SML - Formelsammlung

Probability distributions

Normal distribution pdf: $\frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$; if $p(x,y)$ is gaussian, then conditional $p(x|y)$ and marginal $p(x)$ are also gaussian

Expectation: $E(X) = \int_{-\infty}^{\infty} x f(x)\,dx$, Variance: $E((X-\mu)^2) = E(X^2) - E(X)^2$

Marginal distribution: $f_X(x) = \int_{-\infty}^{\infty} f(x,y)\,dy$; Conditional distr. $f(x|y) = \frac{f(x,y)}{f_Y(y)}$

Moments: $m_n = E(x^n)$; Central Moments: $cm_n = E((x-\mu)^n)$; cm_2: variance, cm_3: skewness, cm_4: kurtosis

Bayes rule: $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$; posterior ∝ likelihood · prior; $p(x)$ = Normalization factor $= \sum_j p(x|C_j)p(C_j)$

KL-Divergence: $KL(p,q) = -\int p(x)\ln\frac{q(x)}{p(x)}dx$

Bayesian Decision Theory

Minimizing error: $p(\text{error}) = p(x\in R_1, C_2) + p(x\in R_2, C_1) = \int_{R_1} p(x|C_2)p(C_2)\,dx + \int_{R_2} p(x|C_1)p(C_1)$

Optimal decision: Decide for class C_1 if $p(C_1|x) > p(C_2|x)$ or $\frac{p(x|C_1)}{p(x|C_2)} > \frac{p(C_2)}{p(C_1)}$ (Likelihood-Ratio-Test) Bayes optimal classifier

Risk minimization: Loss function $\lambda(\alpha_i|C_j) = \lambda_{ij}$ with decision α_i and true class C_j

Expected Loss: $R(\alpha_i|x) = \sum_j \lambda(\alpha_i|C_j)\cdot p(C_j|x)$ ← class posterior

Decide for α_1 if $R(\alpha_2|x) > R(\alpha_1|x)$ or $\frac{p(x|C_1)}{p(x|C_2)} > \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})}\cdot\frac{p(C_2)}{p(C_1)}$

Probability Density Estimation

Is about finding the class conditional prob. $p(x|C_k)$

Parametric Models: Small number of parameters completely define pdf

Non-parametric: No explicit parameters (except hyperparameters) but every point is a parameter

Mixture Models: Combination of both

Maximum Likelihood:

PDF is given by parameters in θ: $p(x|\theta)$

We want to find θ that explains data the best

$L(\theta) = p(D|\theta) = p(x_1, x_2, \dots|\theta)$

$= p(x_1|\theta)\cdot p(x_2|\theta)\cdot\dots$ ⇒ i.i.d assumption

$= \prod_{i=1}^{N} p(x_i|\theta)$

or $\ln L(\theta) = \sum \ln p(x_i|\theta)$

to find maximum, derive w.r.t. θ, set 0 and solve

Does not work for single point, provides point estimates

Histograms:

Split data to bins, count datapoints in each bin

Problem: Doesn't scale well to high dimensions. Needs specified bin-width

Kernel-Density-Estimation:

Fix volume V and determine number of datapoints K

$p(x) \approx \frac{K}{NV} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h}k\left(\frac{\|x-x_n\|}{h}\right)$ with kernel k,

Problem: Need to choose kernel and bandwith, choice of kernel not so important for many datapoints

e.g. gaussian kernel $k(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}u^2\right)$

or parzen window $k(u) = \begin{cases}1 & |u| < 0.5\\ 0 & \text{otherwise}\end{cases}$

K-Nearest Neighbor:

Calculate distance to k-th neighbor. Higher distance means lower density

Mixture models:

$p(x) = \sum_{j=1}^{M} p(x|z_j)\cdot p(z_j)$ ← prior of cluster j

(see EM algorithm) ← conditional prob. of x given its in cluster j

Clustering

Unsupervised learning, unlabeled data

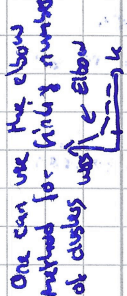
Goal: Find similar datapoints and put them in discrete set of clusters

K-means clustering:

1. Initialization, pick K centroids
2. Assign each datapoint to closest centroid
3. Adjust centroids to be cluster means
4. Back to step 2 until no change

Problems: Needs specified K, sensitive to initialization, only local optimum, only spherical clusters

One can use the elbow method for finding number of clusters (K at elbow)



Mean shift clustering:

Assign data to clusters by iteratively shifting datapoints to the mode (region with highest density) of a specified region. In the end, assign to points to the clust they converge to.

