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

§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, $X, Y \in \{x_i\} \times \{y_i\}$

  1. $\text{prob}(X = x_i, Y = y_i) = \frac{n_{ij}}{N} = p(x_i, y_i)$ – 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

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

  $\implies$ 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 $f : \mathbb{R} \to \mathbb{R}$ 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:

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

Covariances:

Definition 1.3 — Variance is defined as

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

Covariance (random variables) is defined as

$$\begin{aligned}\text{cov}[x, y] &:= \mathbb{E}\left[(x - \mathbb{E}[x])\,(y - \mathbb{E}[y])\right] \\ &= \mathbb{E}[xy] - \mathbb{E}[x]\mathbb{E}[y]\end{aligned}$$

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:

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

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

1. Frequentist: maximize likelihood

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

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

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

   Then, we can estimate

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

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

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

§1.2 Gaussian Distribution

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

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

where μ is the mean and σ^2 is the variance.
For D-dimensional,

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

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 <u>precision matrix</u> Λ as

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

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:

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

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.

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

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

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

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?

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

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

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

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

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

  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)

  $$\max_{\mu} p\left(\mu,\Sigma|X\right) \overset{\text{Bayes'}}{\Longrightarrow} \max_{\mu} p\left(X|\mu,\Sigma\right)\cdot p\left(\mu\right)$$

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

$$-\ln p\left(X|\mu,\Sigma\right)\cdot p\left(\mu|\mu_0,\Sigma_0\right)$$

$$\min_{\mu}\frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^\top\Sigma^{-1}(x_n-\mu)+\frac{1}{2}(\mu-\mu_0)^\top\Sigma_0^{-1}(\mu-\mu_0)$$

$$\frac{d}{d\mu}=0:\ \sum_{n=1}^{N}\Sigma^{-1}(x_n-\mu)+\Sigma_0^{-1}(\mu-\mu_0)=0$$

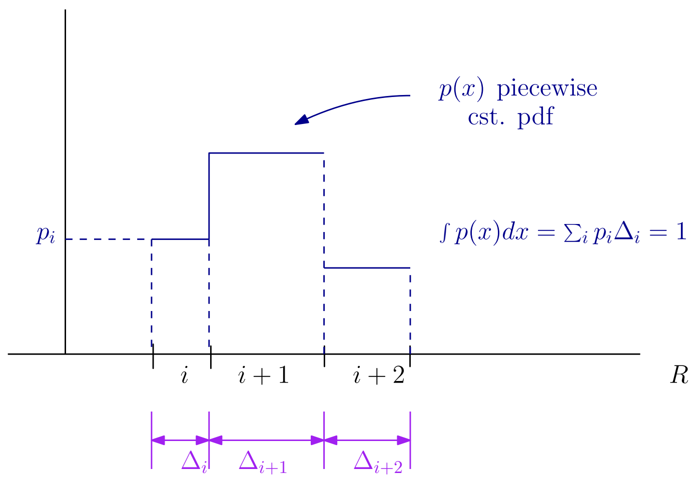
$$\implies\mu_{\text{MAP}}=\left(N\Sigma^{-1}+\Sigma_0^{-1}\right)^{-1}\left(N\Sigma^{-1}\overline{x}+\Sigma_0^{-1}\mu_0\right)$$

§2.2 Non-parametric Probability Density Function (Estimation)

Let's consider the following

- Histograms
- partition domain of x into distinct bins of width $\triangle_i$
- count number of observations n_i of x falling into bin i
- divide by $N,\triangle_i$ 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., $\Delta_i=\Delta$

refer to fig 2.24 in textbook for other cases

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 $K\sim\text{Binomial}(N,p)$

$$\mathbb{E}\left[\frac{K}{N}\right]=p$$

$$\text{var}\left[\frac{K}{N}\right]=\frac{p(1-p)}{N}$$

$$\text{var}\left[\frac{K}{N}\right]\xrightarrow[N\to\infty]{}0$$

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. $p(x) \sim$ 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,

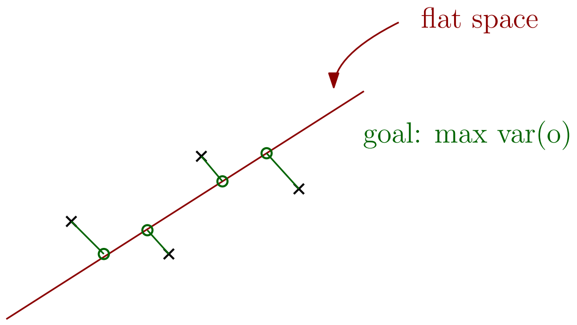
$$\begin{cases} k(u) \geq 0 \\ \int k(u)\, du = 1 \end{cases}$$

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 $\{x_n\}$, $n = 1, \ldots, N$, $x_n \in \mathbb{R}^D$. 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: $\overline{x} = \frac{1}{N}\sum_{n=1}^N x_n$. Note that the variance before projection is

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

and after projection is

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

with

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

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

$$\begin{aligned} \frac{\partial L}{\partial u_1} &: \; 2Su_1 - 2\lambda_1 u_1 = 0 \\ & \quad Su_1 = \lambda_1 u_1 \end{aligned}$$

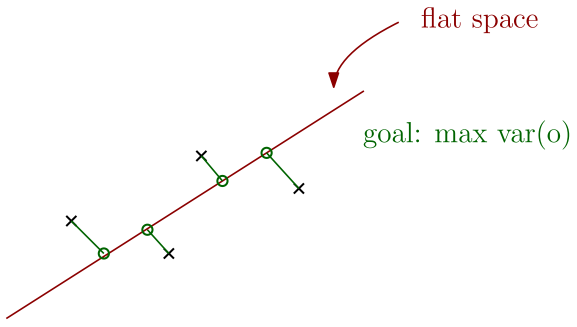
So, the eigen-problem: (λ_1, u_1) is eigenpair of S.

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

$\implies$ 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 $u_1, \ldots, u_n$ as the M leading eigenvectors of S where all u_i are orthogonal and

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

<u>**Minimum Error Formulation**</u>:



Goal: introduce as little distortion as possible.
Consider: $\{u_i\}, i = 1, \ldots, D$ orthonormal basis of $\mathbb{R}^D$

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

Then each data point x_n has unique expansion in that basis

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

where

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

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. $b_i = \overline{x}^\top u_i$, $i = M+1, \dots, D$

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

$$\tilde{x}_n = \sum_{i=1}^{M} \left(x_n^\top u_i\right) u_i + \sum_{i=M+1}^{D} \left(\overline{x}^\top u_i\right) u_i$$

$$x_n - \tilde{x}_n = \sum_{i=M+1}^{D} \left(x_n^\top u_i - \overline{x}^\top u_i\right) u_i$$

In addition, the error term can be written as

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

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 $\overline{x} = 0$. Then, $S = \frac{1}{N} x^\top x$

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

where each x_n is a row of X. As $\overline{x} = 0$, rows sum up to 0.
Let's examine the eigenvalues of $x^\top x$ v.s. eigenvalues of $xx^\top$.

$$\frac{1}{N} x^\top x u_i = \lambda_i u_i$$

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

$$\frac{1}{N} x x^\top v_i = \lambda_i v_i$$

§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

$$p(z) = N\left(z | 0, I\right)$$

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

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

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.*

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

Proof. Sufficient statistics

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

For the covariance,

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

Remark 3.1. $\mathbb{E}\left[z\varepsilon^\top\right] = 0 = \mathbb{E}\left[\varepsilon z^\top\right]$ 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 $p(x) = N\left(x|\mu, C\right)$. We need C^{-1}.

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

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,

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

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

refer to Bishop's paper

W, σ^2 are more tricky but again

$$W = \begin{bmatrix} u_1 & \dots & u_n \end{bmatrix}$$

where u_i are leading eigenvectors of S.

