Lecture 2

$Q(y, f(x)) \hat{=}$ loss between label y and estimate $f(x)$

Conditional expected risk (given x): $R(f,x) = \int Q(y, f(x)) P(y|x)\, dy$

Total expected risk (for x, y): $R(f) = \mathbb{E}_x[R(f,x)] = \int_x R(f,x) P(x)\, dx$

$$= \int_y \int_x Q(y, f(x)) P(x,y)\, dx\, dy$$

Empirical risk on training data $z^{train} = \{(x_1, y_1), \dots (x_n, y_n)\}$

test data $z^{test} = \{(x_{n+1}, y_{n+1}), \dots, (x_{n+m}, y_{n+m})\}$

Training error $\hat{R}(\hat{f}, z^{train})$ for Empirical Risk Minimizer $\hat{f}$ (ERM)

↳ select ERM $\hat{f} \in \arg\min_{f \in \mathcal{C}} \hat{R}(f, z^{train})$

↳ training error $\hat{R}(\hat{f}, z^{train}) = \frac{1}{n} \sum_{i=1}^{n} Q(y_i, \hat{f}(x_i))$

↳ test error $\hat{R}(\hat{f}, z^{test}) = \frac{1}{m} \sum_{i=n+1}^{n+m} Q(y_i, \hat{f}(x_i))$

$\hat{R}(\hat{f}, z^{test}) \neq \mathbb{E}_x[R(\hat{f}, x)]$, as we can not guarantee that test set has the "real" distribution of X.

Taxonomy of Data

Measurement: Given an object set, a measurement X maps an object set into a domain $\mathbb{K}$

$$X : O^{(1)} \times \dots \times O^{(R)} \to \mathbb{K}$$
$$(o_1 \dots o_R) \longmapsto x_{o_1 \dots o_R}$$

monadic data: $X : O \to \mathbb{R}^d$

dyadic data: $X : O^{(1)} \times O^{(2)} \to \mathbb{R}$, pairwise if $O^{(1)}$ is same space as $O^{(2)}$

polyadic data: $O^{(1)} \times O^{(2)} \times O^{(3)} \to \mathbb{R}$

		+ transformation invariances
		f is bijective
Nominal scale	categories/stand on own	
Ordinal scale	only meaning that in comparison with other samples e.g. rank	$f(x_1) < f(x_2)$ for $\forall x_1 < x_2$
interval scale	difference carries the information (Fahrenheit)	$ax + c$ $a \in \mathbb{R}^+$
ratio scale	zero carries information, but not measurement unit (Kelvin)	ax $a \in \mathbb{R}^+$
absolute scale	absolute values are meaningful (~~trade~~ grades or exam)	Identity

Probability Spaces

Elementary event : $\omega_1 \dots \omega_N$ are samples points @ one coin flip

Sample space : $\Omega = \{\omega_1, \dots, \omega_N\}$ ~~e.g.~~ N ~~variations of~~ coin-flips

Family of sets : event A is a set of elementary events with
↳ $A \subset \Omega$
↳ $\omega \in A$ or $\omega \notin A$

Algebra of events: $\mathcal{A}$ is algebra of events, for $A \subset \Omega$
↳ $\mathcal{A} \; \Omega \in \mathcal{A}$
↳ if $A \in \mathcal{A} \wedge B \in \mathcal{A}$
then $A \cup B \in \mathcal{A} \wedge A \cap B \in \mathcal{A} \wedge A \setminus B \in \mathcal{A}$

Probability of events: Assign weights $p(\omega_i)$ to all $\omega \in \Omega$ with
$0 \leq p(\omega_i) \leq 1$
$\sum_i p(\omega_i) = 1$

Probability of an event: $A \in \mathcal{A}$ with $P(A) = \sum_{i : \omega_i \in A} p(\omega_i)$

Probability model : $(\Omega, \mathcal{A}, P)$

Lecture 3

Bayes Rule: $P(\text{model}|\text{data}) = \frac{P(\text{data}|\text{model})\,P(\text{model})}{P(\text{data})} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$ (posterior)

Assume model $Y = f(X,\theta) + \epsilon$, ~~$Y$=output~~

- Bayesian view: X, Y, θ are random variables
- Parametric statistics: functional form of $P(X,Y|\theta)$ is given, now estimate θ of $P(\text{data}|\text{model})$
- Non-parametric statistics: sample X, Y to estimate $P(\text{data}|\text{model})$
- statistical learning theory: minimize empirical risk $\min_{f \in \mathcal{C}} \hat{R}(f, Z^{train})$, don't estimate $p(\text{data}|\text{model})$

Bayesian-hood to model data (not assumptions on data model)
- evidence is Huge to compute
- parameters are rand. variables
- output is a distribution, not a value

Frequentism:
- Maximal likelihood ✓
- Fisher information ≙ measure for information content of densities: $I_n(\theta^*)$
- sampling theory
- hypothesis testing

Maximal likelihood method:
1. Define a parametric model
2. Define a likelihood as a funct. of the parametric model
3. Compute an estimator by maximizing the likelihood

ML is consistent: $\theta_{ML} \to \theta^*$ in probability as $n \to \infty$

asymptotically normal: $\frac{1}{\sqrt{n}}(\theta_{ML} - \theta^*)$ converges in distribution to a random variable with distribution $N(0, J^{-1}(\theta) I(\theta) J^{-1}(\theta))$

asymptotically efficient: θ_{ML} minimizes $\mathbb{E}[(\theta_{ML} - \theta^*)^2]$ as $n \to \infty$

Rao-Cramer bound: $\mathbb{E}[(\hat{\theta} - \theta^*)^2] \geq \frac{1}{I_n(\theta^*)}$, for any unbiased estimator $\hat{\theta}$

↳ ML equalizes Rao-Cramer bound for $n \to \infty$ ≙ asymptotically efficient

Bayesian methods
- Allow priors
- Provides a distribution of parameters when estimating
- requires efficient tricks when estimating posterior
- Prior often induces regularization (interpretation)

Frequentist methods
- No priors
- provides a single parameter when estimating
- requires only differentiation

$\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta^*$ Left out: Rao-Cramer derivation, converse of ML

ML is easy, unbiased, efficient to θ^*

Bayesian Learning

Parameter distribution: θ is considered as a random variable with $p(\theta|\mathcal{X})$ $\mathcal{X}$ ≙ sample set

Assumption: $p(x)$ is unknown ($X \sim p(x)$), $p(x|\theta)$ is known

"wanted": $p(x=x|\mathcal{X}) = \int p(x|\theta)\, p(\theta|\mathcal{X})\, d\theta$

Approximation: $p(\theta|\mathcal{X}) \sim \delta(\theta-\hat{\theta}) \Rightarrow p(x|\mathcal{X}) \approx p(x|\hat{\theta})$

Recursive Bayesian Estimation — also is an online algorithm

Assumption: Data are available in sequential order e.g. $\mathcal{X}^n = (x_n, x_{n-1} \dots x_1)$ are used one after another

likelihood of data: $p(\mathcal{X}^n|\theta) = p(x_n|\theta)\, \overbrace{p(\mathcal{X}^{n-1}|\theta)}^{\text{similar}}$

posterior: $p(\theta|\mathcal{X}^n) = \frac{p(x_n|\theta)\, p(\theta|\mathcal{X}^{n-1})}{\int p(x_n|\theta)\, p(\theta|\mathcal{X}^{n-1})\, d\theta}$ ← similar

prior: $p(\theta|\mathcal{X}^0) = p(\theta)$

Lecture 4

- (Parametric) maximum likelihood: Assume $Y|X \sim N(f(X), \sigma^2 I)$

  solve: $\arg\max_f \sum_{i=1}^{n} \log P(Y = y_i \mid X = x_i, \sigma^2)$,

- statistical learning theory: $\arg\min_f \sum_{i=1}^{n} (y_i - f(x_i))^2$ ——— same f for linear regression

