{N}$. For the kernel, it's essentially the same but bin locations are not predefined.

Kernel Approach: If we want to know $p(x)$ at arbitrary x , we put a bin of predefined size around x then count $\frac{K}{N}$ for that bin.

Pick a smooth kernel, e.g., the Gaussian

$$p_h(x) := \frac{1}{N} \sum_{n=1}^N \frac{1}{(2\pi h^2)^{\frac{D}{2}}} e^{-\frac{\|x-x_n\|_2^2}{2h^2}}$$

where h is standard deviation of Gaussian. Recall from 131BH that this is a convolution.

$$(f * g)(x) := \int f(y)g(x-y) dy$$

So $k * \sum \delta(-x_n)$. More general,

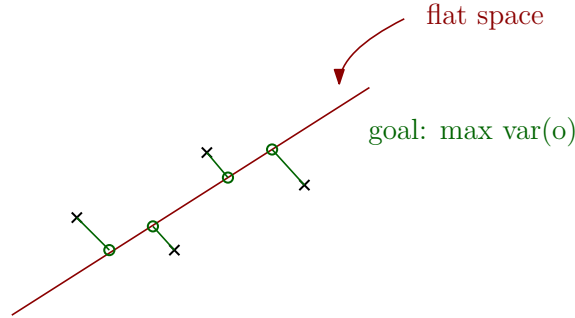
$$\begin{cases} k(u) \geq 0 \\ \int k(u) du = 1 \end{cases}$$

is sufficient criteria to be a kernel for kernel density estimation (KDE).

§3 | Lec 3: Jun 24, 2021

§3.1 Principal Component Analysis

Maximum Variance Formulation: consider $\{x_n\}$, $n = 1, \dots, N$, $x_n \in \mathbb{R}^D$. The goal is to project x onto a flat space with dimension $M \ll D$ while maximizing the variance of the projected data.



Let's start with $M = 1$ (a line) defined by a single vector $\vec{u} \in \mathbb{R}^D$ with unit norm, i.e.,

$$u_1^\top u_1 = \langle u_1, u_1 \rangle = \|u_1\|_2^2 = 1$$

Define: $\bar{x} = \frac{1}{N} \sum_{n=1}^N x_n$. Note that the variance before projection is

$$\text{var} = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})^2$$

and after projection is

$$\text{var} = \frac{1}{N} \sum_{n=1}^N (u_1^\top x_n - u_1^\top \bar{x})^2 = u_1^\top S u_1$$

with

$$\begin{aligned} S &= \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})(x_n - \bar{x})^\top \\ &= \text{cov}(x) \end{aligned}$$

Our optimization goal is

$$\max_{u_1} u_1^\top S u_1 \quad \text{s.t.} \quad u_1^\top u_1 = 1$$

This is a constrained optimization problem – let's introduce Lagrange multipliers for constraint:

$$\max_{u_1, \lambda_1} \left\{ \underbrace{u_1^\top S u_1 + \lambda_1 (1 - u_1^\top u_1)}_{=: L[u_1, \lambda_1]} \right\}$$

We have

$$\begin{aligned} \frac{\partial L}{\partial u_1} : 2S u_1 - 2\lambda_1 u_1 &= 0 \\ S u_1 &= \lambda_1 u_1 \end{aligned}$$

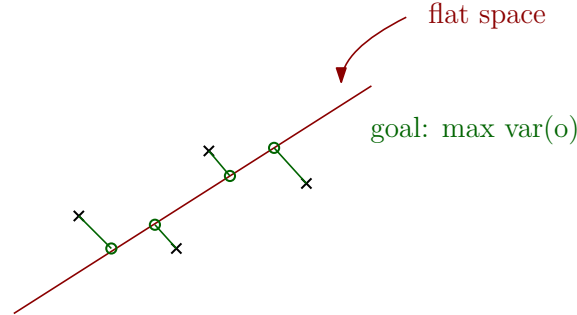
So, the eigen-problem: (λ_1, u_1) is eigenpair of S .

$$\text{var} = u_1^\top S u_1 = u_1^\top (\lambda_1 u_1) = \lambda_1 u_1^\top u_1 = \lambda_1$$

\Rightarrow we need to pick the dominant eigenpair of S . So if we want to project onto a flat with $M > 1$, we can simply pick u_1, \dots, u_n as the M leading eigenvectors of S where all u_i are orthogonal and

$$\text{var} = \sum_{i=1}^N \lambda_i$$

Minimum Error Formulation:



Goal: introduce as little distortion as possible.

