Exam questions

Lucie Kunčarová
exam date:
16.01.2023

Lecture 1 Questions

- Define prediction function of a linear regression model and write down L^2-regularized mean squared error loss. [5]

prediction

$$y(x; w, b) = x^T w + b$$

kde $x \in \mathbb{R}^D$ ... input, b ... bias, w ... weights

L^2 regularization

$$\frac{1}{2}\sum_{i=1}^{N}(y(x_i; w) - t_i)^2 + \frac{\lambda}{2}\|w\|^2$$

- Starting from unregularized sum of squares error of a linear regression model, show how the explicit solution can be obtained, assuming $X^T X$ is regular. [10]

unregularized sum of squares:

$$\frac{1}{2}\sum_{i=1}^{N}(y(x_i; w) - t_i)^2 \rightarrow \text{chceme minimalizovat}$$

→ zkoumáme hodnoty, kde derivace error fce je 0 přes všechna w_j

$$\frac{\partial}{\partial w_j}\frac{1}{2}\sum_{i}^{N}(x_i^T w - t_i)^2 = \frac{1}{2}\sum_{i}^{N} 2(x_i^T w - t_i)x_{ij}) = \sum_{i}^{N} x_{ij}(x_i^T w - t_i)$$

$\Rightarrow$ tedy chceme $\forall j$ $\sum_{i}^{N} x_{ij}(x_i^T w - t_i) = 0$

↳ $X_{*j}^T (Xw - t) = 0 \Rightarrow X^T(Xw - t) = 0$

→ z čehož $X^T X w = X^T t$ → zde použijeme reg. $X^T X$

↳ $w = (X^T X)^{-1} X^T t$

Lecture 2 Questions

- Define expectation $\mathbb{E}[f(x)]$ and variance $\mathrm{Var}(f(x))$ of a discrete random variable. Then define the bias of an estimator and show that estimating an expectation using a single sample is unbiased. [5]

$$\mathbb{E}_{x\sim p}[f(x)] \overset{\text{def}}{=} \sum_x p(x)\cdot f(x)$$

přes $\forall x$ ↗ ↑ prob. dist.

$$\mathrm{var}(f(x)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$$

also $\mathrm{var}(x) = \mathbb{E}[x^2] - (\mathbb{E}[x])^2$

bias = $\mathbb{E}[\text{estimate}]$ − true estimated value

let's estimate $\mathbb{E}_p[f(x)]$ by single sample x from p and returning $f(x)$ ⇒ je to unbiased, protože $\mathbb{E}[\text{estimate}] = \mathbb{E}_p[f(x)]$ což je přesně true estimated value

2. Describe standard gradient descent and compare it to stochastic (i.e., online) gradient descent and minibatch stochastic gradient descent. [5]

Gradient Descent:

→ chceme najít nejlepší váhy weights inkrementální cestou

→ chceme-li minimalizovat error fci $\arg\min_w E(w)$

↳ grad. desc.: $w \leftarrow w - \alpha \nabla E(w)$

↳ learning rate = délka kroku v $\forall$ iteraci

→ nechť $X \in \mathbb{R}^{N\times D}$, $t \in \mathbb{R}^N$ train data

a $\hat{p}_{data}(x,t) = \frac{|\{i : (x,t) = (x_i,t_i)\}|}{N}$ a předp. že

$$E(w) = \mathbb{E}_{(x,t)\sim\hat{p}_{data}} L(y(x;w),t)$$

↳ $\nabla_w E(w) = \mathbb{E}_{(x,t)\sim\hat{p}_{data}} \nabla_w L(y(x;w),t)$

- standard GD
  ↳ použijeme na výpočet $\nabla_w E(w)$ všechna train data

- stochastic GD
  - odhadneme $\nabla_w E(w)$ pomocí jednoho náhodného examplu z train dat → unbiased but very noisy

$$\nabla_w E(w) \propto \nabla_w L(y(x_i; w), t) \text{ pro náhodně zvolené } (x, t) \text{ z } \hat{P}_{data}$$

- Minibatch GD → běžně používán
  - něco mezi → $\nabla_w E(w)$ se odhadne pomocí B nezávislých náhodných examplů z train dat

$$\nabla_w E(w) \approx \frac{1}{B} \sum_{i=1}^{B} \nabla_w L(y(x_i; w), t_i) \text{ pro náhodně zvolené } (x_i, t_i) \text{ z } \hat{P}_{data}$$

2. Formulate conditions on the sequence of learning rates used in SGD to converge to optimum almost surely. [5]

→ nechť α_i je posloupnost learning rates
  - potom podmínky:

$$\forall i: \alpha_i > 0, \quad \sum_i \alpha_i = \infty, \quad \sum \alpha_i^2 < \infty \left(\Rightarrow \alpha_i \to 0\right)$$

(+ error fce musí být konvexní a spojitá → MSE je)

2. Write an L^2-regularized minibatch SGD algorithm for training a linear regression model, including the explicit formulas of the loss function and its gradient. [5]

Input: Dataset ($\boldsymbol{X} \in \mathbb{R}^{N \times D}$, $\boldsymbol{t} \in \mathbb{R}^N$), learning rate $\alpha \in \mathbb{R}^+$, L^2 strength $\lambda \in \mathbb{R}$.
Output: Weights $\boldsymbol{w} \in \mathbb{R}^D$ hopefully minimizing the regularized MSE of a linear regression model.