Linear regression model: $Y = \beta_0 + \sum_{j=1}^{d} X_j \beta_j + \varepsilon$ X = data, Y = output, β = params, ε = noise $\sim N(0, \sigma^2)$

Residual sum of squares: $RSS(\beta) = (y - X\beta)^T (y - X\beta)$, $\hat{\beta} = (X^T X)^{-1} X^T y$

Distribution of $\hat{\beta}$: $\hat{\beta} \sim N(\beta, (X^T X)^{-1} \sigma^2)$

~~y without noise~~

$$\hat{\beta} = (X^T X)^{-1} X^T y = (X^T X)^{-1} X^T (X\beta + \varepsilon), \quad \varepsilon \text{ being the noise vector}$$

$$= \beta + (X^T X)^{-1} X^T \varepsilon \qquad \text{only random variable, 0 in mean}$$

$$\mathbb{E}_{y|X}[\hat{\beta}] = \beta + (X^T X)^{-1} X \, \mathbb{E}_{y|X}[\varepsilon] = \beta$$

$$Cov[\hat{\beta}] = \mathbb{E}[(\hat{\beta} - \beta)(\hat{\beta} - \beta)^T]$$

$$= \mathbb{E}[(X^T X)^{-1} X^T \varepsilon \varepsilon^T X (X^T X)^{-1}]$$

$$= (X^T X)^{-1} X^T \, \mathbb{E}[\varepsilon \varepsilon^T] X (X^T X)^{-1}$$

$$= \sigma^2 (X^T X)^{-1} \underbrace{X^T X}_{\text{scalar}} (X^T X)^{-1} = \sigma^2 I$$

Optimality of Least Squares Estimates:

assume we want to find some $k = a^T \beta$, we only have $\hat{k} = a^T \hat{\beta}$

$$\mathbb{E}[a^T \hat{\beta}] = \mathbb{E}[a^T (X^T X)^{-1} X^T y] = a^T (X^T X)^{-1} X^T (X\beta + \underbrace{\mathbb{E}[\varepsilon]}_{=0}) = a^T \beta$$

$\hat{\beta}$ is unbiased

$$Var[a^T \hat{\beta}] = Var[a^T (X^T X)^{-1} X^T \underbrace{(X\beta + \varepsilon)}_{\text{no variance}}] = Var[a^T (X^T X)^{-1} X \varepsilon]$$

$$= \mathbb{E}[a^T (X^T X)^{-1} X^T \varepsilon \varepsilon^T X (X^T X^{-1}) a] = \sigma^2 a^T (X^T X)^{-1} a$$

We know define $q = c^T y = a^T \hat{\beta} + a^T D y = a^T ((X^T X)^{-1} X + D) y = a^T \hat{\beta}$

q unbiased as seen above

$$\mathbb{E}[a^T c^T y] = a^T \mathbb{E}[\hat{\beta}] + a^T D \mathbb{E}[y]$$

$a^T c^T y$ unbiased

$$= a^T \beta + a^T D X \beta + a^T D \underbrace{\mathbb{E}[\varepsilon]}_{=0} = a^T \beta$$

$\Rightarrow DX = 0$

Gauss-Markov Theorem – continuation of Optimality

For any linear estimator $q = c^T y$ that is unbiased for $a^T\beta$ we have

$\mathrm{Var}[a^T\hat\beta] \le \mathrm{Var}[c^T y]$

$\mathrm{Var}[c^T y] = a^T c^T \mathrm{Cov}[y] c a = a^T c^T \sigma^2 I c a$

$= \sigma^2 a^T ((X^TX)^{-1}X^T + D)(X(X^TX)^{-1} + D^T) a$

$= \sigma^2 a^T ((X^TX)^{-1}X^TX(X^TX)^{-1} + DD^T) a$ cross-terms get eliminated because $DX = 0$

$= \sigma^2 a^T (X^TX)^{-1} a + \sigma^2 \|D^T a\|^2$

$= a^T \mathrm{Cov}(\hat\beta)\, a + m \geq 0 \geq \mathrm{Var}[a^T\hat\beta]$ □

Claim: the least squares estimate of β has the smallest variance among all linear unbiased estimates

Bias-variance trade-off ≙ trade bias for variance reduction

Mean squared error = bias² + variance + noise variance

$$\mathbb{E}_{xy}[(y - \hat f(x^*))^2] = (\mathbb{E}[y|x=x^*] - \mathbb{E}_\mathcal{L}[\hat f(x^*)])^2 + \mathbb{E}_\mathcal{L}[(\mathbb{E}_\mathcal{L}[\hat f(x^*)] - \hat f(x^*))^2] + \mathbb{E}_y[y - \mathbb{E}_{y|x}[y|x=x^*])^2 | x=x^*]$$

Good exercise

Small data set, large set of functions ⇒ Variance large, bias small

overfitting!

Shrinkage / Regularization

can be combined to Elastic Net

added

Ridge regression

$\min_\beta \sum_i (y_i - x_i^T\beta)^2 + \lambda\|\beta\|^2$

+ Protection against multicollinearity

Bayes: prior $\beta = N(0, \frac{\sigma^2}{\lambda} I)$

Solution: $\hat\beta_r = (X^TX + \lambda I)^{-1} X^T y$

Lasso

$\min_\beta \sum_i (y_i - x_i^T\beta)^2 + \lambda\|\beta\|_1$

+ sparse ⇒ interpretable

Bayes prior $\beta_i = \frac{\lambda}{4\sigma^2}\exp(-|\beta_i|\frac{\lambda}{2\sigma^2})$

Solution LARS

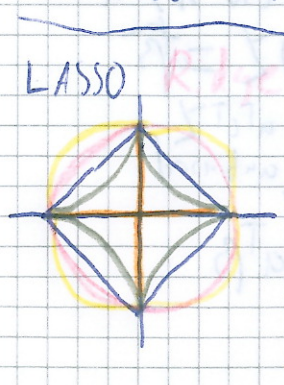
multicollinearity: $\hat\beta = \beta + (X^TX)^{-1}X^T\xi$

normal

can be huge

Ridge $\hat\beta = \beta + (X^TX + \lambda I)^{-1} X\xi$

$\lambda = 0$ + be huge



for $\hat\beta = \arg\min_\beta \sum_{i=1}^n (y_i - \beta_0 - \sum_{j=1}^d x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^d |\beta_j|^q$

Lecture 5

Bayesian:

$(y = X^T\beta + \varepsilon$ with $\varepsilon \sim N(\varepsilon|0,\delta^2)) \hat{=} N(y|X^T\beta, \delta^2)$

Add Ridge Prior $\hat{=} p(\beta|\Delta) = N_d(\beta|0, \Delta^{-1})$

Model inversion: assume $p(\beta|X, y, \Delta) = N(\beta|\mu_\beta, \Sigma_\beta)$

then $\mu_\beta = (X^TX + \delta^2\Delta)^{-1}X^Ty$

$\Sigma_\beta = \delta^2(X^TX + \delta^2\Delta)^{-1}$

Moments of Bayesian linear regression: of $y = X\beta + \varepsilon$, $\varepsilon \sim N(\varepsilon|0, \delta I_n)$