Consider: $\{u_i\}, i = 1, \dots, D$ orthonormal basis of \mathbb{R}^D

$$\Rightarrow u_i^\top u_j = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases}$$

Then each data point x_n has unique expansion in that basis

$$x_n = \sum_{i=1}^D \alpha_{ni} u_i \quad \alpha_{ni} \in \mathbb{R}$$

where

$$\begin{aligned} x_n^\top u_j &= u_j^\top x_n = u_j^\top \sum_{i=1}^D \alpha_{ni} u_i \\ &= \sum_{i=1}^D \alpha_{ni} u_j^\top u_i = \alpha_{nj} \\ \Rightarrow x_n &= \sum_{i=1}^D (x_n^\top u_i) u_i \end{aligned}$$

As we project to a flat, we need only the first M terms

$$\tilde{x}_n = \sum_{i=1}^M z_{ni} u_i + \sum_{i=M+1}^D b_i u_i$$

Now, we choose z_{ni}, u_i, b_i so as to minimize the distortion.

$$J = \frac{1}{N} \sum_{n=1}^N \|x_n - \tilde{x}_n\|_2^2$$

The results we should've obtained are

1. $z_{ni} = x_n^\top u_i, i = 1, \dots, M$
2. $b_i = \bar{x}^\top u_i, i = M + 1, \dots, D$

We can substitute these into the expression of \tilde{x}_n as follow

$$\begin{aligned}\tilde{x}_n &= \sum_{i=1}^M (x_n^\top u_i) u_i + \sum_{i=M+1}^D (\bar{x}^\top u_i) u_i \\ x_n - \tilde{x}_n &= \sum_{i=M+1}^D (x_n^\top u_i - \bar{x}^\top u_i) u_i\end{aligned}$$

In addition, the error term can be written as

$$J = \frac{1}{N} \sum_{n=1}^N \sum_{i=M+1}^D (x_n^\top u_i - \bar{x}^\top u_i)^2 = \sum_{i=M+1}^D u_i^\top S u_i$$

So the problem now becomes

$$\min_{u_i, i=M+1, \dots, D} \sum_{i=M+1}^D u_i^\top S u_i \quad \text{s.t.} \quad u_i^\top u_i = 1$$

Analogous to the case of maximum variance, we “throw away” the weakest eigenpairs of S .

§3.2 High-Dimensional PCA

Assume we have N data points with D dimensions and $\bar{x} = 0$. Then, $S = \frac{1}{N} x^\top x$

$$X = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{bmatrix}$$

where each x_n is a row of X . As $\bar{x} = 0$, rows sum up to 0.

Let's examine the eigenvalues of $x^\top x$ v.s. eigenvalues of xx^\top .

$$\begin{aligned}\frac{1}{N} x^\top x u_i &= \lambda_i u_i \\ \frac{1}{N} x x^\top (x u_i) &= \lambda_i \underbrace{(x u_i)}_{v_i} \\ \frac{1}{N} x x^\top v_i &= \lambda_i v_i\end{aligned}$$

§3.3 Probabilistic PCA

Consider $x_n \in \mathbb{R}^D$ where

$$x_n = Wz + \mu + \varepsilon$$

where $z \in \mathbb{R}^M$ is latent variable and μ is mean and ε is noise & $\varepsilon \sim N(0, \sigma^2 I)$; z is the coordinates within the lower-dim flat, and W is the basis of the flat. The probabilistic formulation is

$$p(z) = N(z|0, I)$$

\Rightarrow latent variable \sim zero-mean, unit variance Gaussian. The conditional distribution $x|z$ is again Gaussian

$$p(x|z) = N\left(x \mid \underbrace{Wz + \mu}_{\text{nozzle location}}, \underbrace{\sigma^2 I}_{\text{spray size}}\right)$$