§4 Lec 4: Jun 30, 2021

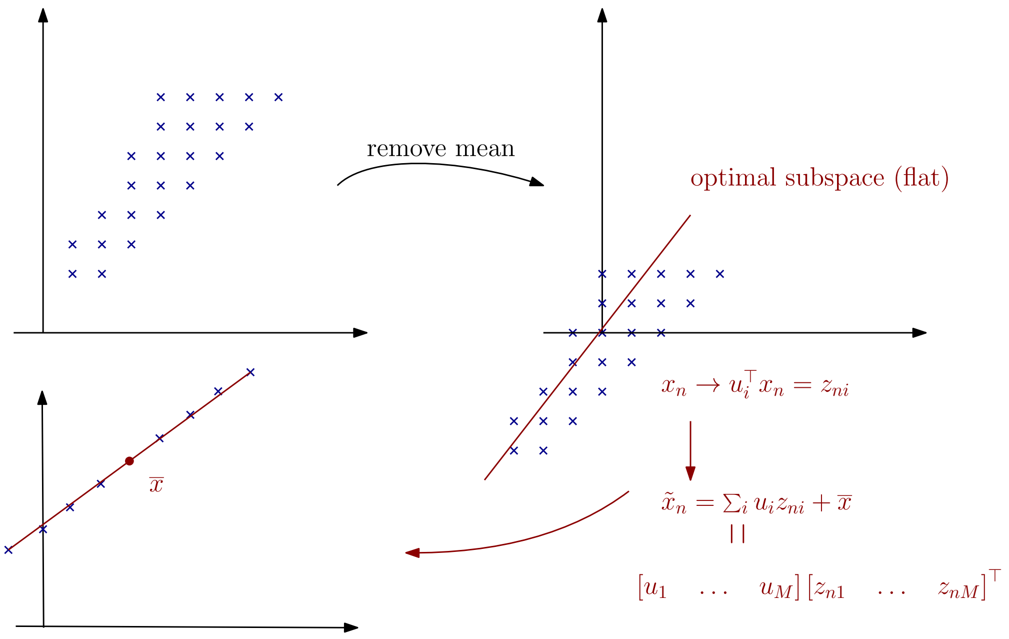
§4.1 Kernel PCA

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

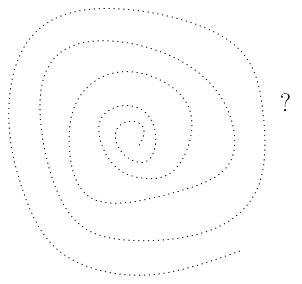
$$S = \frac{1}{N}\sum_{n=1}^N x_n x_n^\top$$

$\implies$ Principal components are found as leading eigenvectors of S.

$$Su_i = \lambda_i u_i \qquad \text{where oftentimes} \qquad u_i^\top u_i = 1$$


remove mean
optimal subspace (flat)
$x_n \to u_i^\top x_n = z_{ni}$
$\tilde{x}_n = \sum_i u_i z_{ni} + \overline{x}$
$||$
$[u_1 \; \dots \; u_M]\,[z_{n1} \; \dots \; z_{nM}]^\top$
$\overline{x}$

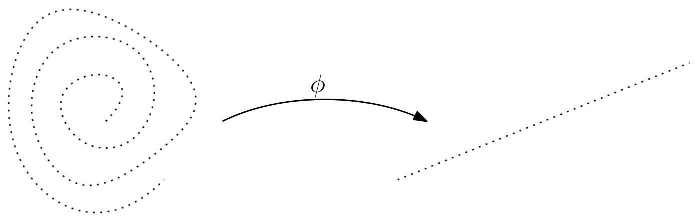
Question 4.1. What happens if we have non-flat data?



One way we can introduce non-linearity:

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

where $\phi : \mathbb{R}^D \to \mathbb{R}^E$ (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,

$$k(x_n, x_m) \coloneqq \phi(x_n)^\top \phi(x_m)$$

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 $\phi : \mathbb{R}^D \to \mathbb{R}^E$
- Build kernel $k(x,y) \coloneqq \phi(x)^\top \phi(y)$
- Build kernel matrix: $K_{\text{min}} \coloneqq k(x_m, x_n) \in \mathbb{R}^{N\times N}$
- Eigenpairs $(\lambda_i N, a_i)$ of $Ka_i = \lambda_i N a_i$
- 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, $x \in \mathbb{R}^D$
- $t = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_D x_D$
- $t = w_0 + w_1\phi_1(x) + w_2\phi_2(x) + \ldots$ 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:

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

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

$$\phi_0(x) \coloneqq 1$$

Then

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

where

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

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.

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

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.

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

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

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

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

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

The usual approach:

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

Thus,

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

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

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

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

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

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

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

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

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

Same approach as above but we take the derivative <u>with repspect to σ^2</u> 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}

$$w^0 = \text{initial guess}$$
$$w^{n+1} = \underbrace{w^n - \eta\frac{d}{dw}E}_{\text{gradient descent}} \tag{*}$$

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,

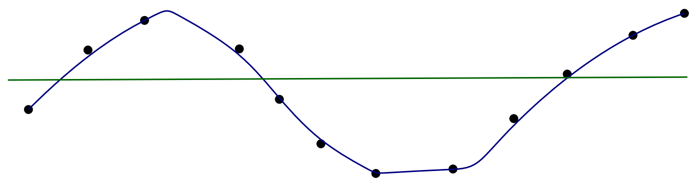
$$w^{n+1} = w^n - \eta\left(t_n - w^{nT}\phi(x_n)\right)\phi(x_n)^\top$$

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)
   $\rightarrow$ 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:

$$\begin{cases} A\vec{x} = \vec{b} \text{ does not have a solution} \\ A \text{ ill-conditioned (no solution, not unique solution, sensitive to } \triangle\vec{b}) \\ \implies \text{Tikhonov – regularization :} \\ \quad \min \frac{1}{2}\|A\vec{x}-\vec{b}\|_2^2 + \| \underbrace{\Gamma\vec{x}}_{\text{regularizer}} \|_2^2 \end{cases}$$

Data-term for linear regression

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

Simple regularizer: quadratic penalty on w

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

then

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

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

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

This shrinks the component of w towards 0 if compared to

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

More generally (more modern),

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

- $q = 1 \rightarrow$ "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 details

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

The posterior will also be Gaussian.

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

which is in basically through Bayes' rule

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

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

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

Notice that

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

<u>Gaussian</u>: $\operatorname{argmax}_w n\left(w|m_N, S_N\right) = m_N$

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

The second special case: $N \to 0$

$$m_N \to m_0 \qquad \text{(prior becomes dominant)}$$

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

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

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

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

<u>Predictive Distribution</u>

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,

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

then

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

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

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

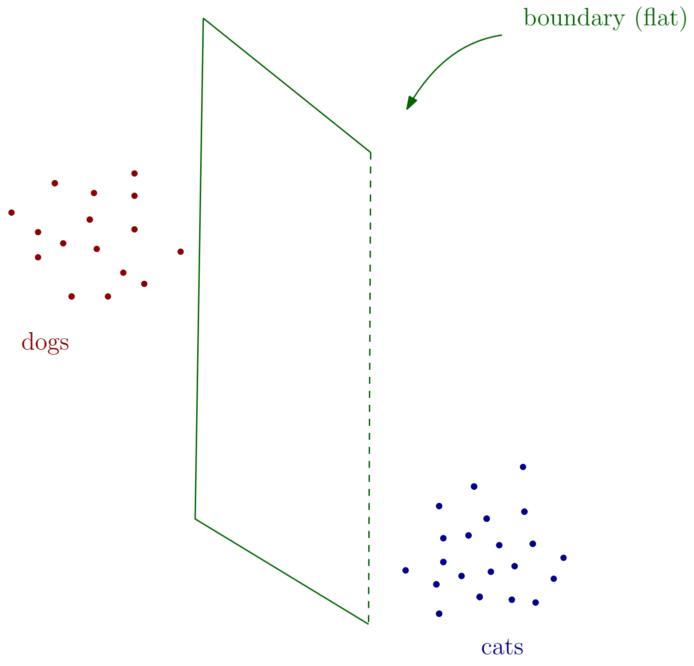
where $\sigma_N^2(x) = \sigma^2 + \phi(x)^\top S_N\phi(x)$ in which the first term is general noise, and second term is uncertainty on w due to proximity/distance of training data.