$p(\beta|\Delta) = N_d(\beta|0, \Delta^{-1})$

$\mathbb{E}_{\beta,\varepsilon}[y] = 0$

$\mathrm{cov}[y] = \mathbb{E}_{\beta,\varepsilon}[(X\beta+\varepsilon)(X\beta+\varepsilon)^T] = X\Delta^{-1}X^T + \delta^2 I_n$

$\mathrm{cov}[y] = \lambda^{-1}(XX^T + \lambda\delta^2 I_n)$ if $\Delta = \lambda I_n$

Moments of joint Gaussian: $\mathbb{E}[y] = 0$, $\mathrm{cov}[y] = X\Delta^{-1}X^T + \delta^2 I_n$

$$\downarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \sim N\left(y \,\middle|\, 0, \begin{bmatrix} k_{1,1}+\delta^2 & k_{1,2} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2}+\delta^2 & \cdots & k_{2,n} \\ \vdots & \vdots & \ddots & \\ k_{n,1} & k_{n,2} & \cdots & k_{n,n}+\delta^2 \end{bmatrix}\right)$$

with $k_{i,j} = k(x_i, x_j) = x_i^T\Delta^{-1}x_j$

$k(\cdot,\cdot) \hat{=}$ kernel function

Kernelized linear regression $\hat{=}$ Gaussian Process

Good kernels specify the degree of similarity between any two data points.

more similar $\hat{=}$ higher covariance $\hat{=}$ less ~~about~~ axis aligned ellipse

Kernels

Properties: (Mercer's theorem)

I Symmetry: $k(x,y) = k(y,x)$

II positive semi-definiteness: $\int k(x,y) f(x) f(y)\, dx\, dy \geq 0 \quad \forall f \in L_2$ ~~(Mercer's theorem)~~

or Gram matrix $K = \begin{pmatrix} k(x_1,x_1) & \cdots & k(x_1,x_n) \\ \vdots & & \\ k(x_n,x_1) & & k(x_n,x_n) \end{pmatrix}$ is ~~set~~ positive semi-definite

$\Downarrow$

$x^T K x \geq 0$, for $\forall x$

Examples:

linear	$x^T y$	differentiate
poly	$(x^T y + 1)^p$ for $p \in \mathbb{N}$	differentiate
Gauss/RBF	$\exp(-\|x-y\|^2/h^2)$	
Sigmoid/tanh	$\tanh(\kappa x^T y - b)$	

Kernel engineering, kernels can be combined in the following ways to result in another kernel:

- addition $k_1(x,y) + k_2(x,y)$ ~ also holds if kernels depend on x or y
- multiplication $k_1(x,y) \cdot k_2(x,y)$
- scaling $k(x,y) \cdot c$ with $c > 0$
- composition $f(k_1(x,y))$ where f is a poly with positive coefficients or exp

Prediction by Gaussian processes

General: Conditional Gaussian distributions

$\hookrightarrow p\left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right)$ with $a, U \in \mathbb{R}^x$, $\Sigma \in \mathbb{R}^{x,x}$, p.s.d

then $p(u_2 \mid u_1 = z) = \mathcal{N}(u_2 \mid U_2 + \Sigma_{21}\Sigma_{11}^{-1}(z - U_1),\ \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12})$

Gaussian Process prediction: $p(y_{n+1} \mid x_{n+1}, X, y) = \mathcal{N}(y_{n+1} \mid \mu_{y_{n+1}}, \delta^2_{y_{n+1}})$

with $\mu_{y_{n+1}} = k^T (k + \delta^2 I)^{-1} y \quad , k = k(x_{n+1}, X)$

$\delta^2_{y_{n+1}} = k(x_{n+1}, x_{n+1}) + \delta^2 - k^T (k + \delta I)^{-1} k$

Combining Regressors

B estimators, simple average, $\hat{f}(x) = \frac{1}{B}\sum_{i=1}^{B} \hat{f}_i(x)$

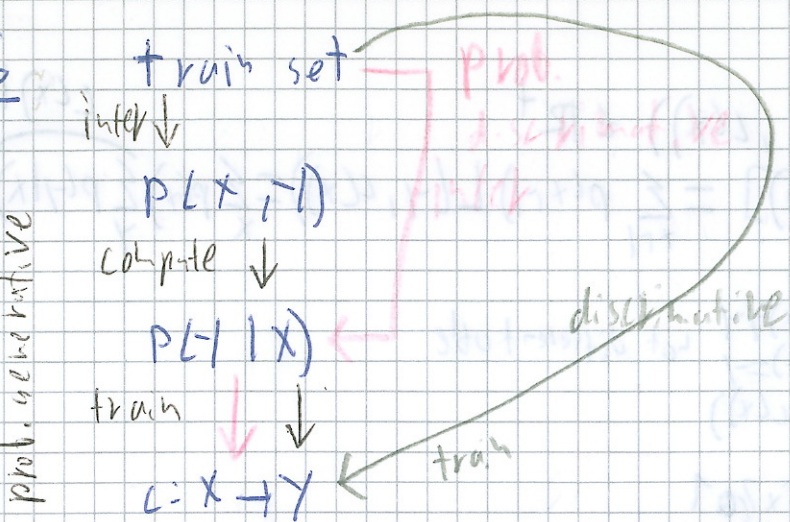
Bias: $\text{bias}[\hat{f}(x)] = \frac{1}{B}\sum_{i=1}^{B} \text{bias}[\hat{f}_i(x)]$ $\Rightarrow$ unbiased learners unbiased

Variance: $\text{Var}[\hat{f}(x)] = \frac{1}{B^2}\sum_{i=1}^{B} \text{Var}_D[\hat{f}_i(x)] + \frac{1}{B^2}\sum_{i \neq j}\sum \text{Cov}[\hat{f}_i(x), \hat{f}_j(x)]$

$\hookrightarrow$ if $\text{cov} \approx 0 \rightarrow$ Variance goes down by $\frac{1}{B}$

Good ensemble

Lecture 6



infer $p(x,y) \Rightarrow$ better understanding, new synthetic samples, outlier detection

more steps $\Rightarrow$ more bias, and harder

| comp. to $p(y|x) \Rightarrow$ degrees of belief

fitting $p(x,y)$: I guess a fam. of parametrized probabilistic models — done in practice

cause → effect

II infer θ with $\arg\max_\theta \log p(x_{train}, y_{train}|\theta)$

$= \arg\max_\theta \log p(y_{train}|\theta) \cdot p(x_{train}|y_{train}, \theta)$

compute $p(y|x)$, with $p(x,y) = p(y)p(x|y) = p(y)\mathcal{N}(x|\mu_y, \Sigma_y)$

↳ linear discriminate analysis (LDA) iff $\Sigma_0 = \Sigma_1 = \Sigma$, classes have same size

$$p(1|x) = \frac{p(x|1)p(1)}{p(x|1)p(1) + p(x|0)p(0)} \qquad 0 \hat{=} \text{class } 0,\ 1 \hat{=} \text{class } 1$$

$$p(1|x) = \frac{1}{1 + \frac{p(x|0)p(0)}{p(x|1)p(1)}} = \frac{1}{1 + \exp\left(-\log \frac{p(x|1)p(1)}{p(x|0)p(0)}\right)} = \delta(w^T x + w_0)$$

$$p(x|i) = |2\pi\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T \Sigma^{-1}(x-\mu_i)\right) \hat{=} \text{Normal distribution}$$

$$\log p(x|i) = -\frac{1}{2}\log|2\pi\Sigma| - \frac{1}{2}x^T\Sigma^{-1}x + x^T\Sigma^{-1}\mu_i - \frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i$$

$$\log \frac{p(x|1)p(1)}{p(x|0)p(0)} = \log(x|1) - \log(x|0) + \frac{\log(1)}{\log(0)}$$

$$= \underbrace{x^T\Sigma^{-1}(\mu_1 - \mu_0)}_{w} \underbrace{-\frac{1}{2}(\mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0)}_{w_0} + \frac{\log p(1)}{\log p(0)}$$