Resulting point cloud is governed by predictive density $p(x)$.

$$p(x) = \int \underbrace{p(x|z) \cdot p(z)}_{p(x,z)} dz$$

Claim 3.1. $p(x)$ is Gaussian, too.

$$\begin{aligned} p(x) &= N(x|\mu, C) \\ C &= WW^\top + \sigma^2 I \in \mathbb{R}^{D \times D} \end{aligned}$$

Proof. Sufficient statistics

$$\begin{aligned} \mathbb{E} &= \mathbb{E}[Wz + \mu + \varepsilon] \\ &= \mathbb{E}[Wz] + \mu + \mathbb{E}[\varepsilon] \\ &= W\mathbb{E}[z] + \mu = \mu \end{aligned}$$

For the covariance,

$$\begin{aligned} \text{cov}[x] &= \mathbb{E}[(x - \mu)(x - \mu)^\top] \\ &= \mathbb{E}[(Wz + \mu + \varepsilon - \mu)(Wz + \mu + \varepsilon - \mu)^\top] \\ &= \mathbb{E}[(Wz + \varepsilon)(Wz + \varepsilon)^\top] \\ &= \mathbb{E}[(Wz(Wz)^\top) + Wz\varepsilon^\top + \varepsilon(Wz)^\top + \varepsilon\varepsilon^\top] \\ &= \mathbb{E}[Wzz^\top W^\top] + \mathbb{E}[Wz\varepsilon^\top] + \mathbb{E}[\varepsilon z^\top W^\top] + \mathbb{E}[\varepsilon\varepsilon^\top] \\ &= W\mathbb{E}[zz^\top]W^\top + \cancel{W\mathbb{E}[z\varepsilon^\top]} + \cancel{\mathbb{E}[\varepsilon z^\top]W^\top} + \mathbb{E}[\varepsilon\varepsilon^\top] \\ &= WW^\top + \sigma^2 I \end{aligned}$$

Remark 3.1. $\mathbb{E}[z\varepsilon^\top] = 0 = \mathbb{E}[\varepsilon z^\top]$ because z is independent from ε .

□

Note: Redundancy w.r.t. rotations in latent space (lack of uniqueness). Let $\tilde{W} = WQ$ where Q is orthonormal.

$$\begin{aligned} C &= \tilde{W}\tilde{W}^\top + \sigma^2 I \\ &= W \underbrace{QQ^\top}_I W^\top + \sigma^2 I \\ &= WW^\top + \sigma^2 I \end{aligned}$$

To evaluate $p(x) = N(x|\mu, C)$. We need C^{-1} .

$$C^{-1} = \sigma^{-2}I - \sigma^2 W M^{-1} W^\top$$

for $M = W^\top W + \sigma^2 I \in \mathbb{R}^{M \times M}$.

§3.4 Maximum Likelihood PCA

We need to learn W, μ, σ^2 from given data. By i.i.d,

$$\begin{aligned}
 p(X|W, \mu, \sigma^2) &= \prod_{n=1}^N p(x_n|W, \mu, \sigma^2) \\
 \implies \ln p(X|W, \mu, \sigma^2) &= \sum_{n=1}^N \ln N(x_n|\mu, WW^\top + \sigma^2 I) \\
 &= -\frac{ND}{2} \ln(2\pi) - \frac{N}{2} \ln |C| - \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^\top C^{-1} (x_n - \mu)
 \end{aligned}$$

where $C = WW^\top + \sigma^2 I$; $\frac{d}{d\mu} = 0 \rightarrow \mu = \bar{x}$.