- $\boldsymbol{w} \leftarrow 0$ or we initialize $\boldsymbol{w}$ randomly
- repeat until convergence (or until our patience runs out):
  - sample a minibatch of examples with indices $\boldsymbol{b}$
    - either uniformly randomly,
    - or we may want to process all training instances before repeating them, which can be implemented by generating a random permutation and then splitting it into minibatch-sized chunks
      - the most common option; one pass through the data is called an **epoch**
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \sum_{i \in \boldsymbol{b}} \frac{1}{|\boldsymbol{b}|} \left((\boldsymbol{x}_i^T \boldsymbol{w} - t_i) \boldsymbol{x}_i \right) - \alpha \lambda \boldsymbol{w}$

Lecture 3 Questions

3. Define binary classification, write down the perceptron algorithm and show how a prediction is made for a given example. [5]

→ klasifikace do dvou tříd

→ můžeme rozšířit lin. regr.
↳ najdem threshold (0): $y(x_i; w) = x^T w + b$ $\begin{cases} > \text{thresh.} \\ < \text{—"—} \end{cases}$

perceptron:

input: linearly separable dataset $(X \in \mathbb{R}^{N \times D}, t \in \{-1, 1\}^N)$

output: weights $w \in \mathbb{R}^D$ t.z. $t_i x_i^T w > 0 \quad \forall i$

- $w \leftarrow 0$
- until all examples are classified correctly, process example i:
  - $y \leftarrow x_i^T w$
  - if $t_i y \leq 0$: (incorrectly classified)
    - $w \leftarrow w + t_i x_i$

3. Show that the perceptron algorithm is an instance of stochastic gradient descent. Why are the learning rates not needed (i.e., why are the predictions of a trained model the same for all positive learning rates)? [5]

algoritmus: (tělo)

- $y \leftarrow x_i^T w$
- if $t_i y \leq 0$: (incorrectly classified)
  - $w \leftarrow w + t_i x_i$

→ umíme přepsat pomocí online grad. desc. s loss fcí:

$$L(y(x_i; w), t) = \begin{cases} -tx^T w & \text{if } tx^T w \leq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= \max(0, -tx^T w) = \text{ReLU}(-tx^T w) \rightarrow \text{loss function}$$

→ zajímá nás jen, jestli je predikce větší, nebo menší než 0 → jak moc větší/menší už nám je jedno

↳ vynásobení w konstantou nezmění predikci a loss derivative nezávisí na w

3. For discrete random variables, define entropy, cross-entropy, Kullback-Leibler divergence, and prove the Gibbs inequality (i.e., that KL divergence is non-negative). [5]

entropy:

P discrete : $H(P) = -\sum_x P(x) \log P(x)$

cross-entropy:

$$H(P,Q) = -\mathbb{E}_{x\sim P}[\log Q(x)]$$

(P, Q: distribuce)

Gibbs inequality:

- $H(P,Q) \geq H(P)$
- $H(P) = H(P,Q) \Leftrightarrow P = Q$

↳ proof:

uvažme $H(P) - H(P,Q) = \sum_x P(x) \log \frac{Q(x)}{P(x)}$

↳ využijeme faktu, že $\log x \leq (x-1)$ kde $\log x = (x-1)$ pro $x = 1$

↳ $\sum_x P(x) \log \frac{Q(x)}{P(x)} \leq \sum P(x)\left(\frac{Q(x)}{P(x)} - 1\right) =$

$= \sum_x Q(x) - \sum_x P(x) = 0$

↳ aby rovnost platila, musí $\frac{Q(x)}{P(x)} = 1 \ \forall x \Rightarrow P = Q$ □

KL divergence:

$$D_{KL}(P\|Q) = H(P,Q) - H(P) = \mathop{\mathbb{E}}_{x\sim P}[\log P(x) - \log Q(x)]$$

↳ Gibbs ineq. $D_{KL}(P\|Q) \geq 0$, $D_{KL}(P\|Q) = 0 \Leftrightarrow P = Q$

3. Define data-generating distribution, empirical data distribution and likelihood. [5]

data-generating dist. P_{data}

↳ process, který umí ta data generovat
→ neznáme ho a je abstraktní
↳ můžeme ho odhadnout

empirical data dist. $\hat{P}_{data}$

$$\hat{P}_{data}(x) = \frac{|\{i : x_i = x\}|}{N}$$

↳ přesně popisuje train data, je diskrétní
→ aproximace skutečné distribuce

likelihood:

nechť $p_{model}(x; w)$ je rodina distribucí

- if weights fixed, $p_{model}(x; w)$ je prob. dist.
- if naopak X je fixní:

↳ train data

$$L(w) = p_{model}(X; w) = \prod_{i=1}^{N} p_{model}(x_i; w)$$

↳ likelihood

→ hodnota $\in [0,1]$, ale nemusí se sčítat na 1

3. Describe maximum likelihood estimation, as minimizing NLL, cross-entropy and KL divergence. [10]

MLE of w:

$$w_{MLE} = \arg\max_w p_{model}(X; w) = \arg\max_w \prod_{i=1}^{N} p_{model}(x_i; w) =$$

$$= \arg\min_w \sum_{i=1}^{N} -\log p_{model}(x_i; w) =$$

$$= \arg\min_w \mathbb{E}_{x \sim \hat{p}_{data}} [-\log p_{model}(x; w)] =$$

$$= \arg\min_w H(\hat{p}_{data}(x), p_{model}(x; w)) =$$

$$= \arg\min_w D_{KL}(\hat{p}_{data}(x) \| p_{model}(x; w)) + H(\hat{p}_{data})$$

$\Rightarrow$ generalized to conditional case
↳ predict t given x

$$w_{MLE} = \arg\max_w p_{model}(t|X; w) = \arg\max_w \prod_{i=1}^{N} p_{model}(t_i|x_i; w) =$$

$$= \arg\min_w \sum_{i=1}^{N} -\log p_{model}(t_i|x_i; w) =$$

NLL

$$= \arg\min_w \mathbb{E}_{x \sim \hat{p}_{data}} [-\log p_{model}(t|x_i; w)] =$$

$$= \arg\min_w H(\hat{p}_{data}(x), p_{model}(t|x_i; w)) =$$

CE

$$= \arg\min_w D_{KL}(\hat{p}_{data}(x) \| p_{model}(t|x_i; w)) + H(\hat{p}_{data})$$

KL-div

→ kde conditional entropy: $H(\hat{p}_{data}) = \mathbb{E}_{(x,t) \sim \hat{p}_{data}} [-\log \hat{p}_{data}(t|x; w))]$
a cond. cross. entropy: $H(\hat{p}_{data}, p_{model}) = \mathbb{E}_{(x,t) \sim \hat{p}_{data}} [-\log (p_{model}(t|x; w))]$

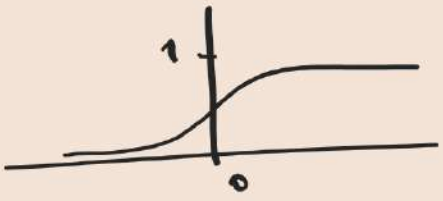
→ výsledná loss fce je NLL / KL divergence / cross-entr.