↳ generalized quadratic discriminant $\hat{=} \Sigma_0 \neq \Sigma_1 \hat{=} \frac{1}{2}x^T\Sigma_1 x \neq \frac{1}{2}x^T\Sigma_0 x$

$$\log \frac{p(x|1)p(1)}{p(x|0)p(0)} = x^T W x + w^T x + w_0 \Rightarrow p(i|x) = \delta(x^T W x + w^T x + w_0)$$

Bayes decision theory

I Define loss: $L(y, c(x)) \in \mathbb{R}^+$

II $\min_c \mathbb{E}_{x,y}[L(y, c(x))] = \sum_{x,y} p(x,y) L(y, c(x)) = \sum_x p(x) \overbrace{\sum_y p(y|x) L(y, c(x))}^{c(x)|x}$

Example losses

0-1 loss: $\begin{cases} 1 & \text{if } c(x) \neq y \\ 0 & \text{if } c(x) = y \end{cases}$ not differentiable

exponential loss: $\exp^{-y\, c(x)}$

Hinge: $\begin{cases} 0 & \text{if } -y w^T x \geq 0 \\ -y w^T x & \text{else} \end{cases}$

Perception loss: $\begin{cases} 0 & \text{if } -y w^T x \geq 0 \\ -y w^T x & \text{else} \end{cases}$

Maximum Likelihood estimation

best weights $\leq \arg\max_w$ log-likelihood training set $P(w)$

$\exp(LL(w)) = p(\text{train}|w) = \prod_{i \leq n} p(x_i, y_i | w) = \prod_{i \leq n} p(y_i | x_i, w)\, p(x_i | w)$

if Training set $\downarrow p(w)$

$= \text{const} \prod_{i \leq n} \delta(w^T x_i)^{y_i} (1 - \delta(w^T x_i))^{1-y_i}$

$p(1|x)$

$L(y)$ is analytically intractable, but differentiable $\Rightarrow$ do gradient descent

Gradient descent

$NL = -L$

$k = 0$

$w^{(k)} = \text{random}$

while $\|\eta(k) \nabla NL(w^{(k)})\| \geq \varepsilon$:

$w^{(k+1)} = w^{(k)} - \eta(k) \nabla NL(w^{(k)})$

$k += 1$

$\text{Good}\,\eta(k) = \dfrac{\|\nabla NL(w^{(k)})\|^2}{\nabla NL(w^{(k)})^T \nabla^2 NL(w^{(k)}) \nabla NL(w^{(k)})}$

derived using Taylor expansion and $\eta(k) = w^{(k+1)} - w^{(k)}$ → exercise

Newton's method

better updates — no learn rate/polarity

$w^{(k+1)} = w^{(k)} - \nabla^2 NL(w^{(k)})^{-1} \nabla NL(w^{(k)})$ found by deriving Taylor on $w^{(k+1)}$

$\nabla^2 NL(w^{(k)})^{-1}$ is hard to compute

Lecture 7 & 8

Perceptron did not distinguish between classifiers, when all samples are correct (hard)
↳ SVM wants max-margin

Slater's condition: $g_i(w)=0$, $i\le k$, $h_j(w)<0$, $j\le m$ $\Rightarrow$ strong duality

Lagrangian: $L(w,\lambda,\alpha)=f(w)+\sum_i \lambda_i g_i(w)+\sum_j \alpha_j h_j(w)$, $\alpha\ge 0$ (inequality for SVM)

complementary slackness: $\alpha_j^* h_j(w^*)=0$, $w^*=\min_w L(w,\lambda^*,\alpha^*)$
↳ true if strong duality (Slater's)

Constrained convex optimization via the dual

$\min_w f(w)$ s.t. $g_i(w)=0$, $h_j(w)\le 0$
- e.g. min distance
- e.g. stay on ground
- e.g. stay in circle

SVM: or ~~$L(w,\lambda,\alpha)$~~, $\frac{1}{2}\|w\|^2$ or ~~or~~ or $1-y_i(w^T x_i+w_0)$ (see SVM formulation)

↳ ~~dual~~ strong if Slater's holds

$\theta(\lambda,\alpha)\triangleq \min_w L(w,\lambda,\alpha)$ → easy with complementary slackness

$\max_{\lambda,\alpha} \theta(\lambda,\alpha)$ s.t. $\alpha_j\ge 0$

SVM – Maximum-margin classifier

x^+, x^- closest to border, proj. $\triangleq$ Projection

$\max_{w,w_0} 2m(w,w_0) = \|\text{proj}_w x^+ - \text{proj}_w x^-\| = \frac{2}{\|w\|}$ (margin)

$= \left\| \frac{w^T x^+}{\|w\|^2} w - \frac{w^T x^-}{\|w\|^2} w \right\| = \frac{1}{\|w\|} |w^T x^+ - w^T x^-|$.

such that: $y_i(w^T x_i+w_0)\ge 0 \triangleq$ correct class $\hat{=} |w^T x^+ - w^T x^-|$

since both are invariant to scaling of w it has exist w^* with
$w^{*T} x^+ + w_0^* = 1$, $w^{*T} x^- + w_0^* = -1$ setting that in gives the

SVM formulation $\min_w \frac{1}{2}\|w\|^2$ s.t. $-y_i(w^T x_i+w_0)\ge 1$
$f(w)$ $1-$ $\downarrow$ $\hat{=} h_i(w)$

Good, for kernels

SVM dual formulation $\max_{\alpha} \min_{w, w_0} L(w, w_0, \alpha)$ s.t. $\alpha_i \geq 0$

Start with $L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 + \sum_i \alpha_i (1 - y_i (w^T x_i + w_0))$, $\alpha_i \geq 0$

$\frac{\partial L}{\partial w} = 0 \Rightarrow w^* = \sum_i \alpha_i y_i x_i$, $\frac{\partial L}{\partial w_0} = 0 \Rightarrow \sum_i \alpha_i y_i = 0$

turns into $\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \cdot x_i x_j$, $\alpha_i \geq 0$ & $\sum_i \alpha_i y_i = 0$

can be kernelized to $\varphi(x_i)^T \varphi(x_j)$; α_i is often 0

$w_0^* = -\frac{1}{2}(w^{*T} x^+ + w^{*T} x^-)$

Soft-margin formulation

$\min_{w, \xi} \frac{1}{2}\|w\|^2 + \frac{C}{n} \sum_{i \leq n} \xi_i$

s.t. $w^T \varphi(x_i, y_i) \geq \Delta(y_i, y') + w^T \varphi(x_i, y') - \xi_i$ for $y' \neq y_i$, $i \leq n$

$\xi_i \geq 0$

hyperparameter, high C → more restricted, big margin

embedding of x: ~~[illegible]~~

horizontally stacked

slack variable $\hat{=}$ neglect example, lower the bar for neglected example

$\xi_i^* = \max(0, 1 - y_i (w^{*T} x_i + w_0^*))$ if standard margin 1

$\xi_i^* \geq \Delta(y_i, h_{w^*}(x_i))$ classified if Δ defined

Training algorithm $\varepsilon \hat{=}$ tolerance

$w = 0$

$\xi = 0$

constraints $= \emptyset$

while constraints change:

for $i \leq n$

$y' = \arg\max_{y \neq y_i} \{\Delta(y_i, y) + w^T \varphi(x_i, y)\}$

if $w^T \varphi(x_i, y_i) \leq \Delta(y_i, y') + w^T \varphi(x_i, y') - \xi_i - \varepsilon$

constraints = constraints $\cup \{ w^T \varphi(x_i, y_i) \geq$ … $\}$

$w, \xi = \text{solve}(\min_w \frac{1}{2}\|w\|^2 + \frac{C}{n} \sum_i \xi_i$ s.t. constraints$)$