W, σ^2 are more tricky but again

$$W = [u_1 \quad \dots \quad u_n]$$

where u_i are leading eigenvectors of S .

refer to
Bishop's pa-
per

§4 | Lec 4: Jun 30, 2021

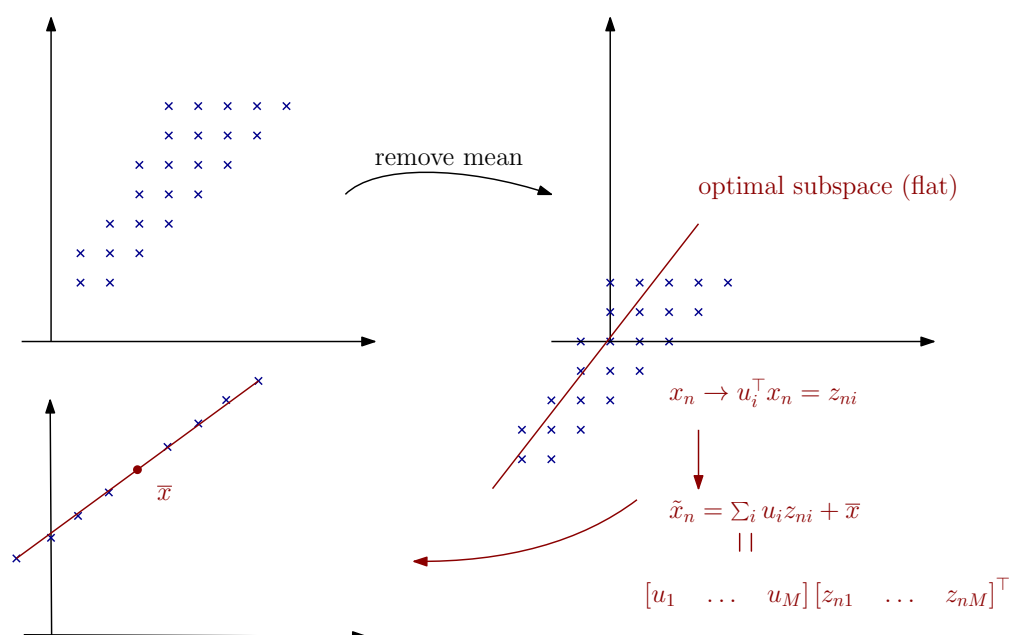
§4.1 Kernel PCA

Recap: Consider standard PCA – $\{x_n\}_{n=1}^N$ where each $x_n \in \mathbb{R}^D$. Assume w.l.o.g, $\sum \bar{x}_n = 0$, then

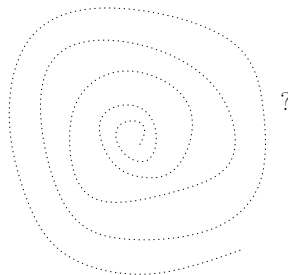
$$S = \frac{1}{N} \sum_{n=1}^N x_n x_n^\top$$

\Rightarrow Principal components are found as leading eigenvectors of S .

$$S u_i = \lambda_i u_i \quad \text{where oftentimes} \quad u_i^\top u_i = 1$$



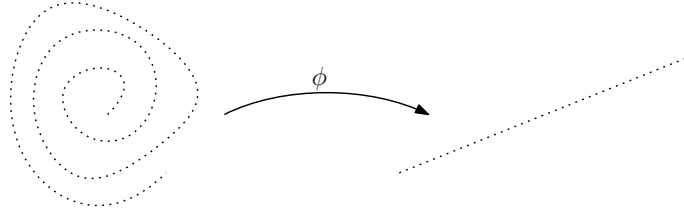
Question 4.1. What happens if we have non-flat data?



One way we can introduce non-linearity:

$$\{x_n\} \rightarrow \{\phi(x_n)\}$$

where $\phi: \mathbb{R}^D \rightarrow \mathbb{R}^E$ (E is possibly much bigger than D)



Let's assume $\{\phi(x_n)\}$ has 0 mean. Then, we can perform PCA on that data set.

$$C = \frac{1}{N} \sum_{n=1}^N \phi(x_n) \phi(x_n)^\top$$

is the relevant covariance matrix. We then need eigenpairs $Cv_i = \lambda_i v_i$, $i = 1, \dots, M$. In fact,

$$k(x_n, x_m) := \phi(x_n)^\top \phi(x_m)$$

where k is the kernel – it would be interesting if we can compute k easily even for large/infinite dimensional ϕ .