Problem: Needs specified window size, hard to compute

EM-Clustering:

Soft assignment to underlying prob. distribution

1. E-step: Assign each datapoint prob. of belonging to cluster j: $p(C_j|x) = \frac{p(x|C_j)\cdot p(C_j)}{\sum_{k=1}^{K} p(x|C_k)\cdot p(C_k)}$
2. M-step: Update priors, means and variances

$p(C_j)^{new} = \frac{1}{N}\sum_{i=1}^{N} p(C_j|x_i)$

$\mu_j^{new} = \sum_{i=1}^{N} p(C_j|x_i)\cdot x_i \;/\; \sum_{i=1}^{N} p(C_j|x_i)$

$\sigma_j^{new} = \sqrt{\sum_{i=1}^{N} p(C_j|x_i)\cdot(x_i-\mu_j)^2 \;/\; \sum_{i=1}^{N} p(C_j|x_i)}$

3. Repeat

Problems: Sensitive to initialization, only local optimum, needs specified number of mixture components

Evaluation

Occams Razor: choose smallest model that fits the data

Bias vs Variance: Too simple model has high bias, underfitting

Too complex model has high variance, overfitting

Maximum Likelihood Estimation is not always unbiased, e.g. Variance estimator of gaussian

Classification: Supervised learning approach with labeled data. Goal is to predict labels for new data

Fisher Discriminant Analysis (FDA)

Find vector w that maximizes the distance between the class means while minimizing the class variances

$$\max_w J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_w w}$$

Solution: $w \propto S_w^{-1}(m_1 - m_2)$

FDA does not classify, but makes it easier to classify

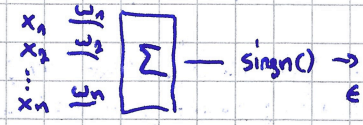
Linear Discriminant Functions: $y(x) = w^T x + w_0$ are hyperplanes that separate classes. They are useful if we only care about a decision boundary

Perceptron Algorithm

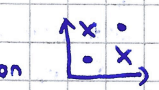
To find separating hyperplane

$y(x) = \text{sign}(w^T x + b)$

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$



Fails at the XOR function



Data can sometimes be transformed to other feature spaces to make it linearly separable

---

Logistic regression

$y_i \in \{0, 1\}$

$$p(C_1|x) = \frac{p(x|C_1)p(C_1)}{p(x)} = \frac{\exp(w^T x + w_0)}{1 + \exp(w^T x + w_0)} = \sigma(\cdot)$$

Now code as Bernoulli experiment and maximize Likelihood

$$L(w) = \prod_{i=1}^N \sigma(w^T x + w_0)^{y_i} \cdot (1 - \sigma(w^T x + w_0))^{1-y_i}$$

Regression

$X^T w + \epsilon = y$ with $X = (x_1, x_2, \ldots x_n)$

$\Rightarrow w = (XX^T)^{-1} X y$ after $\min \sum_{i=1}^N (x_i^T w - y)^2$

or with features $\phi(x) = (1, x, x^2, \ldots)$ and Φ

Maximum-Likelihood approach:

Assume $\epsilon \sim N(0, \sigma^2) \Rightarrow y_i \sim N(x_i^T w, \sigma^2)$

$p(y|X) = \prod_{i=1}^N N(x_i^T w, \sigma^2)$, max w.r.t w

We also find variance σ^2

Bayesian (Ridge) Regression

Put prior on w, $p(w) \sim N(0, \sigma_0^2)$

$$p(w|X,y) \propto p(y|X,w) \cdot p(w) \quad \Rightarrow w_{MAP} = (XX^T + \frac{\sigma^2}{\sigma_0^2})^{-1} X y$$