Prediction: $c(x) = \arg\max_y w^T \varphi(x, y)$

One vs. rest

For each class compute compatibility function f_k, $c(x) = \arg\max_k f_k(x)$

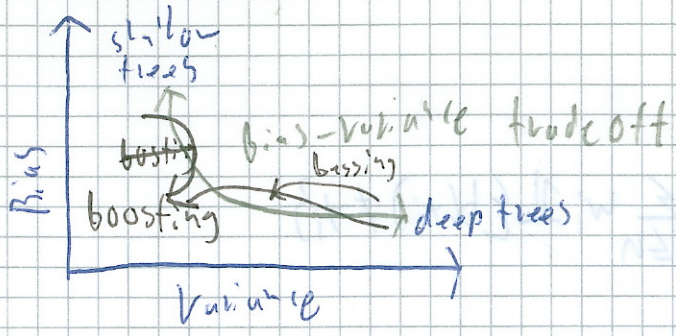
- Lots of work, no interclass learning compared to normal SVM with one score for all classes

SVM with kernels

+ works with infinitely dimensional embeddings
+ adapts to structured classification
+ formulated as quadratic programming, which is efficient

- requires careful kernel selection
- requires feature engineering
- with kernels, training algos might suffer from curse of dimensionality

Lecture 9



wisdom of crowds: If estimates are diverse, independent, decentralized, well aggregated and trusted, then the estimate ~~but~~ aggregation will be better than that of each on it's own.

Bagging: Bootstrap Aggregation: I Draw M bootstrap sets $Z'_1, Z'_2, \dots Z'_M$

II Train M base models $b^{(1)} \dots b^{(M)}$

III Aggregate $\hat{b}^{(M)}(x) = \frac{1}{M}\sum_m b^{(m)}(x)$ regression

majority$(b^{(1)}(x))$ classification

Theorem: $\mathbb{E}_{\substack{x,y \\ Z, Z'_1, Z'_2 \dots Z'_M}}\left[(y-\hat{b}^{(M)}(x))^2\right] \leq \mathbb{E}_{x,y,Z}\left[(y-b(x))^2\right]$

Proof $\mathbb{E}_{x,y,Z}\left[(y-b(x))^2\right] = \mathbb{E}_{x,y,Z}\left[(y-\mathbb{E}_Z[b(x)]+\mathbb{E}_Z[b(x)]-b(x))^2\right]$

$= \mathbb{E}_{x,y}\left[(y-\mathbb{E}_Z[b(x)])^2\right] + \mathbb{E}_x\left[\mathrm{Var}_Z[b(x)]\right] + \dots \geq 0$

$\approx \mathbb{E}_{x,y}\left[(y-\hat{b}^{(M)}(x))^2\right] + \mathbb{E}_x\left[\mathrm{Var}_Z[b(x)]\right]$ (diversity)

$\geq \mathbb{E}\left[(y-\hat{b}^{(M)}(x))^2\right]$

for

$\mathbb{E}\left[\hat{b}^{(M)}(x)\right] = \mathbb{E}_Z[b(x)]$ & $Pr_{Z'_1 \dots Z'_M}(|\mathbb{E}_Z[b(x)] - \hat{b}^{(M)}(x)| < \varepsilon) \xrightarrow[M\to\infty]{} 1$

Each classifier is trained ~~independently~~ on different requires independence

but the training sets are dependent, the correlation is small though.

(1) on different training sets which ensures diversity

Random Forests: bootstrap a set, but train ~~a~~ tree only on m features, not on all

AdaBoost

$\hat{=} \min_b \frac{1}{n} \sum L(y_i, b(x_i))$ with $\exp(-y_i f(x_i)), f(x) = \sum \alpha_t b^{(t)}(x)$

$f^{(0)} = 0$

$w_i = \frac{1}{n}$

for t in $[1 \dots M]$

train $b^{(t)} = \arg\min_b L^w(b) = \arg\min_b \sum_{i \le n} w_i \mathbb{1}\{b(x_i) \neq y_i\}$

eval $err_t = L^w(b^{(t)})$

add $f^{(t)} = f^{(t-1)} + \alpha_t b^{(t)}$, with $\alpha_t = \log\left(\frac{1}{err_t} - 1\right)$

reweight $w_i = w_i \exp(\alpha_t \mathbb{1}\{b^{(t)}(x_i) \neq y_i\})$

normalize $w_1 \dots w_n$

end

Forward stagewise additive modeling (FSAM) $\Leftarrow$ AdaBoost

$\min_{\alpha_M, b^{(M)}} \dots \min_{\alpha_1, b^{(1)}} \min_{\alpha^{(0)}, b^{(0)}} \frac{1}{n} \sum L(y_i, \alpha_0 b^{(0)}(x_i) + \alpha_1 b^{(1)}(x_i) + \dots + \alpha_M b^{(M)}(x_i))$

AdaBoost trains max-margin classifiers

$f^{(M)}(x_i) = \text{sign}\left(\sum_{t \le M} \frac{\alpha_t}{\sum_{t \le M} \alpha_t} b^{(t)}(x_i)\right)$, $\text{margin} = y_i f^{(M)}(x_i) \xrightarrow{M \to \infty} 1$

Boosting does not overfit

I with prob $\ge 1 - \delta$, gen. error $f^{(M)} \le$ [fraction of examples with margin $\le \mu$]

$+ O\left(\dots \frac{1}{n} \dots \log|H| \dots \frac{1}{\mu}\right)$

shrinks with big n; grows with big H; shrinks with big μ

II fraction of examples w/ margin $\ge \mu$ increases exponentially with M

smooth thanks to average of ensembles

interpolating $\hat{=}$ no training error

Random forests and AdaBoost train

- spiky interpolating $\Leftarrow$ mostly smooth, sharp localized changes to interpolate noisy examples
- self-averaging models
  - trained model is an average of interpolating models
    - average further localizes noise interference

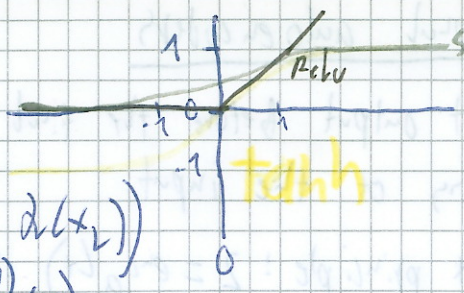
complex base $\rightarrow$ spikier

	shallow vertical stumps	stumps	tall trees
expected risk	high	medium	low
interpolating	✓	✓	✓
spiky	X	~	✓

~~1-NN~~

1-layer NN not spiky

Lecture 10



Linear functions $L(x) \longmapsto W_x$

Activation function $\alpha(x) \longmapsto (\alpha(x_1) \dots \alpha(x_2))$

Neural Net $= \alpha^{(l)} \circ L^{(l)} \circ \dots \circ \alpha^{(2)} \circ L^{(2)} \circ \alpha^{(1)} \circ L^{(1)}(x)$

Deep neural networks can approximate complex non-linear functions.