more in later lecture!

$$\frac{1}{N} \sum \phi(x_n) \phi(x_n)^\top = \lambda_i v_i$$

For $\lambda_i > 0$, v_i (RHS) is a linear combination of $\phi(x_n)$. So

$$v_i = \sum_{n=1}^N a_{in} \phi(x_n)$$

and we don't know a yet. Let's substitute this into the eigen-equation:

$$\frac{1}{N} \sum_{n=1}^N \phi(x_n) \phi(x_n)^\top \sum_{m=1}^N a_{im} \phi(x_m) = \lambda_i \sum_{n=1}^N a_{in} \phi(x_n)$$

Multiply both sides with $\phi(x_l)^\top$, we have

$$\frac{1}{N} \sum_{n=1}^N \phi(x_l)^\top \phi(x_n) \sum_{m=1}^N a_{im} \phi(x_n)^\top \phi(x_m) = \lambda_i \sum_{n=1}^N a_{in} \phi(x_l)^\top \phi(x_n)$$

Now, we can replace all $\phi^\top \phi$ by the appropriate kernel k :

$$\frac{1}{N} \sum_{n=1}^N k(x_l, x_n) \sum_{m=1}^N a_{im} k(x_n, x_m) = \lambda_i \sum_{n=1}^N a_{in} k(x_l, x_n)$$

Notice that a_{in}, λ_i are the unknowns. In matrix notation,

$$K^2 a_i = \lambda_i N K a_i \implies K a_i = (\lambda_i N) a_i$$

where

$$\begin{cases} K_{m,n} = k(x_m, x_n) \\ a_i = \begin{bmatrix} a_{i1} \\ \vdots \\ a_{iN} \end{bmatrix} \in \mathbb{R}^N \end{cases}$$

So we just need to look for eigenpairs of K , i.e., instead of eigenpairs of $C(\mathbb{R}^{E \times E})$, we now look for eigenpairs of K ($\mathbb{R}^{N \times N}$). But we don't actually need v_i (we will project onto principal components, so all we do is compute inner products with v_i)

z – “Score” = coordinates within manifold

$$\begin{aligned} z(x) &= \phi(x)^\top v_i = \sum_{n=1}^N a_{in} \phi(x)^\top \phi(x_n) \\ &= \sum_{n=1}^N a_{in} k(x, x_n) \end{aligned}$$

Summary of Kernel PCA (kPCA):

- Start with $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^E$
- Build kernel $k(x, y) := \phi(x)^\top \phi(y)$
- Build kernel matrix: $K_{\min} := k(x_m, x_n) \in \mathbb{R}^{N \times N}$
- Eigenpairs $(\lambda_i N, a_i)$ of $K a_i = \lambda_i N a_i$
- For new data point x , $z_i(x) = \sum_{n=1}^N a_{in} k(x, x_n)$

Example 4.1

A kernel that we usually see:

$$k(x, y) = e^{-\frac{\|x-y\|^2}{\sigma^2}}$$

§4.2 Linear Basis Function Models

Goal: Predict the value of one or more continuous (real value) target variables t given the D -dimensional input vector x .

Example 4.2

We have

- $x \mapsto t$ is linear, $x \in \mathbb{R}^D$
- $t = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_D x_D$
- $t = w_0 + w_1 \phi_1(x) + w_2 \phi_2(x) + \dots$ where $\phi(x)$ could be some non-linear basis function (polys, exponentials, ...). The important part here is the linearity w.r.t. parameters.