§6 Lec 6: Jul 7, 2021

§6.1 Linear Models for Classification

Goal: $\mathbb{R}^D \ni x \mapsto C_k$ one of k discrete classes $k = 1 \ldots k$. Note that classes are disjoint, i.e., x belongs to exactly 1 class. Thus, classification is equivalent to partitioning of $\mathbb{R}^D$ (decision regions separated by decision boundaries/surfaces).

$$\text{linear model} \iff \text{decision surfaces are } D-1 \text{ dimensional hyperplanes}$$



Representation:

$$\begin{cases} t \in \{0;1\}, & k=2 \\ \text{or } \{-1;+1\}, & \text{“binary”} \\ t \in \{1,\ldots,k\}, & k \geq 2 - \text{not use in practice} \\ t \in \{0,1\}^k;\ |t| = 1, & \text{1 in k coding (vector)} \end{cases}$$

There are 3 approaches to classification problems

1. discriminant function: $x \mapsto C_k$
2. probability based
   a) discriminative: $p(C_k|x)$
   b) generative model: $p(C_k) \cdot p(x|C_k)$

   Let's dive right into the first approach.

General form of linear model:

$$y(x) = f\left(w^\top x + w_0\right)$$

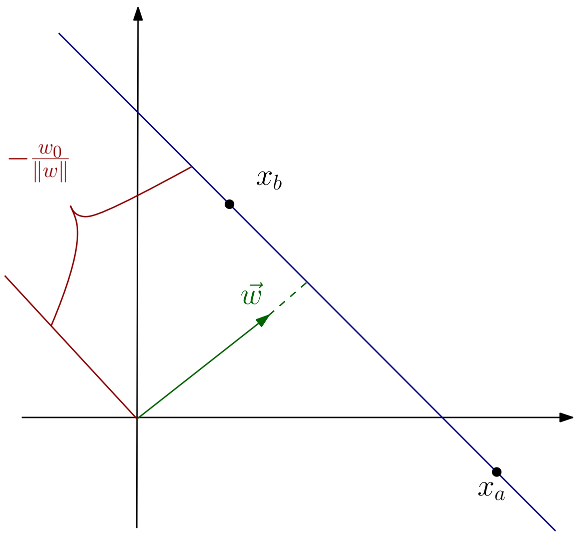
where f is an activation function.
Decision surface: $y(x) =$ constant which means

same notation later on for deep learning

$$w^\top x + w_0 = \text{constant}$$

Take $k = 2$, $f :=$ sign, i.e., $y(x) \in \{-1;+1\}$.

Claim 6.1. Let x_a, x_b on decision boundary (which means $w^\top x_a + w_0 = 0$, etc). Then $w \perp x_a - x_b$.



Proof. WTS: $w^\top(x_a - x_b) = 0$.

$$\begin{aligned} w^\top(x_a - x_b) &= w^\top x_a - w^\top x_b + w_0 - w_0 \\ &= \left(w^\top x_a + w_0\right) - \left(w^\top x_b + w_0\right) \\ &= 0 - 0 = 0 \end{aligned}$$ □

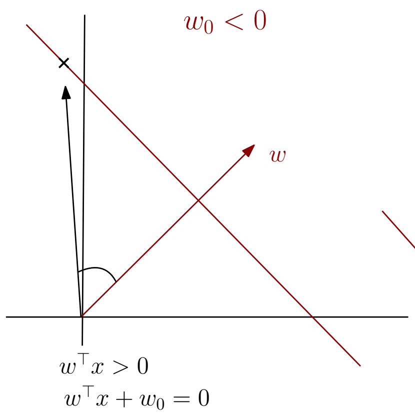
So w is orthogonal to $x_a - x_b$.

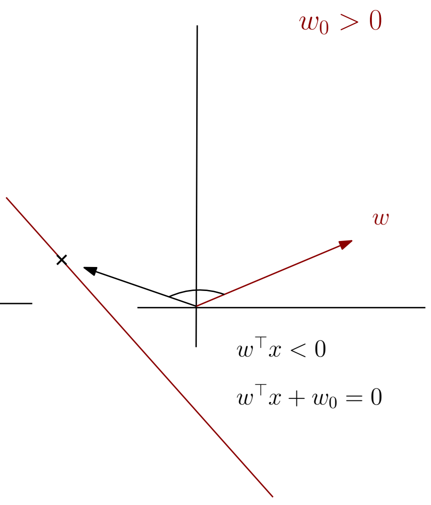
Claim 6.2. Signed distance between the origin and decision surface is $\frac{-w_0}{\|w\|}$.

Proof. As x is on decision surface, we have $w^\top x = -w_0$. Signed distance between origin and decision surface

$$\frac{w^\top}{\|w\|} x = -\frac{w_0}{\|w\|}$$

□





Claim 6.3. Signed distance between x and decision surface is $\frac{w^\top x + w_0}{\|w\|}$.

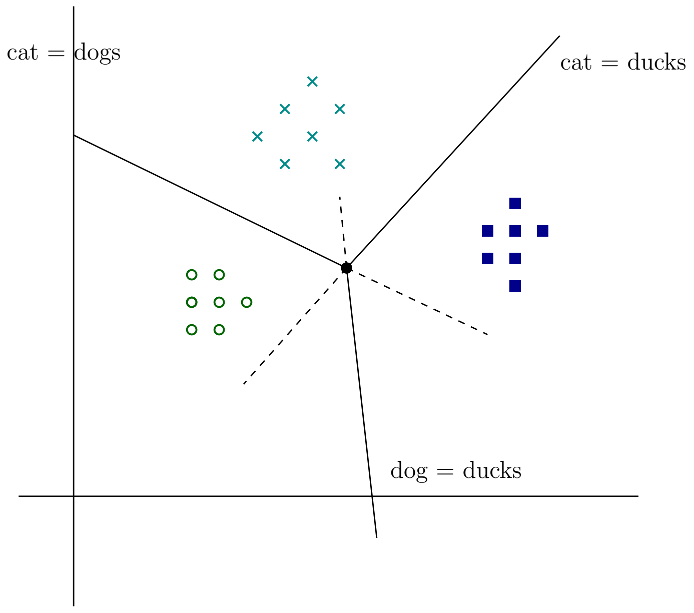
Proof. We have

$$\begin{aligned}
x &= x_p + r\frac{w}{\|w\|} \\
w^\top x &= w^\top x_p + r\frac{w^\top w}{\|w\|} \\
w^\top x + w_0 &= (w^\top x_p + w_0) + r\frac{w^\top w}{\|w\|} \\
w^\top x + w_0 &= r\|w\| \\
r &= \frac{w^\top x + w_0}{\|w\|}
\end{aligned}$$

□

That's all the geometry we need. Next, let's take a quick look at multiclass extension. Define $y_k(x) = w_k^\top x + w_{k0}$ for each class. Then assign class by winner takes all. For a new x: $y_1(x), y_2(x), \ldots, y_k(x)$ then say x class j if $y_i \geq y_k(x)\ \forall k$.
Decision boundary: ($b|w$ class k & j)

$$\begin{gathered}
w_k^\top x + w_{k_0} = w_j^\top x + w_{j_0} \\
\underbrace{(w_k - w_j)^\top}_{\text{orth}} x + \underbrace{(w_{k_0} + w_{j_0})}_{\text{bias}} = 0 \\
\implies \text{decision surfaces} = D - 1 \text{ dimensional hyperplanes}
\end{gathered}$$



Claim 6.4. Decision regions are always convex.