3. Considering binary logistic regression model, write down its parameters (including their size) and explain how prediction is performed (including the formula for the sigmoid function). Describe how we can interpret the outputs of the linear part of the model as logits. [5]

(z perceptronu)

sigmoid:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



parametrization:

$$p(C_1|x) = \sigma(x^T w + b)$$

$$p(C_0|x) = 1 - p(C_1|x)$$

→ predikce je výsledek lineární regrese "prohnaný" sigmoidem

↳ $y(x_i; w) = \sigma(\bar{y}(x_i; w)) = \sigma(x^T w)$

lin. část $\bar{y}(x_i; w) = x^T w$

$$p(C_1|x) = \sigma(\bar{y}(x_i; w)) = \frac{1}{1+e^{-\bar{y}(x_i; w)}}$$

$$\Rightarrow \bar{y}(x_i; w) = \log\left(\frac{p(C_1|x)}{1-p(C_1|x)}\right) = \log\left(\frac{p(C_1|x)}{p(C_0|x)}\right)$$

logit

logit = logaritmus odds of the probs. of the 2 classes

3. Write down an L^2-regularized minibatch SGD algorithm for training a binary logistic regression model, including the explicit formulas of the loss function and its gradient. [10]

Input: Input dataset ($\boldsymbol{X} \in \mathbb{R}^{N \times D}$, $\boldsymbol{t} \in \{0, +1\}^N$), learning rate $\alpha \in \mathbb{R}^+$.

- $\boldsymbol{w} \leftarrow 0$
- until convergence (or patience runs out), process a minibatch of examples with indices $\boldsymbol{b}$:
  - $\boldsymbol{g} \leftarrow \frac{1}{|\boldsymbol{b}|} \sum_{i \in \boldsymbol{b}} \nabla_{\boldsymbol{w}} - \log\left(p(C_{t_i}|\boldsymbol{x}_i; \boldsymbol{w})\right)$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \boldsymbol{g}$

→ loss for a minibatch $X = \{(x_1, t_1), \dots, (x_N, t_N)\}$ je

$$E(w) = \frac{1}{N} \sum_i -\log\left(p(C_{t_i} | x_i; w)\right)$$

→ gradient: $(y(x) - t)\, x$

↳ přímým výpočtem

Lecture 4 Questions

- Define mean squared error and show how it can be derived using MLE. [5]

MSE:

$$\arg\min_w \frac{1}{N} \sum_{i=1}^{N} (y(x_i; w) - t_i)^2$$

→ vezmeme dist. s nejvyšší entropií: normální

↳ předpokl. tedy, že náš model generuje dist.

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

→ aplikujeme MLE:

$$\arg\max_w p(t | X; w) = \arg\min_w \sum_{i=1}^{N} -\log p(t_i | x_i; w) =$$

$$= \arg\min_w - \sum_{i=1}^{N} \log \sqrt{\frac{1}{2\pi\sigma^2}} \cdot e^{-\frac{(t_i - y(x_i; w))^2}{2\sigma^2}} =$$

$$= \arg\min_w - N \log (2\pi\sigma^2)^{-\frac{1}{2}} - \sum_{i=1}^{N} - \frac{(t_i - y(x_i; w))^2}{2\sigma^2} =$$

$$= \arg\min_w \sum_{i=1}^{N} \frac{(t_i - y(x_i; w))^2}{2\sigma^2} = \arg\min_w \frac{1}{N} \sum_{i=1}^{N} (y(x_i; w) - t_i)^2$$

- Considering K-class logistic regression model, write down its parameters (including their size) and explain how prediction is performed (including the formula for the softmax function). Describe how we can interpret the outputs of the linear part of the model as logits. [5]

→ bin. logistic regression extension

→ generate K outputs, každý se svými vahami

$W \in \mathbb{R}^{D \times K}$

→ lin. část

$\bar{y}(x; W) = x^T W \quad \dots \quad \bar{y}(x, W)_i = x^T (W_{*i})$

aktivační fce: softmax (zobecněný sigmoid)

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

→ predikce je lineární část prohnaná softmaxem

↳ $p(C_i | x; W) = y(x; W)_i = \text{softmax}(\bar{y}(x; W))_i = \text{softmax}(x^T W)_i$

→ interpretace lin. části → logits

$\bar{y}(x, W)_i = \log(p(C_i | x; W)) + c$ → konst. kvůli overparametrizaci

↳ softmax je invariantní k přičítání konstanty

- Write down an L^2-regularized minibatch SGD algorithm for training a K-class logistic regression model, including the explicit formulas of the loss function and its gradient. [10]

Input: Input dataset ($\boldsymbol{X} \in \mathbb{R}^{N \times D}$, $\boldsymbol{t} \in \{0, 1, \dots, K-1\}^N$), learning rate $\alpha \in \mathbb{R}^+$.
Model: Let $\boldsymbol{w}$ denote all parameters of the model (in our case, the parameters are a weight matrix $\boldsymbol{W}$ and maybe a bias vector $\boldsymbol{b}$).

- $\boldsymbol{w} \leftarrow 0$ or we initialize $\boldsymbol{w}$ randomly
- until convergence (or patience runs out), process a minibatch of examples with indices $\boldsymbol{b}$:
  - $\boldsymbol{g} \leftarrow \frac{1}{|\boldsymbol{b}|} \sum_{i \in \boldsymbol{b}} \nabla_{\boldsymbol{w}} - \log\left(p(C_{t_i} | \boldsymbol{x}_i; \boldsymbol{w})\right)$
  - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \boldsymbol{g}$

gradient → $(y(x) - 1_t)\, x^T$

↳ one-hot

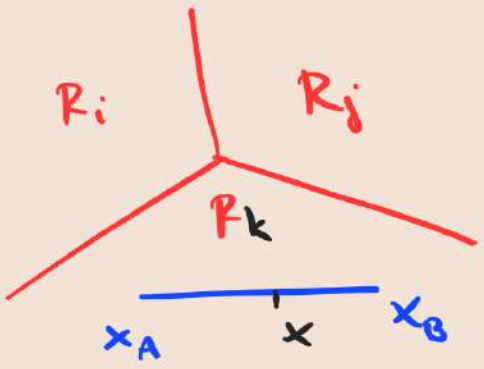
4. Prove why are decision regions of a multiclass logistic regression convex. [5]