Linear regression:

$$y(x, w) = w_0 + w_1 x_1 + \dots + w_D x_D$$

where $x \in \mathbb{R}^D$ is data, and w is parameters. Extend to non-linear functions of input

$$y(x, w) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(x)$$

where

$$\begin{cases} \phi_j(x) = \text{basis function} \\ M = \text{degree of freedom} \\ w_0 = \text{offset/bias} \end{cases}$$

For convenience, we often denote

$$\phi_0(x) := 1$$

Then

$$y(x, w) = w^\top \phi$$

where

$$\phi = \begin{bmatrix} \phi_0(x) = 1 \\ \phi_1(x) \\ \vdots \\ \phi_{n-1}(x) \end{bmatrix}, \quad w = \begin{bmatrix} w_0 \\ \vdots \\ w_{n-1} \end{bmatrix}$$

in which $\phi, w \in \mathbb{R}^n$

Example 4.3

Consider:

- $\phi_j = x^j$
- $\phi_j = e^{-\frac{\|x - \mu_j\|^2}{2\sigma^2}}$ – Gaussian.

Question 4.2. How do we find the optimal weights, train/fit the model given the data point?

Approach 1 – Maximum Likelihood/Least Squares

Assumption: Want to find w that make the observed data most likely.

$$\text{Model : } t = y(x, w) + \varepsilon$$

where $\varepsilon \sim N(0, \sigma^2)$ – Gaussian noise. Note that the book uses $\frac{1}{\beta} = \sigma^2$, σ^2 = variance, and β = precision.

$$p(t|x, w, \sigma^2) := N(t|y(x, w), \sigma^2)$$

in which x : location, w : parameter, and σ^2 : noise. Next, let's look at the conditional mean

$$\begin{aligned} \mathbb{E}[t|x] &= \int t p(t|x) dt \\ &= y(x, w) \end{aligned}$$

We have training data $X = \{x_1 \dots x_N\}$ with associated target values $t = \{t_1 \dots t_N\}$. Observed samples are i.i.d.

$$p(\vec{t}|X, w, \sigma^2) = \prod_{n=1}^N N(t_n|y(x_n, w), \sigma^2)$$

The usual approach:

$$\begin{aligned} \ln(\vec{t}|X, w, \sigma^2) &= \sum_{n=1}^N \ln N(t_n|w^\top \phi(x_n), \sigma^2) \\ &= -\frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln(2\pi) - \frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - w^\top \phi(x_n))^2 \end{aligned}$$

Thus,

$$-\ln p \cong \frac{1}{2} \sum_{n=1}^N (t_n - w^\top \phi(x_n))^2 = \text{sum of squared errors}$$

i.e., maximum likelihood is equivalent to minimum squared error.

To minimize error: $\frac{d}{dw} \stackrel{\text{set}}{=} 0$

$$\frac{d}{dw} - \ln p(\vec{t}|X, w, \sigma^2) = \frac{1}{\sigma^2} \sum_{n=1}^N (t_n - w^\top \phi(x_n)) \phi(x_n)^\top$$

As LHS = 0, we have

$$\sum_{n=1}^N t_n \phi(x_n)^\top = w^\top \sum_{n=1}^N \phi(x_n) \phi(x_n)^\top$$

Let's define

$$\Phi := \begin{bmatrix} \phi_0(x_1) & \dots & \phi_{n-1}(x_1) \\ \vdots & & \vdots \\ \phi_0(x_N) & \dots & \phi_{n-1}(x_N) \end{bmatrix} - \text{design matrix}$$

So the solution is

$$w_{ML} = \underbrace{(\Phi^\top \Phi)^{-1} \Phi^\top}_{\text{Moore Penrose pseudo inverse}} \vec{t}$$

From the pseudo-inverse idea, recall that for $A\vec{x} = \vec{b}$ where $\vec{b} \notin \text{range}(A)$, we have

$$\begin{aligned} \min \frac{1}{2} \|A\vec{x} - \vec{b}\|^2 \quad & \text{(least squares problem)} \\ \implies \vec{x} &= (A^\top A)^{-1} A^\top \vec{b} \end{aligned}$$

Question 4.3. What is the MLE for σ^2 ?

Same approach as above but we take the derivative with respect to σ^2 and set that equal to 0.