Proof. Let $x_a, x_b \in$ class k. Convexity means that for any $\lambda \in (0, 1]$

$$\lambda x_a + (1 - \lambda)x_b \in \text{class } k$$

We then have

$$\begin{aligned}\forall j \neq k : \, w_k^\top x_a + w_{k_0} &> w_j^\top x_a + w_{j_0} \\ w_k^\top x_b + w_{k_0} &> w_j^\top x_b + w_{j_0}\end{aligned}$$

Consider: $x = \lambda x_a + (1-\lambda) x_b$

$$\begin{aligned}w_k^\top x + w_{k_0} &= w_k^\top (\lambda x_a + (1-\lambda)x_b) + w_{k_0} \\ &= \lambda \left(w_k^\top x_a + w_{k_0}\right) + (1-\lambda)\left(w_k^\top x_b + w_{k_0}\right) \\ w_j^\top x + w_{j_0} &= \ldots = \lambda \left(w_j^\top x_a + w_{j_0}\right) + (1-\lambda)\left(w_j^\top x_b + w_{j_0}\right)\end{aligned}$$

$\implies$ x is also in class k. □

Probabilistic Generative Models

Goal: get linear models as a result of probabilistic modeling.
Generative model:

1. class conditional probabilities: $p(x|C_k)$
2. class priors: $p(C_k)$

We want to use Bayes' to compute the estimate $p(C_k|x)$. We will consider the two-class case only.

$$p(C_1|x) = \frac{p(x|C_1) \cdot p(C_1)}{p(x|C_1)p(C_1) + p(x|C_2)p(C_2)}$$

Let's define

$$a := \ln \frac{p(x|C_1)p(C_1)}{p(x|C_2)p(C_2)}$$

Then,

$$p(C_1|x) = \frac{1}{1+e^{-a}} =: \sigma(a)$$

which we call sigmoid(a). Notice that the logistic sigmoid $\sigma : \mathbb{R} \to [0,1]$

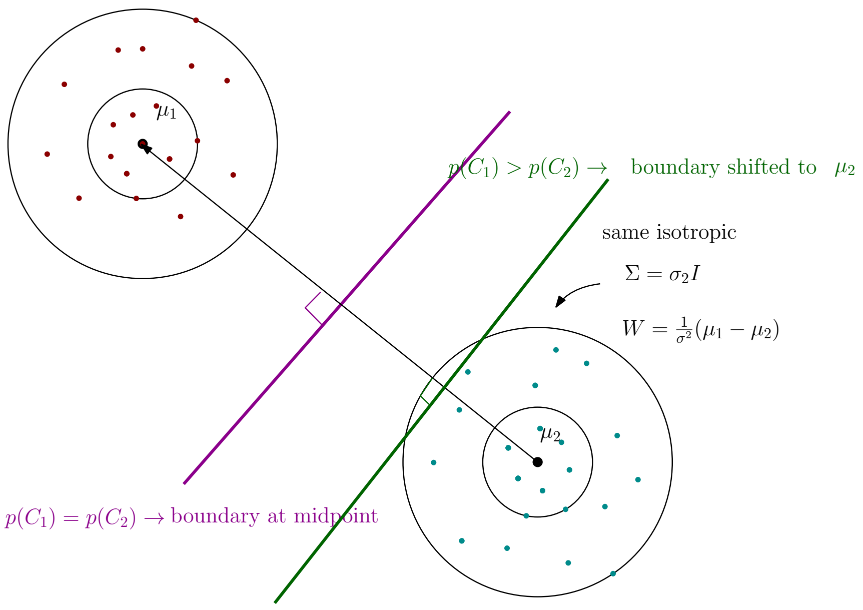
- symmetry: $\sigma(-a) = 1 - \sigma(a)$
- inverse: $a = \ln\left(\frac{\sigma}{1-\sigma}\right)$ – logit function

We now define class-conditional probability (Gaussian). Then

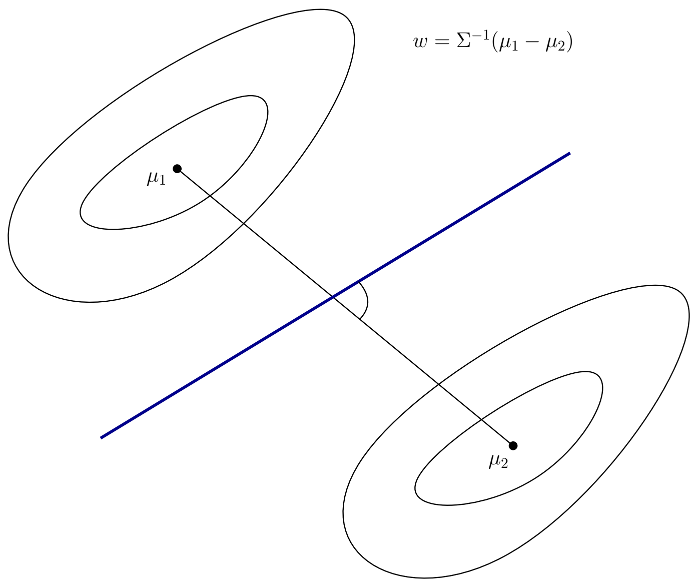
$$p(x|C_k) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^\top \Sigma^{-1}(x-\mu_k}$$

where we assume class-specific μ_k; common Σ. So

$$p(C_1|x) = \sigma(w^\top x + w_0), \quad \begin{cases} w = \Sigma^{-1}(\mu_1 - \mu_2) \\ w_0 = -\frac{1}{2}\mu_1^\top \Sigma^{-1}\mu_1 + \frac{1}{2}\mu_2^\top \Sigma^{-1}\mu_2 + \ln \frac{p(C_1)}{p(C_2)} \end{cases}$$


μ_1
μ_2
$p(C_1) > p(C_2) \to$ boundary shifted to μ_2
same isotropic
$\Sigma = \sigma_2 I$
$W = \frac{1}{\sigma^2}(\mu_1 - \mu_2)$
$p(C_1) = p(C_2) \to$ boundary at midpoint

For a more general Σ (not necessarily isotropic)



We will use MLE to learn $\Sigma, p(C_1)/p(C_2)$, and μ_1, μ_2

$$\text{Data:} \quad X = \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix}$$
$$\vec{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}; \quad t_n \in \{0,1\}$$
$$p(C_1) = \underbrace{\pi}_{\neq 3.14\ldots}, \quad p(C_2) = 1 - \pi$$

For $x_n \in C_1 : t_n = 1$, and

$$p(x_n, C_1) = p(C_1)p(x_n|C_1) = \pi N\left(x_n|\mu_1, \Sigma\right)$$

Similarly, for $x_n \in C_2 : t_n = 0$, and

$$p(x_n, C_2) = p(C_2)p(x_n|C_2) = (1-\pi)N(x_n|\mu_2, \Sigma)$$

Because the data is assume i.i.d:

$$p\left(\vec{t}, x|\pi, \mu_1, \mu_2, \Sigma\right) = \prod_{n=1}^{N} \left[\pi \underbrace{N\left(x_n|\mu_1, \Sigma\right)}_{N_1}\right]^{t_n} \left[(1-\pi) \underbrace{N\left(x_n|\mu_2, \Sigma\right)}_{N_2}\right]^{1-t_n}$$

As usual, we want to maximize the log-likelihood.

$$\ln p\left(\vec{t}, x|\pi, \mu_1, \mu_2, \Sigma\right) = \sum_{n=1}^{N} t_n \left[\ln \pi + \ln N_1\right] + (1-t_n)\left[\ln(1-\pi) + \ln N_2\right]$$

and set the derivatives of each term to zero.