- každý bod $\in x_A x_B$ je jejich konvexní kombinací

$x = \lambda x_A + (1-\lambda) x_B$ a z linearity

$\bar{y}(x) = x^T W$:

$\bar{y}(x) = \lambda \bar{y}(x_A) + (1-\lambda)\bar{y}(x_B)$



→ protože $\bar{y}(x_A)_k$ bylo největší z $\bar{y}(x_A)$ a taky $\bar{y}(x_B)_k$ bylo největší z $\bar{y}(x_B) \Rightarrow \bar{y}(x)_k$ musí být největší z $\bar{y}(x)$

4. Considering a single-layer MLP with D input neurons, H hidden neurons, K output neurons, hidden activation f and output activation a, list its parameters (including their size) and write down how the output is computed. [5]

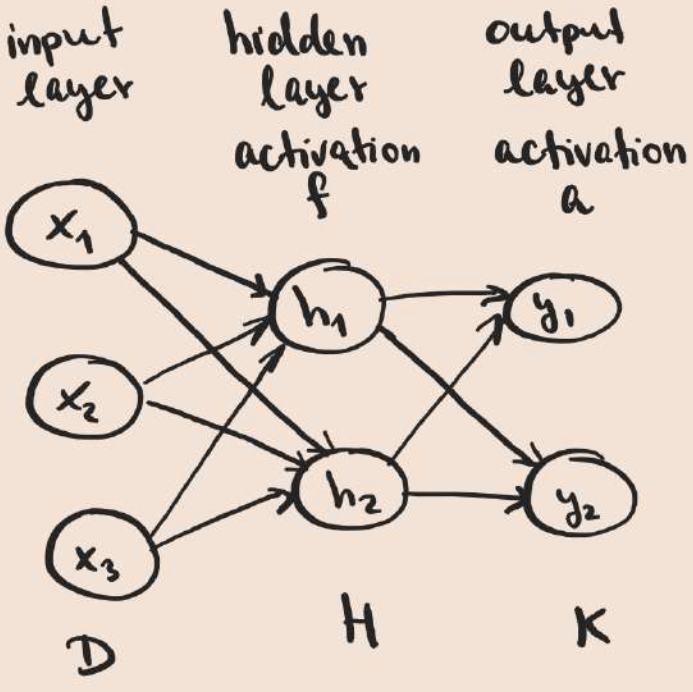
$$h_i = f\left(\sum x_j w_{ji}^{(h)} + b_i^{(h)}\right)$$

$$y_i = a\left(\sum h_j w_{ji}^{(y)} + b_i^{(y)}\right)$$

→ matrix form:

$$h = f\left(x^T W^{(h)} + b^{(h)}\right)$$

$$y = a\left(h^T W^{(y)} + b^{(y)}\right)$$



→ for batch of inputs:

$$H = f\left(XW^{(h)} + b^{(h)}\right) \quad \text{and} \quad Y = a\left(XW^{(y)} + b^{(y)}\right)$$

4 • List the definitions of frequently used MLP output layer activations (the ones producing parameters of a Bernoulli distribution and a categorical distribution). Then write down three commonly used hidden layer activations (sigmoid, tanh, ReLU). [5]

output:

for binary classification:

- sigmoid $\sigma(x)$: we model Bernoulli dist.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

for K-class classification:

- softmax(x): we model (usually overparametrized) categorical dist.

$$\text{softmax}(x) \propto e^{x} \qquad \text{softmax}(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

hidden:

- identity is useless
- sigmoid → není symetrický → works suboptimally
  ↳ $\frac{d\sigma}{dx}(0) = \frac{1}{4}$
- tanh → making σ symmetrical and derivation in zero is 1
  ↳ $\tanh(x) = 2\sigma(2x) - 1$
- ReLU → $\max(0, x)$ → most common nowadays

4 • Considering a single-layer MLP with D input neurons, a ReLU hidden layer with H units and a softmax output layer with K units, write down the formulas of the gradient of all the MLP parameters (two weight matrices and two bias vectors), assuming input $\boldsymbol{x}$, target t and negative log likelihood loss. [10]

→ postup pro spočítání gradientu of the loss L vzhledem ke všem vahám:

1) spočítat $\frac{\partial L}{\partial y}$

2) spoč. $\frac{\partial y}{\partial y^{(in)}}$ → inputs to the output layer → $y = a(y^{(in)})$

3) spoč. $\frac{\partial y^{(in)}}{\partial W^{(y)}}$ a $\frac{\partial y^{(in)}}{\partial b^{(y)}}$ → z čehož získáme

$$\frac{\partial L}{\partial W^{(y)}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial y^{(in)}} \cdot \frac{\partial y^{(in)}}{\partial W^{(y)}}$$ a analogicky $\frac{\partial L}{\partial b^{(y)}}$

4) spoč. $\frac{\partial y^{(in)}}{\partial h}$ a $\frac{\partial h}{\partial h^{(in)}}$

5) nakonec použijeme $\frac{\partial h^{(in)}}{\partial W^{(h)}}$ a $\frac{\partial h^{(in)}}{\partial b^{(h)}}$ k $\frac{\partial L}{\partial W^{(h)}}$ a $\frac{\partial L}{\partial b^{(h)}}$

4. Formulate the Universal approximation theorem. [5]

'89

Let $\varphi(x): \mathbb{R} \to \mathbb{R}$ be nonconstant, bounded, nondecreas. continuous function.

For any $\varepsilon > 0$ and any continuous function $f: [0,1]^D \to \mathbb{R}$ there exists $H \in \mathbb{N}$, $v \in \mathbb{R}^H$, $b \in \mathbb{R}^H$, $W \in \mathbb{R}^{D \times H}$ s.t.

$$F(x) = v^T \varphi(x^T W + b) = \sum_{i=1}^{H} v_i \varphi(x^T W_{*i} + b_i)$$

where φ is applied elementwise, then $\forall x \in [0,1]^D$:

$$|F(x) - f(x)| < \varepsilon$$

Lecture 5 Questions

- How do we search for a minimum of a function $f(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ subject to equality constraints $g_1(\boldsymbol{x}) = 0, \ldots, g_m(\boldsymbol{x}) = 0$? [5]

• potřebujeme pár předpokladů:

$f, g_1 - g_m$ mají spojité parciální derivace a gradienty

$\nabla_x g_1(x) - \nabla_x g_m(x)$ jsou lin. nezávislé

→ potom $\exists \lambda_1 \in \mathbb{R} - \lambda_m \in \mathbb{R}$ s.t. Lagrangian function

$$\mathcal{L}(x, \lambda) = f(x) - \sum_{i=1}^{m} \lambda_i g_i(x)$$

má nulový gradient v x a v λ

5• Prove which categorical distribution with N classes has maximum entropy. [5]

→ chceme minimalizovat $-H(p)$ (entropy) za podmínek:

1) $p_i \geq 0 \quad \forall i$

2) $\sum_{i=1}^{N} p_i = 1$

→ prozatím ignorujeme první podmínku a zformulujeme Lagrangian:

$$\mathcal{L} = \left(\sum_i p_i \log p_i\right) - \lambda\left(\sum_i p_i - 1\right)$$

→ spoč. derivace podle $\forall p_i$ a položíme rovno 0:

$$0 = \frac{\partial \mathcal{L}}{\partial p_i} = 1 \cdot \log(p_i) + p_i \cdot \frac{1}{p_i} - \lambda = \log(p_i) + 1 - \lambda$$

→ $p_i = e^{\lambda - 1} \Rightarrow \forall p_i$ jsou stejná a z 2) podmínky:

$$p_i = \frac{1}{n} \quad \forall i \in 1-N \rightarrow P = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)$$

5 • Consider derivation of softmax using maximum entropy principle, assuming we have a dataset of N examples $(x_i, t_i), x_i \in \mathbb{R}^D, t_i \in \{1, 2, \ldots, K\}$. Formulate the three conditions we impose on the searched $\pi : \mathbb{R}^D \to \mathbb{R}^K$, and write down the Lagrangian to be maximized. [10]

chceme minimalizovat: $-\sum_{i=1}^{N} H(\pi(x_i))$

podmínky:

1) $\forall 1 \leq i \leq N, \forall 1 \leq k \leq K : \pi(x_i)_k \geq 0$

2) $\forall 1 \leq i \leq N : \sum_{k=1}^{K} \pi(x_i)_k = 1$

3) $\forall 1 \leq j \leq D, \forall 1 \leq k \leq K : \sum_{i=1}^{N} \pi(x_i)_k \cdot x_{ij} = \sum_{i=1}^{N} [t_i = k] x_{ij}$

→ prozatím ignorujeme 1) a Lagrangian:

$$\mathcal{L} = \sum_{i=1}^{N}\sum_{k=1}^{K} \pi(x_i)_k \cdot \log(\pi(x_i)_k)$$

$$- \sum_{j=1}^{D}\sum_{k=1}^{K} \lambda_{jk} \left(\sum_{i=1}^{N} \pi(x_i)_k x_{ij} - [t_i = k] x_{ij} \right)$$

$$- \sum_{i=1}^{N} \beta_i \left(\sum_{k=1}^{K} \pi(x_i)_k - 1 \right)$$

5. Define precision (including true positives and others), recall, F_1 score, and F_β score (we stated several formulations for F_1 and F_β scores; any one of them will do). [5]

	Target positive	Target negative
Predicted positive	**True Positive (TP)**	**False Positive (FP)**
Predicted negative	**False Negative (FN)**	**True Negative (TN)**

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{precision} = \frac{TP}{TP + FP} \qquad \text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = \frac{2}{\text{precision}^{-1} + \text{recall}^{-1}} = \frac{TP + TP}{TP + FP + TP + FN}$$

$$F_\beta = \frac{1 + \beta^2}{\text{precision}^{-1} + \beta^2 \cdot \text{recall}^{-1}} = \frac{TP + \beta^2 \cdot TP}{TP + FP + \beta^2 (TP + FN)}$$

↳ F_2 favouring recall, $F_{0.5}$ favoring precision

5. Explain the difference between micro-averaged and macro-averaged F_1 scores. [5]

micro F_1: we first sum all the TP, FP and FN of the individual binary classifications and compute the final F_1-score (freq. of individ. classes taken into account)

macro F_1: first compute the F_1-scores of the individual binary classifications and then compute unweighted average (freq. of individual classes is more or less ignored)

5 • Describe k-nearest neighbors prediction, both for regression and classification. Define L_p norm and describe uniform, inverse, and softmax weighting. [5]

regression:

$$t = \sum_i \frac{w_i}{\sum_j w_j} \cdot t_i$$

→ k nearest neighbors have values t_i and weights w_i

classification:

if weights uniform → voting (most frequent wins)

else → we weight the categorical dist. $t_i \in \mathbb{R}^k$ predicting a distribution:

$$t = \sum_i \frac{w_i}{\sum_j w_j} t_i$$

→ predicted class is the one with largest prob.

$$\arg\max_k \sum_i w_i t_{ik}$$

L_p-norm

for $x, y \in \mathbb{R}^D$ the distance: $\|x-y\|_p$ where

$$\|x\|_p = \left(\sum_i |x_i|^p\right)^{1/p}$$

weights:

- uniform: all k nearest neigh. considered equally
- inverse: weight of example is proportional to the inverse of distance
- softmax: weights proportional to the softmax of negative distances

Lecture 6 Questions

- Define a kernel based on a feature map $\varphi : \mathbb{R}^D \to \mathbb{R}^F$, and write down the formulas for (1) a polynomial kernel of degree d, (2) a polynomial kernel of degree at most d, (3) an RBF kernel. [5]

we define a kernel corresponding to a feature map φ as a function:

$$K(x,z) = \varphi(x)^T \varphi(z)$$

1) also called homogenous polynomial kernel

$$K(x,z) = (\mu x^T z)^d$$

↳ feature map returning all combinations of exactly d input features

$$\rightarrow (a_1 + \dots + a_k)^d = \sum_{\substack{n_i \geq 0 \\ \sum n_i = d}} \binom{d}{n_1 \dots n_k} a_1^{n_1} \cdot \dots a_k^{n_k}$$

$$\hookrightarrow \varphi(x) = \left(\sqrt{\mu^d \binom{d}{n_1 \dots n_k}} x_1^{n_1} \dots x_k^{n_k} \right)_{n_i \geq 0,\ \sum n_i = d}$$