$f: [0,1]^n \to \mathbb{R}$ is continuous, $\varepsilon > 0$ then there is a NN such that

I $NN(x) = \sum_{i \le n} \alpha_i \, \varphi(w_i^T x + b_i)$ → dimension of NN can be huge

II $\max_x |f(x) - NN(x)| < \varepsilon$

softmax : $P(y_i) = \dfrac{e^{\beta z_i}}{\sum_{j \le K} e^{\beta z_j}}$

decision layer at end

Training a NN: $\min_\theta \sum_{i \le n} L(y_i, NN_\theta(x_i))$ → analytically intractable, but easy to differentiate

$L(\theta)$

↓ use gradient descent

mini-batch (stochastic) Gradient descent

repeat

use chain rule in Neural Net e.g. $\dfrac{\partial \alpha^{(l)}}{\partial z^{(l)}} \dfrac{\partial L^{(l)}}{\partial \alpha^{(l)}} \dfrac{\partial z^{(l-1)}}{\partial z^{(l-1)}} \dots$

$\dots \dfrac{\partial L^{(l)}}{\partial w_r^{(l)}} = \dfrac{\partial NN}{\partial w_r^{(l)}}$ ← exercise!!!

do for- and backprop to compute

$\left.\dfrac{\partial NN}{\partial w_r^{(l)}}\right|_{\theta, x_i}$ for each l, r and $i \le n$ and $i \in$ Sample set S

$$\frac{\partial L}{\partial w_r^{(l)}} = \sum_{i \le n} \frac{\partial L(y_i, NN_\theta(x_i))}{\partial w_r^{(l)}} \approx \sum_{i \in S} \frac{\partial L(y_i, NN_\theta(x_i))}{\partial w_r^{(l)}}$$

$$w_r^{(l)} \leftarrow w_r^{(l)} - \eta(k) \frac{\partial L}{\partial w_r^{(l)}}$$

$k \leftarrow k+1$

until test loss decreases significantly

If $\eta(k) > 0$, $\sum_n \eta(k) = \infty$, $\sum_k \eta^2(k) < \infty$ then gradient descent works according to Robbins-Monro (exercises)

Full Batch Gradient Descent

+ higher gradient precision
+ smaller generalisation error

Stochastic gradient descent

+ can handle large training sets
+ faster implementations
+ escapes local minima

Variational autoencoders

the net output before the final classification can be seen as a representation/encoding of the input.

Infomax principle: $Z = enc_\theta(X)$ is good if it maximizes $I(X,Z) = \mathbb{E}_{X,Z}\left[\log\left(\frac{p(X,Z)}{p(X)\,p(Z)}\right)\right]$

$\arg\max_\theta I(X,Z) = \mathbb{E}_{X,Z}[\log p(X|Z)] - \underbrace{\mathbb{E}_{X,Z}[\log p(X)]}_{\text{constant in } \theta}$

$\overset{\text{argmax}}{\theta} \sum_{i\le n} \mathbb{E}_{Z|X_i}[\log p(X_i|Z)]$ $enc_\theta(X_i)$

gives each value in training set a unique representation, but meaningless

Good representations are
- informative: you can guess the image given the representation
- disentangled: modifying one feature in the representation, induces change in only one perceptible feature in the reconstruction
- robust: a slight modification in the input causes a slight change in the representation

$\xrightarrow[p_{\theta'}(Z)]{\text{prior}}$ representation Z $\xrightarrow[p_\theta(X|Z)]{\text{likelihood}}$ result X

$\xleftarrow{\text{posterior}}$ $\frac{p_\theta(X|Z)\,p_{\theta'}(Z)}{p_\theta(Z|X)}$ decode $p(\cdot|\theta_Z)$

$\frac{p_\theta(X|Z)\,p_{\theta'}(Z)}{\int p_\theta(X|Z)\,p_{\theta'}(Z)\,dx}$ encode with $q(\cdot|\phi_X)$

Train VAE with $\arg\max_{\theta',\theta,\phi} \sum_{i\le n} \log p_{\theta',\theta}(x_i)$, where $p_{\theta',\theta}(x_i) = \int p_{\theta'}(z)\,p_\theta(x|z)\,dz$

$= \mathbb{E}_{Z\sim q_\phi(\cdot|x_i)}[\log p_{\theta',\theta}(x_i)]$

$\mathbb{E}_{Z\sim q_\phi(\cdot|x_i)}\left[\log\left(\frac{p_{\theta',\theta}(x_i,Z)}{p_{\theta',\theta}(Z|x_i)}\,\frac{q_\phi(Z|x_i)}{q_\phi(Z|x_i)}\right)\right]$

$= \underbrace{\mathbb{E}_{Z\sim q_\phi(\cdot|x)}\left[\log\left(\frac{p_{\theta',\theta}(x_i,Z)}{q_\phi(Z|x_i)}\right)\right]}_{\text{elbo}_{\theta',\theta,\phi}(x_i)} + \underbrace{\mathbb{E}_{Z\sim q_\phi(\cdot|x_i)}\left[\log\left(\frac{q_\phi(Z|x_i)}{p_{\theta',\theta}(Z|x_i)}\right)\right]}_{KL(q_\phi(\cdot|x_i)\,|\,p_{\theta',\theta}(\cdot|x_i)) \ge 0}$

$= \underbrace{\mathbb{E}_{Z\sim q_\phi(\cdot|x_i)}[\log p_\theta(x_i|Z)]}_{\text{Info max}} + \underbrace{\mathbb{E}_{Z\sim q_\phi(\cdot|x_i)}\left[\log\left(\frac{p_{\theta'}(Z)}{q_\phi(Z|x_i)}\right)\right]}_{-KL(q_\phi(\cdot|x_i)\,|\,p_{\theta'}(\cdot))}$

errates informative

$\hat{=}$ regularization term

robustness by choice of $p_\theta(\cdot|Z)$ and $q_\phi(\cdot|X)$ — disentanglement

Lecture 11

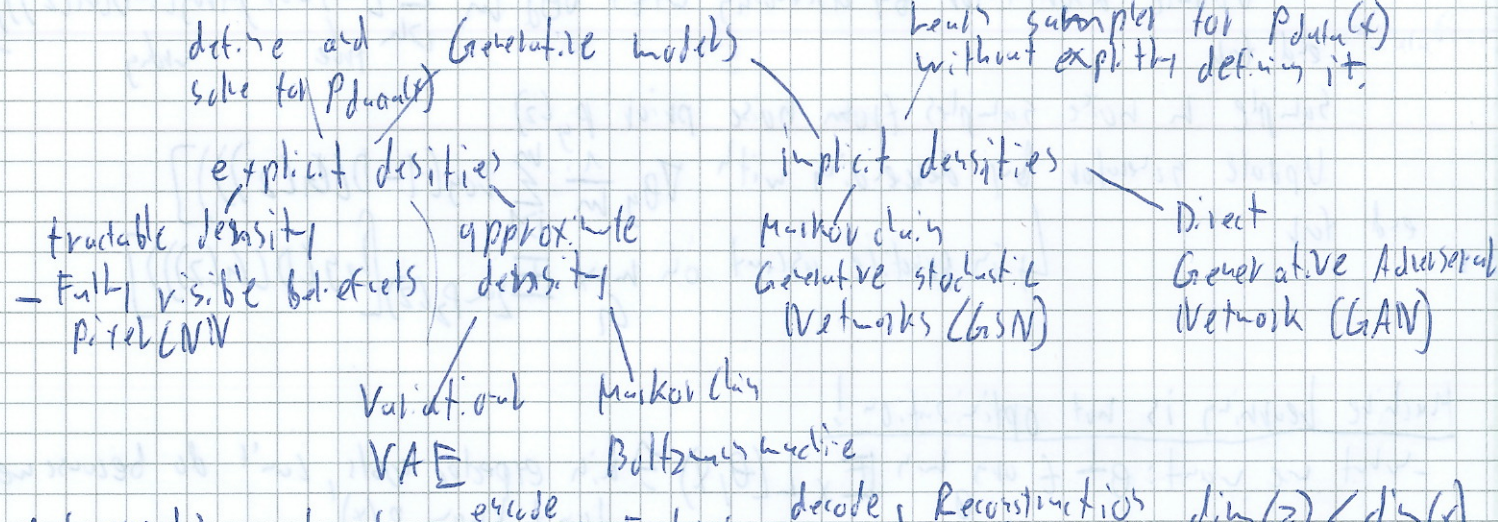
Multiple NN layers: some functions can not be efficiently represented by shallow architectures