- π

$$\sum_{n=1}^{N} t_n \ln \pi + (1-t_n)\ln(1-\pi)$$

  Set $\frac{d}{d\pi} = 0$, then

$$\pi = \frac{1}{N}\sum_{n=1}^{N} t_n = \frac{\text{\# of data pts in class 1}}{N}$$

- μ_k, let's consider $k = 1$, i.e., μ_1

$$\sum_{n=1}^{N} t_n \left(-\frac{1}{2}(x_n - \mu_1)^\top \Sigma^{-1}(x_N - \mu_1)\right)$$

  Set $\frac{d}{du_1} = 0$ as before and we get

$$\mu_1 = \frac{1}{\#1}\sum_{n=1}^{N} t_n x_n$$

  Similarly,

$$\mu_2 = \frac{1}{\#2}\sum_{n=1}^{N} (1-t_n) x_n$$

Probabilistic Discriminative Models

Probabilities are good, but we have to deal with too many parameters, e.g., $\pi, \mu_1, \mu_2, \Sigma$. Knowing that the shared covariance matrix Σ leads to linear models we can try to learn $p(C_k|x)$ directly.

$$p(C_k|x) = \sigma\left(w_k^\top x + w_{k_0}\right) \Longleftarrow \text{logistic regression}$$

§7 Lec 7: Jul 12, 2021

§7.1 Ensemble Methods

The basic idea of ensemble method is "pool" (average) predictions stemming from a *diverse* set of models (regression/classification). This is another way to address the issue of overfitting, in which we hope that multiple models will

1. overfit differently
2. capture the structure coherently

When trained on different parts of the data, noise realizations will hopefully cancel out (in statistical sense) in aggregation because they are statistically independent. Meanwhile, the structural components will be reinforced. There are two crucial steps to this problem.

1. **Bootstrapping**

   Given a data set D of size N (N data points in the set). We can generate M new training sets D_m each of size N' by sampling from D uniformly and with replacement.

   > **Remark 7.1.** If $N' = N$, then each D_m contains $\left(1 - \frac{1}{e}\right) \approx 63\%$ of unique elements of D and the rest is duplicates.

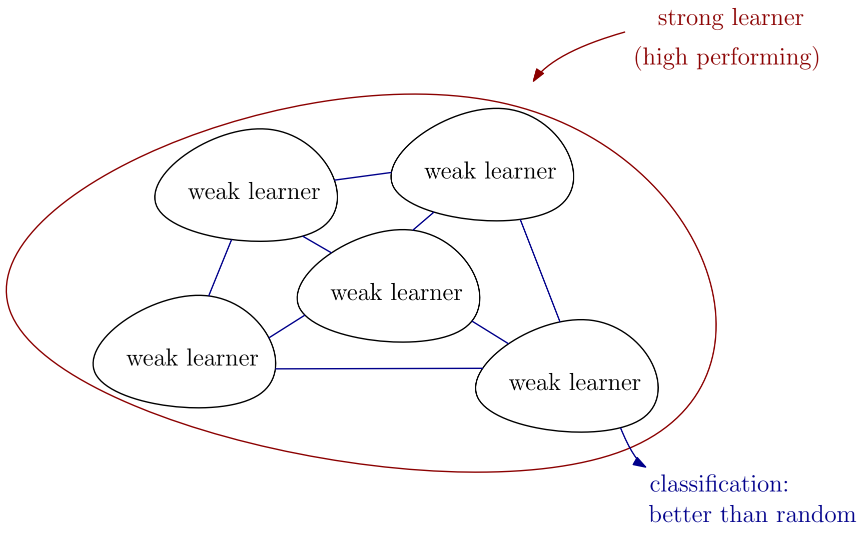
   By using the M "new" training data sets D_m, we train M models.

2. **Aggregation**

   $\rightarrow$ combine output of M models by

   - averaging (regression)
   - majority vote (classification)

This process is also known as "bagging" (bootstrap aggregating). Notice that the M models are all equally dumb as they don't learn from each other's mistakes.



Adaptive Boosting (AdaBoost for classification)
Fundamentally, we still have M weak learners (slightly better than random), and we train them in sequence where training focuses on data points that were previously misclassified. In this case, the output is weighted average; this provably results in strong learners.

Algorithm:
Input:

- location: $x_1, \ldots, x_N \in \mathbb{R}^D$
- binary targets: $t_1, \ldots, t_N \in \{-1; +1\}$

A family of weak learners: $y_w(x)$ where w is parameters (discrete/continuous) and

$$y_w(x) \in \{-1; +1\}$$

For example, $y_w(x) = \operatorname{sign}(w^\top x + w_0)$.
Initial weights:

$$w_n^{(1)} = \frac{1}{N}$$

which is not a parameter, but it's weight attached to each data point. So they all have same weight initially and add up to one. For $m = 1, \ldots, M$,

- train/select classifier $y_w^m(x)$ – minimizing:

$$J_m := \sum_{n=1}^{N} w_n^{(m)} \underbrace{\mathbf{1}\left(y_w^m(x_n) \neq t_n\right)}_{\text{error indicator}}$$

  where $w_n^{(m)}$ is the current weight of x_n, $y_w^m(x_n)$ is the current prediction for x_n, and t_n is the label. Also, when the arguments inside $\mathbf{1}$ are equal, the term becomes zero, or one otherwise.

- estimate learner performance:

$$\varepsilon_m := \frac{\sum_{n=1}^{N} w_n^{(m)} \mathbf{1}\left(y_w^m(x_n) \neq t_n\right)}{\sum_{n=1}^{N} w_n^{(m)}}$$

- weight learner:

$$\alpha_m := \ln \frac{1 - \varepsilon_m}{\varepsilon_m}$$

- update the data weighting:

$$w_n^{(m+1)} = w_n^{(m)} e^{\alpha_m \mathbf{1}(y_w^m(x_n) \neq t_n)}$$

- prediction:

$$y_m(x) = \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m y_w^m(x)\right)$$

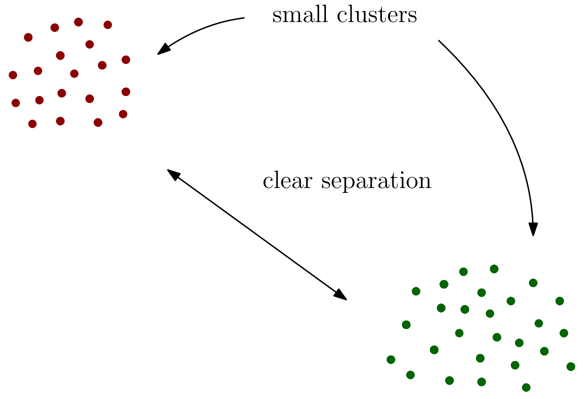
Notice that

- the perfect learner has $\varepsilon_m = 0 \implies \alpha_m \to \infty$
- the perfect liar has $\varepsilon_m = 1 \implies \alpha_m \to -\infty$
- the random learner has $\varepsilon_m = \frac{1}{2} \implies \alpha_m = 0$
- $\varepsilon_m > \frac{1}{2} \implies \alpha_m < 0$

§7.2 K-Means Clustering

Clustering is basically classification without training labels. It's the partitioning of data points into "meaningful" groups from "scratch" (no training data).
Goal: Given data $\{x_n\}_{n=1}^N$, $x_n \in \mathbb{R}^D$ group samples into K clusters s.t.

- within a cluster: distances are small
- between clusters: distances are big



The trick here is to introduce $\mu_k \in \mathbb{R}^D$, $k = 1, \dots, K$ to represent the cluster centers. Cluster membership: $r_{nk} \in \{0, 1\}$

$$\begin{aligned} r_{nk} &= 1 \text{ if } x_n \text{ belongs to class } k \text{ (1-in-k coding)} \\ r_{nk} &= 0 \text{ otherwise} \end{aligned}$$

We can now introduce the objective function as follow

$$\min_{r_{nk}, \mu_k} J := \frac{1}{2} \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|_2^2$$

Our goal here is to minimize J to make sure that each x_n is close to its assigned μ_k. NP-hard problem
Algorithm to approximate the min:

1. Initialize with a random μ_k
2. Given μ_k, assign each x_n to closest μ_k
$$r_{nk} = \begin{cases} 1, & k = \operatorname{argmin} \|x_n - \mu_j\|^2 \\ 0 \end{cases}$$
3. Given r_{nk}, update the cluster centers
$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

Repeat this process until no further class reassignments happen. It can be shown that this converges but not necessarily to the global optimum, and it might be trapped in a local min.