2) also nonhomogenous

$$K(x,z) = (\mu x^T z + 1)^d$$

↳ all combinations of up to d input features

$$\rightarrow (\mu x^T z + 1)^d = \sum_i \binom{d}{i} (\mu x^T z)^i$$

$$\hookrightarrow \varphi(x) = \left(\sqrt{\mu^{d-n_{D+1}} \binom{d}{n_1 \dots n_{D+1}}} x_1^{n_1} \cdots x_D^{n_D} \right)_{\substack{n_i \geq 0 \\ \sum_{i=1}^{D+1} n_i = d}}$$

3) Gaussian Radial basis function

$$K(x,z) = e^{-\mu \|x-z\|^2}$$

→ combination of poly. features of all degrees
↳ assuming $\mu = 1$:

6

- Define a kernel and write down the mini-batch SGD training algorithm of dual formulation of kernel linear regression (including the update for the bias). Then describe how are predictions for unseen data made. [10]

Input: Dataset ($\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \in \mathbb{R}^{N \times D}$, $\boldsymbol{t} \in \mathbb{R}^N$), learning rate $\alpha \in \mathbb{R}^+$.

- set $\beta_i \leftarrow 0$
- compute all values $\boldsymbol{K}_{i,j} = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$ (kernel)
- until convergence (or patience runs out), process a minibatch of examples with indices $\boldsymbol{b}$:
  - simultaneously for all $i \in \boldsymbol{b}$ (the β_j on the right side must not be modified during the batch update):
    - $\beta_i \leftarrow \beta_i - \frac{\alpha}{|\boldsymbol{b}|}\left(\left(\sum_j \left(\beta_j \boldsymbol{K}_{i,j}\right)\right) - t_i\right)$

predictions: $y(z) = \varphi(z)^T w = \sum_{i=1}^{N} b_i \varphi(z)^T \varphi(x_i)$

update of bias in SGD:

$$b \leftarrow b - \frac{\alpha}{|b|} \sum_{i \in b} \left(\left(\sum_{j=1}^{N} \beta_j K_{ij}\right) + b - t_i\right)$$

6

- Derive the primary formulation of hard-margin SVM (the value to minimize, the constraints to fulfill) as a maximum-margin classifier (i.e., start by margin maximization). [5]

dataset $X \in \mathbb{R}^{N \times D}$ $t \in \{-1, 1\}^N$, φ feature map

$y(x) = \varphi(x)^T w + b$

- distance of point x_i to decision boundary

$$\frac{|y(x)|}{\|w\|} = \frac{t_i y(x_i)}{\|w\|}$$ assuming class. are correct

↳ want to maximize:

$$\underset{w,b}{\operatorname{argmax}} \frac{1}{\|w\|} \min_i \left[t_i(\varphi(x_i)^T w + b\right] \rightarrow \text{hard to optimize}$$

↳ the model is invariant to mult. w, b by const.
↳ for the points closest to the decision boundary:

$$t_i y(x_i) = 1$$

↳ then for ∀ points we have that $t_i y(x_i) \geq 1$

$$\arg\max_{w,b} \frac{1}{\|w\|} \min_i [t_i(\varphi(x_i)^T w + b]$$

↳ $\arg\min_{w,b} \frac{1}{2}\|w\|$ given that $t_i y(x_i) \geq 1$

6 • How do we search for a minimum of a function $f(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ subject to an inequality constraint $g(\boldsymbol{x}) \geq 0$? Formulate both the variant with KKT conditions and the variant with the λ maximization, and prove that they are equivalent. [10]

predpoklady: f, g maj. spojité parc. derivace a $\frac{\partial g}{\partial x}(x) \neq 0$
potom $\exists \lambda \in \mathbb{R}$ s.t. Lagrangian

$\mathcal{L}(x, \lambda) = f(x) - \lambda g(x)$ has zero gradient in x and

KKT: $g(x) \geq 0$
$\lambda \geq 0$
$\lambda g(x) = 0$

λ-max → if we have min x and λ fullfilling KKT
↳ the $\mathcal{L}$ has maximum in λ subject to $\lambda \geq 0$:

- if $g(x) = 0$ ... the $\mathcal{L}$ doesn't change when changing λ
- if $g(x) > 0$ ... $\lambda = 0$ from KKT → which is max of $\mathcal{L}$

on the other hand → if we have min x and $\mathcal{L}$ has a max in λ subject to $\lambda \geq 0$ → all KKT must hold

- if $g(x) < 0$ ... increasing λ would increase $\mathcal{L}$
- if $g(x) > 0$ .. decreasing λ increases $\mathcal{L}$ so $\lambda = 0$

→ tedy KKT ⇔ $\mathcal{L}$ má max v λ za podmínky $\lambda \geq 0$

6 • Starting from primary hard-margin SVM formulation, derive the dual formulation (the Lagrangian $\mathcal{L}$ in the form used for training, the required conditions, the KKT conditions of the solution, and how is the prediction performed). [10]

chceme minimalizovat: $\underset{w,b}{\operatorname{argmin}} \frac{1}{2}\|w\|^2$ given that $t_i y(x_i) \geq 1$

→ Lagrangian with multipliers $a = (a_1 - a_N)$:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 - \sum_i a_i [t_i y(x_i) - 1]$$

→ setting the derivates with respect to w, b to 0 :

$$w = \sum_i a_i t_i \varphi(x_i) \qquad 0 = \sum_i a_i t_i$$

→ dosazením do $\mathcal{L}$, chceme maximize

$$\mathcal{L} = \sum a_i - \frac{1}{2} \sum_i \sum_j a_i a_j t_i t_j K(x_i, x_j)$$

→ with respect to a_i subject to $a_i \geq 0$ and $\sum_i a_i t_i = 0$
using kernel $K(x, z) = \varphi(x)^T \varphi(z)$

→ the solution will fullfill KKT:

$$a_i \geq 0 \qquad t_i y(x_i) - 1 \geq 0 \qquad a_i (t_i y(x_i) - 1) = 0$$

→ therefore < x_i is on boundary
$a_i = 0$

↳ given that prediction for x is :

$$y(x) = \sum_i a_i t_i K(x, x_i) + b$$

↳ we only need training points x_i that are on the boundary = support vectors

6 • Considering hard-margin SVM, define what is a support vector, and how are predictions performed for unseen data. [5]