Posterior ∝ Likelihood · Prior

Same result from regularization $w = \min_w \frac{1}{2} \|X^T w - y\|^2 + \frac{1}{2}\lambda \|w\|^2$

Kernel methods

Use kernel instead of high dimensional features, because $k(x, x') = \phi(x)^T \phi(x')$

Kernel Ridge Regression: Predict y for test input $x_\star$

$$\hat{y}(x_\star) = y^T (K(X,X) + n\lambda I)^{-1} K(X, x_\star)$$

with

$$K(x_\star, X) = \begin{pmatrix} k(x_\star, x_1) \\ k(x_\star, x_2) \\ \vdots \end{pmatrix}; \quad K(X,X) = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_N) \\ k(x_2,x_1) & \cdots & \\ \vdots & & \\ k(x_N,x_1) & \cdots & k(x_N,x_N) \end{pmatrix}$$

Gram Matrix

Problem: Inverting $N \times N$ Gram matrix for large datasets

Bayesian linear Regression

Only predict new data with $\hat{y} \sim N(\mu_\star, \sigma_\star^2)$

$$\mu_\star = \phi(x_\star)^T \left(\frac{\sigma^2}{\sigma_0^2} I + \Phi^T \Phi\right)^{-1} \Phi^T y$$

$$\sigma_\star^2 = \frac{\sigma^2}{\sigma^2} + \phi(x_\star)^T \left(\frac{1}{\sigma_0^2} I + \frac{1}{\sigma^2} \Phi^T \Phi\right)^{-1} \phi(x_\star)$$

Gaussian Process Regression

Find posterior predictive dist. $\hat{y} \sim N(\mu_\star, \sigma_\star^2)$

$$\mu_\star = K(X, x_\star)^T (\sigma^2 I + K(X,X))^{-1} y$$

$$\sigma_\star^2 = k(x_\star, x_\star) - K(X, x_\star)^T (\sigma^2 I + K(X,X))^{-1} K(X, x_\star)$$

Latent representations

Linear Dimensionality Reduction

Find mapping $x_n \to z_n$ as $z_n = B x_n$ $B \in \mathbb{R}^{M \times D}$

$\dim D \to \dim M$

So that data keeps as much variation as possible.

Compute data Covariance Matrix, compute Eigenvectors, order by size of Eigenvalue. ~~There are how~~

Choose M-Number of largest eigenvectors, put in B

Eigenvalue/sum(Eigenvalues) shows explained variance

Problem of choosing M: Choose smallest M that "works", choose M so that $x\%$ of variance is maintained

Expectation maximization

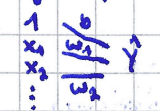
Data is generated by a (hidden) class. See Clustering

Neural networks (NN)

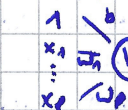
Shift to neural networks: Too little data → Too much data

Features initiliye → Features hard to get

Single layer NN (No hidden layer)



or with activation function

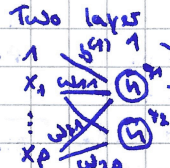


with h = logistic function

$\Leftrightarrow y(x) = w^T x + w_0$

$\Leftrightarrow y(x) = \sigma(w^T x + w_0)$

Two layers



$$\hat{y} = w_1^{(2)} q_1 + w_2^{(2)} q_2 + \ldots + b$$

$$q_1 = h(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + \ldots + b_1^{(1)})$$

$$q_2 = h(w_{21}^{(1)} x_1 + \ldots + b_2^{(1)})$$

or $q = h(W^{(1)} x + b^{(1)})$, $\hat{y} = W^{(2)} q + b^{(2)}$

2-layer-NN can represent any continuous function with enough neurons

But deep NN (>2 layers) can do it more efficiently.

Activation functions:

Relu $f(x) = \max(0, x)$

Sigmoid $f(x) = \sigma(x) = \frac{\exp(x)}{1+\exp(x)}$

Softplus $f(x) = \ln(1 + e^x)$