§8 Lec 8: Jul 14, 2021

§8.1 Mixture of Gaussians

Let's first introduce a latent variable:

$$z \in \{0,1\}^k, \qquad \left(\sum_k z_k = 1\right) \text{ – 1-in-k code}$$

This latent variable describes the "inner" state of a data point (or of an observed x). We can now talk about the joint distribution. Let x be an observed data point and z be its associated latent variable.

$$p(x,z) = \underbrace{p(x|z)}_{\text{Gaussian}} \cdot p(z)$$

- Denote $p(z_k = 1) = \pi_k$, $\sum_{k=1}^k \pi_k = 1$, $\pi_k \in [0,1]$

$$\implies p(z) = \prod_{k=1}^K \pi_k^{z_k} \quad \text{– A compact notation}$$

- Class-conditional is Gaussian:

$$p(x|z_k = 1) := N(x|\mu_k, \Sigma_k)$$

Using the same trick as above, we have

$$p(x|z) = \prod_{k=1}^K N(x|\mu_k, \Sigma_k)^{z_k}$$

and the marginal is

$$\begin{aligned} p(x) &= \sum_z p(x,z) = \sum_z p(x|z)p(z) \\ &= \sum_{k=1}^K N(x|\mu_k, \Sigma_k)\,\pi_k \end{aligned}$$

From Bayes, we can find the likelihood of latent variable as follows

$$\begin{aligned} \gamma(z_n) = p(z_k = 1|x) &\frac{p(x|z_k = 1)p(z_k = 1)}{\underbrace{\sum_{j=1}^k p(x|z_j = 1)\,p(z_j = 1)}_{p(x)}} \\ &= \frac{\pi_k N(x|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x|\mu_j, \Sigma_j)} \end{aligned}$$

Question 8.1. So how do we learn π_k, μ_k, Σ_k?

→ Maximum Likelihood Estimation (MLE). Given a point cloud $X = \{x_1, \ldots, x_N\}$, $x_n \in \mathbb{R}^D$. As the joint are i.i.d., we have

$$\begin{aligned} p(X) &= \prod_{n=1}^N p(x_n) \\ &= \prod_{n=1}^N \sum_{k=1}^K \pi_k N(x_n|\mu_k, \Sigma_k) \end{aligned}$$

Note: When we take the natural logarithm, we obtain

$$\ln p(X) = \sum_{n=1}^{N} \ln \sum_{k=1}^{K} \pi_k N\left(x_n|\mu_k, \Sigma_k\right)$$

Observe that setting taking the derivative and set it equal to 0 (the usual approach) fails here as the problem becomes difficult and "labor-intensive", i.e.,

$$0 = \sum_{n=1}^{N} \frac{\pi_k N\left(x_n|\mu_k, \Sigma_k\right)}{\sum_j \pi_j N\left(x_n|\mu_j, \Sigma_j\right)} \Sigma_k^{-1}(x_n - \mu_k)$$

But recall from earlier when we define $\gamma(z)$,

$$0 = \sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k^{-1}(x_n - \mu_k)$$

Keeping $\gamma(z_{nk})$ fixed, we can solve for μ_k.

$$\mu_k = \sum_{n=1}^{N} \gamma(z_{nk}) x_n / \underbrace{\sum_{n=1}^{N} \gamma(z_{nk})}_{=N_k}$$

– weighted average of data points. Also,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^\top$$

Now, consider π_k s.t. $\sum \pi_k = 1$. Since this is a constraint optimization problem, we include Lagrange multiplier for this constraint in the optimization.

$$\pi_k = \frac{N_k}{N}$$

1. **Maximization Step**: Given $\gamma(z_{nk})$. We need to update
   - μ_k
   - Σ_k
   - π_k

   using MLE
2. **Expectation Step**: Given μ_k, Σ_k, π_k. We will update
   - $\gamma(z_{nk}) = p\left(z_k = 1|x_n\right) = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_j \pi_j N(x_n|\mu_j, \Sigma_j)}$

This is an example of Expectation-Maximization Algorithm on a Gaussian mixture model.

§8.2 General Expectation-Maximization Algorithm

First, we have

$$\begin{cases} \text{observed variables } X \\ \text{latent variables } z \\ \text{parameters } \theta \end{cases}, \text{ the joint: } p\left(X, z|\theta\right)$$

Goal: maximize $p(X|\theta)$ w.r.t θ. Let's get to the steps to solve this.

1. Initialize θ^{old} (estimate)
2. E-step: evaluate $p(z|X, \theta^{\text{old}})$
3. M-step: $\theta^{\text{new}} = \operatorname{argmax}_\theta Q\left(\theta|\theta^{\text{old}}\right)$ where
$$Q\left(\theta|\theta^{\text{old}}\right) := \sum_z p\left(z|X, \theta^{\text{old}}\right) \ln p\left(X, z|\theta\right)$$
4. $\theta^{\text{old}} \leftarrow \theta^{\text{new}}$ repeat if necessary.

Now, we're ready to delve into the derivation of the EM-algorithm. Let's start with the log-likelihood function.
$$L(\theta) := \ln p(X|\theta)$$
E-M is iterative: given θ^{old}, we want to find θ^{new} s.t. $L(\theta^{\text{new}}) > L(\theta^{\text{old}})$ which we can obtain by
$$\max_{\theta^{\text{new}}} L\left(\theta^{\text{new}}\right) - L\left(\theta^{\text{old}}\right) = \ln p\left(X|\theta^{\text{new}}\right) - \ln p\left(X|\theta^{\text{old}}\right)$$
The trick here is to include the latent variable z.
$$\begin{aligned}
p(X|\theta) &= \sum_z p(X, z|\theta) \\
&= \sum_z p\left(X|z, \theta\right) \cdot p(z|\theta) \\
L(\theta^{\text{new}}) - L(\theta^{\text{old}}) &= \ln \sum_z p\left(X|z, \theta^{\text{new}}\right) p\left(z|\theta^{\text{new}}\right) - \ln p\left(X|\theta^{\text{old}}\right) \\
&= \ln \sum_z p\left(z|X, \theta^{\text{old}}\right) \frac{p\left(X|z, \theta^{\text{new}}\right) p\left(z|\theta^{\text{new}}\right)}{p\left(z|X, \theta^{\text{old}}\right)} - \ln p\left(X|\theta^{\text{old}}\right)
\end{aligned}$$
Using Jensen's inequality, which is
$$\ln \sum \lambda_i x_i \geq \sum \lambda_i \ln x_i$$
provided that $\lambda_i \geq 0$ and $\sum \lambda_i = 0$.
$$\begin{aligned}
L(\theta^{\text{new}}) - L(\theta^{\text{old}}) &\geq \sum_z p\left(z|X, \theta^{\text{old}}\right) \left[\ln \frac{p\left(X|z, \theta^{\text{new}}\right) p\left(z|\theta^{\text{new}}\right)}{p\left(z|X, \theta^{\text{old}}\right)} - \ln p\left(X|\theta^{\text{old}}\right)\right] \\
&= \sum_z p\left(z|X, \theta^{\text{old}}\right) \ln \frac{p\left(X|z, \theta^{\text{new}}\right) p\left(z|\theta^{\text{new}}\right)}{p\left(z|X, \theta^{\text{old}}\right) p\left(X|\theta^{\text{old}}\right)} =: \Delta\left(\theta^{\text{new}}|\theta^{\text{old}}\right)
\end{aligned}$$
Thus,
$$\begin{aligned}
L(\theta^{\text{new}}) - L(\theta^{\text{old}}) &\geq \Delta\left(\theta^{\text{new}}|\theta^{\text{old}}\right) \\
L(\theta^{\text{new}}) &\geq \underbrace{L(\theta^{\text{old}}) + \Delta\left(\theta^{\text{new}}|\theta^{\text{old}}\right)}_{=:l(\theta^{\text{new}}|\theta^{\text{old}})}
\end{aligned}$$
$L\left(\theta^{\text{new}}\right)$ is bounded below by $l\left(\theta^{\text{new}}|\theta^{\text{old}}\right)$. The best guess to update θ is
$$\theta^{\text{new}} = \operatorname*{argmax}_\theta l\left(\theta|\theta^{\text{old}}\right) = \operatorname*{argmax}_\theta \left\{L(\theta^{\text{old}}) + \sum_z p\left(z|X, \theta^{\text{old}}\right) \ln \frac{p\left(X|z, \theta\right) p(z|\theta)}{p\left(z|X, \theta^{\text{old}}\right) p\left(X|\theta^{\text{old}}\right)}\right\}$$
which is equivalent to
$$\operatorname*{argmax}_\theta \sum_z p\left(z|X, \theta^{\text{old}}\right) \ln p\left(X|z, \theta\right) p(z|\theta)$$

or

$$\underbrace{\underset{\theta}{\operatorname{argmax}}}_{\text{M-step}} \underbrace{\sum_z p\left(z|X,\theta^{\text{old}}\right) \ln p\left(X,z|\theta\right)}_{=:Q\left(\theta|\theta^{\text{old}}\right)}$$