→ the solution will fullfill KKT:

$$a_i \geq 0 \qquad t_i y(x_i) - 1 \geq 0 \qquad a_i (t_i y(x_i) - 1) = 0$$

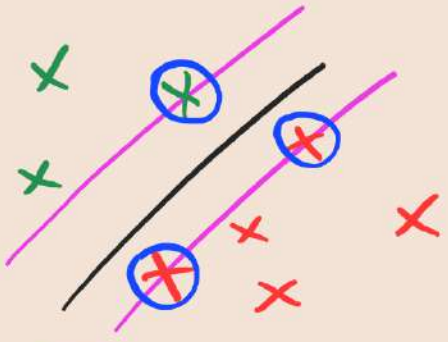
→ therefore < x_i is on boundary
$a_i = 0$

↳ given that prediction for x is:

$$y(x) = \sum_i a_i t_i k(x, x_i) + b$$

↳ we only need training points x_i that are on the boundary = support vectors

⇒ even though SVM is nonparametric model
↳ we only need to store a subset of train data



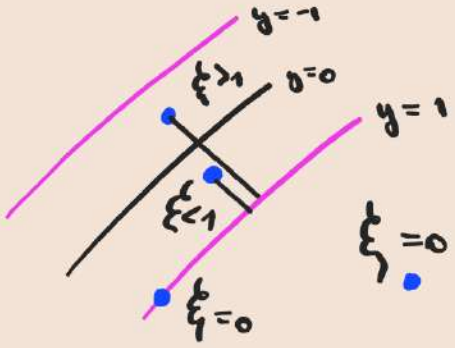
• support vectors → for prediction we use support vectors and the point we want to predict

Lecture 7 Questions

- Write down the primary formulation of soft-margin SVM using the slack variables (the value to minimize, the constraints to fulfill). [5]

slack variables $\xi_i \geq 0$

$$\xi = \begin{cases} 0 & \text{for points fullfilling } t_i y(x_i) \geq 1 \\ |t_i - y(x_i)| & \text{otherwise} \end{cases}$$



⇒ we want to optimize:

$$\underset{w,b}{\operatorname{argmin}} \; \underbrace{C}_{\text{jak moc mě to trápí}} \underbrace{\sum_i \xi_i}_{\text{penalty}} + \frac{1}{2}\|w\|^2$$

↳ given that $t_i y(x_i) \geq 1 - \xi_i$ and $\xi \geq 0$

7
- Starting from primary soft-margin SVM formulation, derive the dual formulation (the Lagrangian $\mathcal{L}$ in the form used for training, the required conditions, the KKT conditions of the solution, and how is prediction performed). [10]

to optimize: $\underset{w,b}{\operatorname{argmin}} \; C \sum_i \xi_i + \frac{1}{2}\|w\|^2$

→ to solve it → create Lagrangian with two sets of multipliers $a = (a_1 \dots a_N)$ and $\mu = (\mu_1 \dots \mu_N)$:

$$\mathcal{L} = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum a_i (t_i y(x_i) - 1 + \xi_i) - \sum_i \mu_i \xi_i$$

→ solving for critical points and substituting w, b:

$$\mathcal{L} = \sum_i a_i - \frac{1}{2}\sum_i \sum_j a_i a_j t_i t_j k(x_i, x_j)$$

↳ which is identical to previous case, but condit. differ: $\forall i : C \geq a_i \geq 0$ and $\sum_i a_i t_i = 0$

→ using KKT conditions → support vectors ($a_i > 0$) are the ones with $t_i y(x_i) = 1 - \xi_i$

↳ on the margin boundary, inside margin, opposite side of decision boundary

prediction (for x): $y(x) = \sum_i a_i t_i k(x, x_i) + b$

7. Write down the primary formulation of soft-margin SVM using the hinge loss. [5]

slack variables can be written as:

$$\xi_i = \max(0, 1 - t_i y(x_i))$$

→ reformulate objective as the hinge loss

$$\mathcal{L}_{hinge}(t, y) = \max(0, 1 - ty)$$

to: $\underset{w,b}{\operatorname{argmin}}\, C\sum_i \mathcal{L}_{hinge}(t_i, y(x_i)) + \frac{1}{2}\|w\|^2$

~ analogous to a regularized loss, where C is inverse regularization $C = \infty$ .. NO regue. $C = 0$ ... ignore data

7. Describe the high-level overview of the SMO algorithm (the test whether the KKT conditions hold, how do we select the a_i and a_j to update, what is the goal of updating the a_i and a_j, how do we detect convergence; but without the update of a_i, a_j, b themselves). [5]

Input: Dataset ($\boldsymbol{X} \in \mathbb{R}^{N \times D}$, $\boldsymbol{t} \in \{-1, 1\}^N$), kernel $\boldsymbol{K}$, regularization parameter C, tolerance tol, $max\_passes\_without\_as\_changing$ value

- Initialize $a_i \leftarrow 0$, $b \leftarrow 0$, $passes \leftarrow 0$
- **while** $passes < max\_passes\_without\_as\_changing$ (or we run out of patience):
  - $changed\_as \leftarrow 0$
  - **for** i in $1, 2, \ldots, N$:
    - $E_i \leftarrow y(\boldsymbol{x}_i) - t_i$
    - **if** ($a_i < C - tol$ **and** $t_i E_i < -tol$) **or** ($a_i > tol$ **and** $t_i E_i > tol$): KKT
      - choose $j \neq i$ randomly
      - try updating a_i, a_j to maximize $\mathcal{L}(a_1, a_2, \ldots, a_N)$ such that $0 \leq a_k \leq C$ and $\sum_i a_i t_i = 0$; if successful, set b fo fulfill the KKT conditions and set $changed\_as \leftarrow changed\_as + 1$
  - $passes \leftarrow 0$ **if** $changed\_as$ **else** $passes + 1$

↳ SMO = coordinated descent

7. • Describe the part of the SMO algorithm which updates a_i and a_j to maximize the Lagrangian. If you explain how the update is derived (so that if I followed the instructions, I would come up with the update rules), you do not need to write explicit formulas. [10]

→ use condition $\sum_k a_k t_k = 0$

$\Rightarrow$ $a_i t_i = - \sum_{k \neq i} a_k t_k$ because $t_i^2 = 1$ →

$a_i = t_i (\delta - a_j t_j)$ where $\delta = - \sum_{\substack{k \neq i \\ k \neq j}} a_k t_k$