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N \left(\underbrace{t_n - w_{ML}^\top \phi(x_n)}_{\text{residual(error)}} \right)^2$$

Remark 4.4. In the context of “big data” – N, D big (or M : # of basis functions big), Φ is not going to fit into memory and/or difficult to handle or data visible in portions (streaming). So we need sequential learning in which we use “gradual updates” to estimates of w_{ML}

$$\begin{aligned} w^0 &= \text{initial guess} \\ w^{n+1} &= w^n - \underbrace{\eta \frac{d}{dw} E}_{\text{gradient descent}} \end{aligned} \quad (*)$$

where E is the loss function evaluated for current batch of data points and η is the stepsize. We hope that $(*)$ converge to optimal parameters. In details,

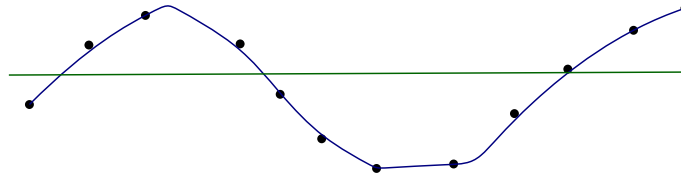
$$w^{n+1} = w^n - \eta \left(t_n - w^{nT} \phi(x_n) \right) \phi(x_n)^\top$$

One of the crucial step here is to choose the step-size carefully.

§5 | Lec 5: Jul 1, 2021

§5.1 Overfitting

Consider:



Through a first glance, it might be intuitive to assume that the data is generated from the blue curve ... However, it's not that clear, and the data may actually stem from the green line. In fact, it's very difficult to tell as the data itself does not communicate well with us here (noise level?). This phenomenon, in one direction, is called **overfitting**.

Assume that $\{x\}$ was generated from the blue curve, and we try to fit/learn the green line instead; this is the definition of overfitting. In essence, we try to explain some of the “wiggle” (variance) we see in the data by a more complicated/unnecessary model – a very dangerous process. Overfit is a consequence of too powerful models (often too many options to learn from).

Thus, to avoid overfitting, we use 3 data sets:

1. training data set $\{x_n\} \& \{t_n\}$
2. validation: $\{x_n\} \& \{t_n\}$ (10-20 % of data not used for training)
→ run “trained” model and see how well it performs.
3. “real data”: $\{x_n\} \rightarrow \{t_n\}$ inferred using trained model.

Overall, our ultimate goal is to test the model's ability to “generalize” or perform on new/unseen data.

refer to **overfitting** for more details on the topic

§5.2 Regularized Least Squares

Goal: we want to control overfitting – include a regularization term in addition to the data term.

Connection to linear algebra:

$$\left\{ \begin{array}{l} A\vec{x} = \vec{b} \text{ does not have a solution} \\ A \text{ ill-conditioned (no solution, not unique solution, sensitive to } \Delta\vec{b}) \\ \Rightarrow \text{Tikhonov - regularization :} \\ \min \frac{1}{2} \|A\vec{x} - \vec{b}\|_2^2 + \underbrace{\|\Gamma\vec{x}\|_2^2}_{\text{regularizer}} \end{array} \right.$$

Data-term for linear regression

$$E_0(w) = \frac{1}{2} \sum_{n=1}^N (t_n - w^\top \phi(x_n))^2$$

Simple regularizer: quadratic penalty on w

$$E_w(w) = \frac{1}{2} w^\top w = \frac{1}{2} \|w\|^2$$

then

$$\min_w E_0(w) + \lambda E_w(w) = \frac{1}{2} \sum_{n=1}^N (t_n - w^\top \phi(x_n))^2 + \frac{\lambda}{2} w^\top w$$

Now, we can set $\frac{d}{dw} = 0$,

$$w = (\lambda I + \Phi^\top \Phi)^{-1} \Phi^\top \vec{t}$$

This shrinks the component of w towards 0 if compared to

$$w_{ML} = (\Phi^\top \Phi)^{-1} \Phi^\top \vec{t}$$

More generally (more modern),

$$E_w(w) := \frac{1}{2} \sum_{j=1}^M |w_j|^q$$

- $q = 1 \rightarrow$ “LASSO” which has tendency to promote sparsity (some coeffs of w will be exactly 0)
- $q = 2 \rightarrow$ same as above.

refer to [wiki](#)
for more de-
tails

To summarize, to address the problem of overfitting, we have the following ways

- Model complexity: keep model simple (restricted set of basis functions)
- Regularization: encourage simple coefficients by adding penalty to complex choices.