Convergence:

$$\left.\begin{array}{l}\theta^{\text{new}} \text{ maximizes } \Delta\left(\theta|\theta^{\text{old}}\right) \\ \Delta\left(\theta^{\text{old}}|\theta^{\text{old}}\right)=0\end{array}\right\} \implies \Delta\left(\theta^{\text{new}}|\theta^{\text{old}}\right) \geq 0$$

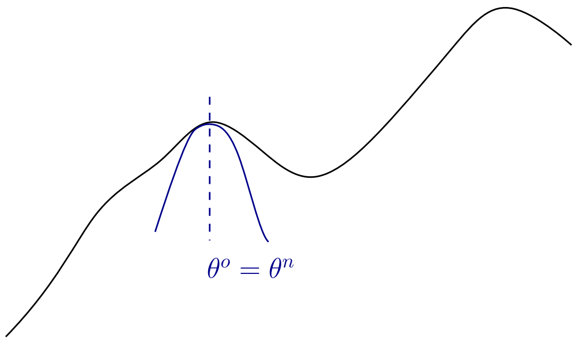
As a result,

$$L\left(\theta^{\text{new}}\right) \geq L\left(\theta^{\text{old}}\right) + \Delta\left(\theta^{\text{new}}|\theta^{\text{old}}\right)$$

Thus,

$$L\left(\theta^{\text{new}}\right) \geq L\left(\theta^{\text{old}}\right) \qquad \text{(non-decreasing)}$$

When θ reaches a fixed point on $l\left(\theta|\theta^{\text{old}}\right)$, we can only conclude that $\nabla L\left(\theta^{\text{old}}\right)=0$ (not necessarily reach global max, could be a local max or even just a local min or saddle point).



§9 Lec 9: Jul 15, 2021

§9.1 Kernel Methods – Linear Regression

Idea: Apply a linear machine learning model onto data after non-linear preprocessing thereof.

$$x_n \overset{\phi}{\mapsto} \phi(x_n) \qquad \mathbb{R}^D \to \mathbb{R}^E$$

We want to avoid isolated $\phi(x_n)$, but we want to have $\phi(x_n)^\top \phi(x_m)$ instead. We define the kernel function as follows

$$\phi(x_n)^\top \phi(x_m) \equiv k(x_n, x_m) \qquad \text{(kernel trick)}$$

Recall: regularized least squares loss function

$$J(w) \coloneqq \frac{1}{2}\sum_{n=1}^{N}\left(w^\top \phi(x_n) - t_n\right)^2 + \frac{\lambda}{2} w^\top w$$

As this is a non-constraint optimization problem, optimality requires:

$$\Delta_w J = \sum_{n=1}^{N}\left(w^\top \phi(x_n) - t_n\right)\phi(x_n)\lambda w \overset{\text{set}}{=} 0$$

Rearrange this a bit and we obtain

$$w = -\frac{1}{\lambda}\sum_{n=1}^{N}\left(w^\top \phi(x_n) - t_n\right)\phi(x_n)$$

Notice that w is a linear combination of the non-linear transform data $\phi(x_n)$.

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

in which the design matrix Φ is defined as

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_E(x_1) \\ \vdots & & \vdots \\ \phi_1(x_N) & \dots & \phi_E(x_N) \end{bmatrix}$$

Also,

$$a_n \coloneqq -\frac{1}{\lambda}\left(w^\top \phi(x_n) - t_n\right)$$

Substitute $w = \Phi^\top a$ into $J(w)$,

$$J(a) \coloneqq \frac{1}{2}\sum_{n=1}^{N}\left(a^\top \Phi \phi(x_n) - t_n\right)^2 + \frac{\lambda}{2} a^\top \Phi \Phi^\top a$$

which is equivalent to

$$J(a) = \frac{1}{2} a^\top \Phi\Phi^\top \Phi\Phi^\top a - a^\top \Phi\Phi^\top t + \frac{1}{2} t^\top t + \frac{\lambda}{2} a^\top \Phi\Phi^\top a$$

We introduce $K \coloneqq \Phi\Phi^\top =$ Gram matrix, $N \times N$ and symmetric.

$$K_{mn} = \phi(x_m)^\top \phi(x_n) = K(x_m, x_n) \qquad (K \succeq 0)$$

So, the loss (in terms of weights a) becomes

$$J(a) = \frac{1}{2}a^\top KKa - a^\top Kt + \frac{1}{2}t^\top t + \frac{\lambda}{2}a^\top Ka$$

The optimality requires

$$\begin{gathered}\nabla_a J = KKa - Kt + \lambda ka \overset{\text{set}}{=} 0 \\ K^{-1}KKa - K^{-1}Kt + \lambda K^{-1}Ka = 0 \quad (\text{as } K \succ 0) \\ Ka - t + \lambda a = 0 \\ (K + \lambda I)\, a = t \\ a = (K + \lambda I)^{-1} t\end{gathered}$$

If $E \to \infty$ then w has ∞-dim as well. We are not actually interested in w but in using w to make predictions for new data points $\tilde{x}$ and

$$\tilde{t} = w^\top \phi(\tilde{x}) = \phi(\tilde{x})^\top w = \underbrace{\phi(\tilde{x})^\top \Phi^\top}_{\vec{K}^\top} a = K^\top (K + \lambda I)^{-1} t$$

where K is the kernel with $\tilde{x}$ and each sample, $K_n = K(\tilde{x}, x_n)$.
The main difference between traditional linear regression and kernel-based linear regression:

- Traditional: use training data to learn optimal parameters (then discard data)
- Kernel (no parameters): the data are in charge which needs to say in memory as we need to evaluate $K(\tilde{x}, x_n)$.

§9.2 Kernel Construction

In principle:

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

In practice, we want to skip the ϕ-part. It's "easy" if we find a ϕ that results in k, then we have confirmation that k is indeed a kernel.

Example 9.1
$k(x, y) := \left(x^\top y\right)^2$, $x, y \in \mathbb{R}^2$

$$\begin{aligned}k(x, y) &= (x_1y_1 + x_2y_2)^2 = x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 \\ &= \begin{bmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{bmatrix}^\top \begin{bmatrix} y_1^2 \\ \sqrt{2}y_1y_2 \\ y_2^2 \end{bmatrix} = \phi(x)^\top \phi(y)\end{aligned}$$

with

$$\phi(z) := \begin{bmatrix} z_1^2 \\ \sqrt{2}z_1z_2 \\ z_2^2 \end{bmatrix}$$

Necessary and sufficient condition for $k(x, y)$ to be a kernel function: If for any $\{x_n\}$, $n = 1 \ldots N$, the Gram matrix K, $(K_{mn} = K(x_m, x_n))$, is positive semidefinite. Then $k(x, y)$ s a valid kernel (however, this is not a very practical rule).
From page 296 of the textbook, given k_1, k_2 valid kernels, then the following new kernels will also be valid (kernel lego)

- $ck_1(x, y)$, $c > 0$

- $f(x)k_1(x,y)f(y)$ for any f
- $q(k_1(x,y))$, q is a polynomial with non-negative coefficients
- $e^{k_1(x,y)}$
- $k_1(x,y)+k_2(x,y)$
- $k_1(x,y)\cdot k_2(x,y)$
- $k_1(\phi(x),\phi(y))$, $\phi:\mathbb{R}^D\to\mathbb{R}^M$
- $x^\top Ay$, A is symmetric positive semidefinite.