Consequences: ~~one hidden~~ we don't need exponentially many nodes per layer

- poor generalisation may be expected when using a too ~~deep~~ shallow representation

Rule of thumb: 5-10 training examples per independent weight at least

Generative models goal: learn density $p_{model}(x)$ close to $p_{data}(x)$



Autoencoder: Input x —encode→ Features z —decode→ Reconstruction $\hat{x}$, $dim(z) < dim(x)$

loss $\|x - \hat{x}\|^2$

For inference: throw away decoder and fine-tune on last tasks

Denoising auto encoders: "blank out" certain pixels in x

Want to learn invariances which are robust to translations, rotations, lighting

- ↳ augment data set
- ↳ resolution-fan tasks
- ↳ pre-process e.g. with SIFT
- ↳ local strategies e.g. CNNs ← convolution, pooling layers aggregate convolayer

VAEs can generate data via choosing from representation

+ principled approach to generative models
+ allows inference of $q(z|x)$, can be useful for other tasks

- maximizes lower bound, not as good as more complex models like PixelCNN
- samples are blurry and more computed to GANs

GANs state-of-the-art, but hard to train, can't use explicit density functions

Generator network tries to fool discriminator by generating fake $G(z)$ (upsampling network usually CNN)

Discriminator network tries to distinguish between real and fake images, often CNN

$$\min_G \max_D V(D,G) = \min_G \max_D \{\mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))]\}$$

for number of training iterations do

  for k steps do

    Sample m noise samples $\{z^1,\dots z^m\}$ from noise prior $p_g(z)$

    sample m examples $\{x^1,\dots x^m\}$ from data distribution $p_{data}(x)$

    Update discriminator by ascending with $\nabla_{\theta_d} \frac{1}{m}\sum_{i=1}^{m}[\log D(x^i) + \log(1-D(G(z^i)))]$ (true / wrong)

  end for

  Sample m noise samples from noise prior $p_g(z)$

  Update generator by descending with $\nabla_{\theta_g} \frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(z^i)))]$

  ↳ in practice ascent on $\max_G \mathbb{E}_{z\sim p_z(z)}[\log(D(G(z)))]$

end for

Machine learning is not optimization!

- what we want: $\theta^* \in \arg\min_\theta \mathbb{E}_x L(\theta, x)$ ≙ min expected costs, can't do because we don't know $p(x)$
- what we can: $\hat\theta(x) \in \arg\min_\theta L(\theta, x)$ ≙ might not be representative, min empirical costs

↙ regularization, but not optimal, but typical

Sensitivity analysis training $(x') = \bar x + \Delta$, test $(x'') = \bar x - \Delta$

expected costs: $\mathbb{E}_{\theta|x'}[-\log p(\theta|x'')] = -\int p(\theta|\bar x+\Delta)\log p(\theta|\bar x-\Delta)\,d\theta$

$$= -\int \left(p(\theta|\bar x) + \nabla_\Delta p(\theta|\bar x+\Delta)\Big|_{\Delta=0}\cdot\Delta\right)\cdot\left(\log p(\theta|\bar x) - \frac{\nabla_\Delta p(\theta|\bar x-\Delta)}{p(\theta|\bar x-\Delta)}\Big|_{\Delta=0}\cdot\Delta\right)$$

$$= -\mathbb{E}_{\theta|\bar x}\log p(\theta|\bar x) - \Delta\int \nabla_\Delta p(\theta|\bar x)(\log p(\theta|\bar x) - 1)\,d\theta + O(\Delta^2)$$

Sample error is linear in Δ

Posterior argument: $\mathbb{E}_{\theta|x'}[p(\theta|x'')] = \int p(\theta|\bar x)^2\,d\theta + O(\Delta^2)$

↳ robustness

Lecture 12

k-means problem: Find $c()$ and μ that minimize

$$R^{km}(c,\mu) = \sum_{x \in X} \|x - \mu_{c(x)}\|^2$$ ($\mu_{c(x)}$: cluster center)

k-means algorithm:

Init $\mu_c = x_c$ ← some random data points

repeat

hold assignment to exactly one cluster

for all x, $c(x) = \arg\min_c \|x - \mu_c\|^2$ Expectation step

for all μ, $\mu_a = \frac{1}{n_a} \sum_{c(x)=a} x$, $n_a = \#$ all points in cluster Maximization

until no changes in $c(x)$

EM for Mixture Gaussians ← detailed in exercise 1

latent variable — soft assignment to, computed for data x and c = probability of x in c

repeat

$$E: \gamma_{xc} = \frac{p(x|c,\theta^{(j)})\, p(c|\theta^{(j)})}{p(x|\theta^{(j)})}$$

$$M: \mu_c^{(j+1)} = \frac{\sum_x \gamma_{xc}\, x}{\sum_x \gamma_{xc}}, \quad s_c^{2\,j+1} = \frac{\sum_x \gamma_{xc}(x-\mu_c)^2}{\sum_x \gamma_{xc}}, \quad \pi_c^{j+1} = \frac{1}{|X|}\sum_x \gamma_{xc} \Big/ p(c|\theta^{(j)})$$

until no changes in E-step

solves $L(X|\pi_1,\dots,\pi_k,\theta_1,\dots,\theta_k) = \sum_x \log \sum_c \pi_c\, p(x|\theta_c)$

hard to track in log

amount of clusters is a hyperparameter

Beta distribution: $p(x|a,b) = \frac{1}{B(a,b)} \cdot x^{a-1}(1-x)^{b-1}$, $x \in [0,1]$, $a,b > 0$

$B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$, where $\Gamma(a) = \int_0^\infty e^{-x} x^{a-1} dx$

Dirichlet $(x|\alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{h} x_k^{\alpha_k - 1}$, $x_i \in [0,1]$, $\alpha_i > 0$

$B(\alpha) = \frac{\prod_{k=1}^{h} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{h} \alpha_k)}$

Beta is Dirichlet with only two variables

have "∞"-many clusters $\hat{=}$ nonparametric Bayesian methods

Probabilities still have to sum up to 1

Use Dirichlet Process (α, H) — base measure

↳ stick breaking — concentration

$\beta_1 \cong$ ~~Beta$(\alpha_1, \alpha_2 \dots \alpha_k)$~~ Beta$(1, \alpha)$, $p_1 = \beta_1$

$\beta_2 \cong$ Beta$(1, \alpha)$, $p_2 = \beta_2(1-\beta_1)$

$\beta_3 \sim$ Beta$(1, \alpha)$, $p_3 = \beta_3(1-\beta_1)(1-\beta_2)$

$\beta_k \sim$ Beta$(1, \alpha)$, $p_k = \beta_k(1 - \sum_{i=1}^{k-1} p_i)$

$G \in$ DM distribution

$G(\theta) = \sum_{k=1}^{\infty} p_k \delta_{\theta_k}(\theta) \sim DP(\alpha, H)$

Dirac, H

Chinese Restaurant Process

$P(\text{customer } n+1 \text{ joins table } T \mid \mathcal{P}) = \begin{cases} \frac{|T|}{\alpha+n} & \text{if } T \in \mathcal{P} \\ \frac{\alpha}{\alpha+n} & \text{new table} \end{cases}$ tables: $\sim O(\alpha \log(n))$

↳ process is order and labeling independent

↳ exchangeable!!!