§5.3 Bayesian Linear Regression

Let's start by introducing a prior distribution on w (for now we consider noise $\sigma^2 = \frac{1}{\text{position}}$ known). Recall from the last lecture,

- $p(t|x_n, w, \sigma^2)$ is Gaussian, i.e., $= N(t_n | w^\top \phi(x_n), \sigma^2)$
- conjugate prior: $p(w) = N(w | m_0, S_0)$ is Gaussian too.

The posterior will also be Gaussian.

$$p(w | \vec{t}, x, \sigma^2) = N(w | m_N, S_N)$$

which is in basically through Bayes' rule

$$p(w | \text{data}) = \frac{p(\text{data} | w) p(w)}{p(\text{data})}$$

From exercise 3.7 in the book,

$$m_N = S_N \left(S_0^{-1} m_0 + \frac{1}{\sigma^2} \Phi^\top \vec{t} \right)$$

and

$$S_N^{-1} = S_0^{-1} + \frac{1}{\sigma^2} \Phi^\top \Phi \quad (\text{precision})$$

So

$$\boxed{\text{posterior precision} = \text{prior precision} + \text{data precision}}$$

Remark 5.1. If $S_0^{-1} = 0 (\implies S_0 = \frac{1}{\alpha} I, \alpha \rightarrow 0)$ non-informative prior

$$S_N^{-1} = \frac{1}{\sigma^2} \Phi^\top \Phi \rightarrow m_N = \sigma^2 (\Phi^\top \Phi)^{-1} \frac{1}{\sigma^2} \Phi^\top \vec{t} = (\Phi^\top \Phi)^{-1} \Phi^\top \vec{t}$$

Notice that

$$\operatorname{argmax}_w p(w|\vec{t}, X, \sigma^2) = \operatorname{argmax}_w N(w|m_N, S_N)$$

Gaussian: $\operatorname{argmax}_w n(w|m_N, S_N) = m_N$

$$w_{\text{MAP}} = m_N = S_N \left(S_0^{-1} m_0 + \frac{1}{\sigma^2} \Phi^\top \vec{t} \right)$$

The second special case: $N \rightarrow 0$

$$m_N \rightarrow m_0 \quad (\text{prior becomes dominant})$$

Choose a special prior: $m_0 = 0; S_0 = \frac{1}{\alpha} I$ (zero mean, isotropic Gaussian prior).

$$\Rightarrow \begin{cases} m_N = \frac{1}{\sigma^2} S_N \Phi^\top \vec{t} \\ S_N^{-1} = \alpha I + \frac{1}{\sigma^2} \Phi^\top \Phi \end{cases}$$

Now, we can take $-\log$ of posterior

$$-\ln p(w|\vec{t}, X, \sigma^2) = \underbrace{\frac{1}{2\sigma^2} \sum_{n=1}^N (t_n - w^\top \phi(x_n))^2 + \frac{\alpha}{2} w^\top w}_{=E_0(w) + E_w(w)} + \text{constant}(w)$$

Predictive Distribution

w is/are not actually the object of interest. Rather, we want to predict t for new x . For a new query location x and new target variable t ,

$$p(t|x, \vec{t}, X, \alpha, \sigma^2) = \int p(t|x, w, \sigma^2) \cdot p(w|\vec{t}, X, \alpha, \sigma^2) dw$$

then

$$\int N(t|w^\top \phi(x), \sigma^2) N(w|m_N, S_N) dw = N * N$$

and the convolution of two Gaussians is Gaussian. So we only need to find the right parameters

$$p(t|x, \vec{t}, X, \alpha, \sigma^2) = N(t|m_N^\top \phi(x), \sigma_N^2(x))$$

where $\sigma_N^2(x) = \sigma^2 + \phi(x)^\top S_N \phi(x)$ in which the first term is general noise, and second term is uncertainty on w due to proximity/distance of training data.