§9.3 Gaussian Processes – Linear Regression

Linear regression: linear combination of M fixed basis functions

$$y(x)=y=w^\top\phi(x),\qquad \phi\in\mathbb{R}^M$$

We will add a prior distribution on w

$$p(w)=N\left(w\middle|0,\frac{1}{\alpha}I\right)$$

For any given value w, $y(x)$ is a particular function. So the pdf over w defines a pdf over function $y(x)$. In practice, we evaluate $y(x)$ not on the entire $\mathbb{R}^D$ but only at discrete locations (e.g. $x_1,\dots,x_N,\tilde{x}$).
We're interested in the joint distribution $y(x_1),\dots,y(x_N)$. We write

$$\vec{y}:\ y_n\coloneqq y(x_n)$$

where $\vec{y}\in\mathbb{R}^N$.

$$\implies \vec{y}=\Phi w\qquad (\Phi_{nk}=\phi_k(x_n))$$

Question 9.1. What does the joint distribution of $\vec{y}$ look like?

It turns out $\vec{y}$ is multiple linear combination of Gaussian random variables $w=\begin{bmatrix}w_0\\ \vdots\\ w_{m-1}\end{bmatrix}$ which is Gaussian itself. To find the parameters, we compute sufficient statistics.

- $\mathbb{E}[y]=\mathbb{E}[\Phi w]=\Phi\mathbb{E}[w]=0$
- $\operatorname{cov}[\vec{y}]=\mathbb{E}\left[(\vec{y}-\mathbb{E}[\vec{y}])(\vec{y}-\mathbb{E}[\vec{y}])^\top\right]=\mathbb{E}\left[\vec{y}\vec{y}^\top\right]$

$$\begin{aligned}\mathbb{E}\left[\Phi ww^\top\Phi^\top\right]&=\Phi\mathbb{E}\left[ww^\top\right]\Phi^\top\\&=\Phi\operatorname{cov}[w]\Phi^\top=\frac{1}{\alpha}\Phi\Phi^\top=k\end{aligned}$$

So, the kernel is

$$k_{mn}=\frac{1}{\alpha}\phi(x_m)^\top\phi(x_n)=k(x_m,x_n)$$

Overall, what just happens is

$$\begin{aligned} w &\sim N\left(0, \frac{1}{\alpha} I\right) \\ &\downarrow \\ y(x) &= w^\top \phi(x) \\ &\downarrow \\ \vec{y} &= \begin{bmatrix} y(x_1) \\ \vdots \\ y(x_N) \end{bmatrix} \\ &\downarrow \\ \vec{y} &\sim N(0, k) \end{aligned}$$

Definition 9.2 (Gaussian Process) — Gaussian process is a pdf over function $y(x)$ s.t. the values evaluated at arbitrary $x_1, \ldots, x_N$ jointly have a Gaussian distribution.

Fundamental Property:

$$\mathbb{E}\left[y(x_n)y(x_m)\right] = k(x_n, x_m)$$

Now, let's get to the Gaussian processes for regression.

$$\text{Model}: \quad t_n = y_n + \varepsilon_n$$

where ε_n is the white noise and $\varepsilon_n \sim N\left(0, \sigma^2 (= \frac{1}{\beta})\right)$

$$\implies p\left(t_n | y_n, \sigma^2\right) = N\left(t_n | y_n, \sigma^2\right)$$

Then, we have

$$\begin{aligned} p\left(\vec{t} | \vec{y}\right) &= \prod_{n=1}^{N} p\left(t_n | y_n\right) \\ &= N\left(\vec{t} | \vec{y}, \sigma^2 I\right) \end{aligned}$$

From the definition of Gaussian process, we know

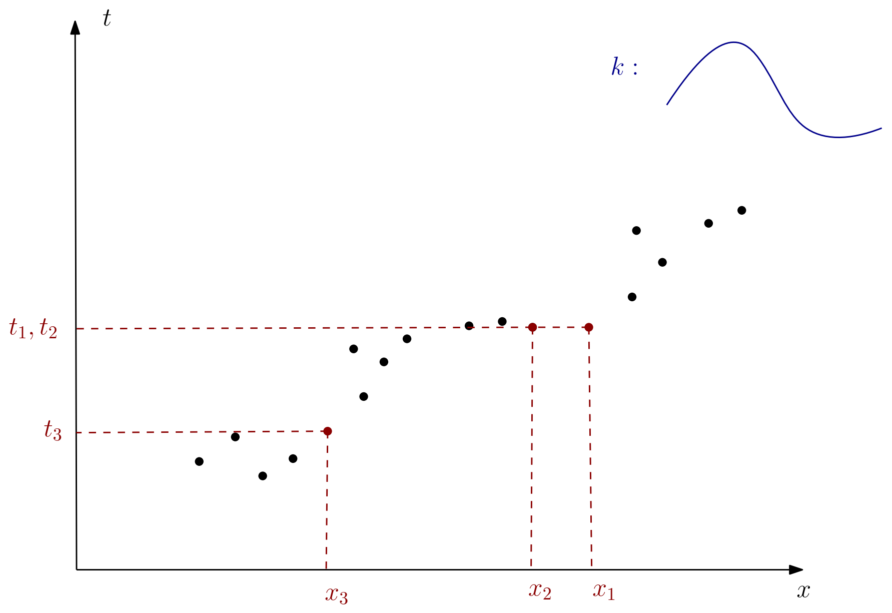
$$p(\vec{y}) = N\left(\vec{y} | 0, k\right)$$

From page 93 of the textbook, we can find the predictive distribution

$$p(\vec{t}) = \int p\left(\vec{t} | \vec{y}\right) P(\vec{y})\, d\vec{y} = N\left(\vec{t} | 0, k + \sigma^2 I\right)$$

We can evaluate the likelihood of a given data set as follows

1. $x_n \sim x_m$, i.e., $k(x_n, x_m)$ is large $\implies t_n \sim t_m$ or "penalty" (unlikely)
2. $x_n \nsim x_m$, i.e., $k(x_n, x_m)$ is small $\implies t_n \perp t_m$ (independent)



$k(x_1, x_2)$ is relatively large: $\exp(-\|x_1 - x_2\|^2)$
$\to$ strong correlation between t_1 & t_2 (need to be similar)

What's even more interesting is we can use this to make predictions, $p\left(\tilde{t}|\vec{t}\right)$ at $\tilde{x}$?

$$\vec{t}_{N+1} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \\ \tilde{t} \end{bmatrix}, \quad \vec{y}_{N+1} = \begin{bmatrix} y(x_1) \\ \vdots \\ y(x_N) \\ y(\tilde{x}) \end{bmatrix}$$

Then,

$$p\left(\vec{t}_{N+1}\right) = N\left(\vec{t}_{N+1}|0, C_{N+1}\right)$$
$$C_{N+1} = k_{n+1} + \sigma^2 I = \begin{bmatrix} C & k \\ k^\top & c \end{bmatrix}$$

where $k_n = k\left(\tilde{x}, x_n\right)$ and $c = k\left(\tilde{x}, \tilde{x}\right) + \sigma^2$. Then, we can use page 87 to get the marginal partition as follows

$$p\left(\tilde{t}|\vec{t}\right) = N\left(\tilde{t}|m\left(\tilde{x}\right), \sigma^2(\tilde{x})\right)$$
$$m(\tilde{x}) = k^\top C^{-1}\vec{t} = k^\top \left(k + \sigma^2 I\right)^{-1}\vec{t}$$
$$\sigma^2(\tilde{x}) = c - k^\top C^{-1} k$$