Exchangeability: $(X_1, X_2 \dots X_n)$ is a sequence of random variables. It is exchangeable if also under the permutation Π $(X_1, X_2 \dots)$ have the same distribution as $(X_{\Pi(1)}, X_{\Pi(2)} \dots)$

De Finetti's theorem: $p(X_1 \dots X_n) = \int \left(\prod_{i=1}^{n} p(x_i|G)\right) dP(G)$ $\hat{=}$ An inf. many exchangeable sequence can be represented by conditionally independent random variables

Pólya Urn: Pull a ball, put that ball + one of same colour back, with

Hoppe Urn: Pull a ball, if coloured → next colour, if black → new colour, put back black and new colour

↳ same as chinese restaurant process

Fit DP Mixture Model with Gibbs Sampling

$$p(z_i = k \mid z_{-i}, x, \alpha, \mu) \propto \underbrace{p(z_i = k \mid z_{-i}, \alpha)}_{\text{prior}} \underbrace{p(x_i \mid x_{-i}, z_i = k, z_{-i}, \mu)}_{\text{likelihood}}$$

$-i \hat{=}$ all existing i

Chinese restaurant

$$\begin{cases} \frac{p(x_i, x_{-i,k} \mid \mu)}{p(x_{-i,k} \mid \mu)} & \text{all points in cluster } k \text{ if } k \text{ exists} \\ p(x_i \mid \mu) & \text{if } k \text{ does not exist} \end{cases}$$

Model selection by minimizing validation costs with posterior

$$p(\theta \mid X) = \exp(-\beta w(\theta, X)) / \# \quad \text{with } w(\theta, X) = R(\theta, X) - F(X)$$

Validation Loss: $\mathbb{E}_{\theta \mid X'}[-\log p(\theta \mid X'')] = \beta \mathbb{E}_{\theta \mid X'}[w(\theta, X'')] = \beta(\underbrace{\mathbb{E}_{\theta \mid X'} R(\theta, X'')}_{\text{common}} - F(X''))$

(validset, training; why this is significant)

posterior selection: $\min_{p(\cdot \mid \cdot)} \mathbb{E}_{\theta \mid X'}[-\log p(\theta \mid X'')] \geq \min_{p(\cdot \mid \cdot)} -\log(\mathbb{E}_{\theta \mid X'} p(\theta \mid X''))$ (Jensen)

$$= -\max_{p(\cdot \mid \cdot)} \log \int \underbrace{p(\theta \mid X') p(\theta \mid X'')}_{\text{probability kernel} \rightarrow \text{more similar} \rightarrow \text{better}} d\theta$$

Lecture 13

Generalization error: $R(\hat{c}) = P(\hat{c}(x) \neq c(x))$ — $\mathbb{E}[\hat{R}_n(\hat{c})] = R(\hat{c})$

Empirical error: $\hat{R}_n(\hat{c}) = \frac{1}{n} \sum_{i \leq n} \mathbb{1}_{\hat{c}(x_i) \neq c(x_i)}$

Instance space X ≙ data in learners world

Concept $c \subseteq X$ ≙ subset of X ≙ classes

Concept class ≙ a set of concepts we want to learn

hypothesis class ≙ another set of concepts that we use to learn the target concept

Learning algorithm ≙ input: labeled samples $Z = \{(x_1, y_1), \dots, (x_n, y_n)\}$ with $y_i = c(x_i)$

output: $\hat{c} \in$ H-pothesis class

Can learn: A can learn C if there is a poly-function such that
- for any distribution D on X and
- for any $0 < \varepsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$

A outputs $\hat{c}$ such that $P_{Z \sim D^n}(R(\hat{c}) \leq \varepsilon) \geq 1 - \delta$ (error, confidence, chance of being too wrong)

~~with a sample Z of~~ training set / how wrong we can be

with a sample Z of size $\geq poly(\frac{1}{\varepsilon}, \frac{1}{\delta}, size(c))$

PAC Learnable: concept class C is PAC learnable for a hypothesis class H if there is an algorithm that can learn any concept in C

Efficient PAC Learnable: A learns in time polynomial in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$

the universal concept class $C = \{0,1\}^*$ is not learnable from itself!

General setting: D on $X \times \{0,1\}$ allows for classes having different labels
↳ much harder, ~~learn~~ instead of $R(\hat{c}) \leq \varepsilon$ try to learn $(R(\hat{c}) - \inf_{c \in C} R(c)) \leq \varepsilon$

General PAC Learning: - for any D on $X \times \{0,1\}$ and
- for any $0 < \varepsilon < \frac{1}{2}$ and $0 < \delta < \frac{1}{2}$

A receives sample Z of size $n \geq poly(\frac{1}{\varepsilon}, \frac{1}{\delta}, dim(X))$ and can output $\hat{c}$ such that

$$P_{Z \sim D^n}(R(\hat{c}) - \inf_{c \in C} R(c) \leq \varepsilon) \geq 1 - \delta$$

Shattering: a set of instances A can be shattered by a concept class C if for every subset $S \subseteq A$, there is a concept $c_S \in C$ such that $A \cap S = c_S \cap A$

e.g. set of two points in $\mathbb{R}^2$ and $C \hat{=}$ rectangles
not 3 numbers in $\mathbb{R}$ and $C \hat{=}$ intervals

VC-dimension: dimensions needed to shatter set → complexity measure for C

weak learner: PAL learnable ~~only~~ for "large" error ε
strong learner: PAL learnable for all errors ε

Error probability for realizable finite hypothesis classes $H = C$

Assume $\min_{c \in C} R(c) = 0$, $\hat{c} \in \arg\min_{c \in C} \hat{R}_n(c)$

$$P(R(\hat{c}) > \varepsilon) = P\left(\max_{c \in C,\, \hat{R}_n(c)=0} R(c) > \varepsilon\right) \quad \text{max definition in sum}$$

$$\leq \sum_{c \in C} P(R(c) > \varepsilon \wedge \hat{R}_n(c) = 0)$$

$$\leq |C| (1 - R(c))^n \leq |C| e^{-n\varepsilon} \leq \delta$$

$$n > \frac{1}{\varepsilon} \left(\log |C| + \log \frac{1}{\delta} \right)$$

Vapnik-Chervonenkis inequality

$\hat{c} \in \arg\min_{c \in C} \hat{R}_n(c) = \arg\min_{c \in C} |\{\hat{c}(x_i) \neq y_i\}|$ $\hat{=}$ minimizer of empirical risk

$c^* \in \arg\min_{c \in C} R(c)$ $\hat{=}$ minimizer of expected risk

$$R(\hat{c}) - \inf_{c \in C} R(c) = \underbrace{R(\hat{c}) - \hat{R}_n(\hat{c})}_{\leq} + \underbrace{\hat{R}_n(\hat{c}) - R(c^*)}_{\leq \hat{R}_n(c^*)} \leq 2 \sup_{c \in C} |R(c) - \hat{R}_n(c)|$$