↳ maximizing $\mathcal{L}(a)$ with respect to a_j and a_i then amounts to maximizing quadratic fce of a_j, which has analytical solution

(heuristics are usually used for choosing a_i, a_j)

↳ to find a_j maximizing $\mathcal{L}$:

$$a_j^{new} \leftarrow a_j - \frac{\partial \mathcal{L} / \partial a_j}{\partial^2 \mathcal{L} / \partial a_j^2}$$

→ denote $E_j = y(x_j) - t_j$ (jak daleko jsem od margin) → first derivative $\frac{\partial \mathcal{L}}{\partial a_j} = t_j (E_i - E_j)$

↳ second derivative $\frac{\partial \mathcal{L}^2}{\partial a_j^2} = 2k(x_i, x_j) - k(x_i, x_i) - k(x_j, x_j)$

↳ if 2nd der. < 0 → vertex is max

$$a_j^{new} \leftarrow a_j - t_j \frac{E_i - E_j}{2K(x_i, x_j) - K(x_i, x_i) - K(x_j, x_j)}$$

↳ constraints: $0 \le a_i \le C$ and $0 \le a_j \le C$

↳ clip it to range $[L, H]$

$t_i = t_j \Rightarrow L = \max(0, a_i + a_j - C)$

↳ zdůvodnit $H = \min(C, a_i + a_j)$

$t_i \neq t_j \Rightarrow L = \max(0, a_j - a_i)$

↳ / $H = \min(C, C + a_j - a_i)$

→ if we know a_j^{new} → get a_i^{new}

↳ $a_i = -t_i t_j a_j + \text{const}$

$$a_i^{new} \leftarrow a_i - t_i t_j (a_j^{new} - a_j)$$

7 • Describe the part of the SMO algorithm which updates b to maximize the Lagrangian. If you explain how the update is derived (so that if I followed the instructions, I would come up with two b candidates and a rule to utilize them), you do not need to write explicit formulas. [10]

To arrive at the bias update, we consider the KKT condition that for $0 < a_j^{\text{new}} < C$, it must hold that $1 = t_j y(\boldsymbol{x}_j) = t_j \left[\left(\sum_l a_l^{\text{new}} t_l K(\boldsymbol{x}_j, \boldsymbol{x}_l) \right) + b^{\text{new}} \right]$. Combining it with the fact that $\left(\sum_l a_l t_l K(\boldsymbol{x}_j, \boldsymbol{x}_l) \right) + b = E_j + t_j$, we obtain

$$b_j^{\text{new}} = b - E_j - t_i (a_i^{\text{new}} - a_i) K(\boldsymbol{x}_i, \boldsymbol{x}_j) - t_j (a_j^{\text{new}} - a_j) K(\boldsymbol{x}_j, \boldsymbol{x}_j).$$

Analogously for $0 < a_i^{\text{new}} < C$ we get

$$b_i^{\text{new}} = b - E_i - t_i (a_i^{\text{new}} - a_i) K(\boldsymbol{x}_i, \boldsymbol{x}_i) - t_j (a_j^{\text{new}} - a_j) K(\boldsymbol{x}_j, \boldsymbol{x}_i).$$

Finally, if $a_j^{\text{new}}, a_i^{\text{new}} \in \{0, C\}$, we know that all values between b_i^{new} and b_j^{new} fulfill the KKT conditions. We therefore arrive at the following update for bias:

$$b^{\text{new}} = \begin{cases} b_i^{\text{new}} & \text{if } 0 < a_i^{\text{new}} < C, \\ b_j^{\text{new}} & \text{if } 0 < a_j^{\text{new}} < C, \\ (b_i^{\text{new}} + b_j^{\text{new}})/2 & \text{otherwise.} \end{cases}$$

7 • Describe the one-versus-one and one-versus-rest schemes of constructing a K-class classifier by combining multiple binary classifiers. [5]

↳ multiclass SVM

one-versus-one

↳ $\binom{k}{2}$ binary classifiers are constructed, one for each (i, j) pair of class indices

↳ the class with majority of votes wins (used by SVM)
(downside: tied votes)

one-versus-rest

↳ k binary classifiers are constructed
→ the i-th separating instances of class i from all others

↳ during prediction → the one with highest prob. wins

→ bin. classifiers need to return calibrated probabilities (not SVM)

Lecture 8 Questions

- Explain how is the TF-IDF weight of a given document-term pair computed. [5]

$$TF(t,d) = \frac{\text{number of occurences of } t \text{ in } d}{\text{number of terms in } d}$$

$$IDF(t) = \log\left(\frac{\text{number of documents}}{\text{number of documents containing } t \; (\text{optim.} +1)}\right)$$

$$= I(P(d \ni t))$$

t... term (word)

d.. document

TF-IDF = TF · IDF → how important a term is to a document in a corpus

18 • Define conditional entropy, mutual information, write down the relation between them, and finally prove that mutual information is zero if and only if the two random variables are independent (you do not need to prove statements about D_{KL}). [5]

rand vars x, y $\quad x \sim X$, $y \sim Y$

conditional entropy:

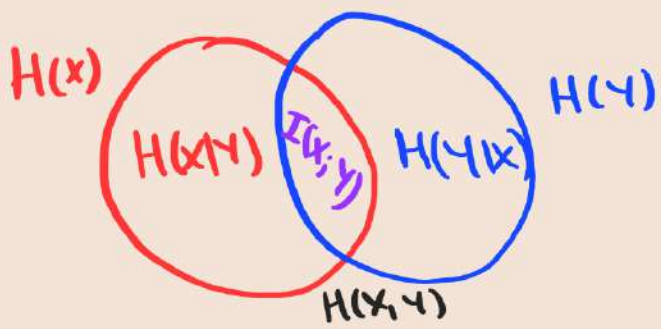
$$H(Y|X) = \mathbb{E}_{x,y}[I(y|x)] = -\sum_{x,y} P(x,y) \log P(y|x)$$

shared info:

$$H(Y) - H(Y|X) = \mathop{\mathbb{E}}_{x,y}[-\log P(y)] - \mathop{\mathbb{E}}_{x,y}[I(y|x)] = \mathop{\mathbb{E}}_{x,y}\left[\log \frac{P(x,y)}{P(x)P(y)}\right]$$

mutual information

$$I(X;Y) = \mathop{\mathbb{E}}_{x,y}\left[\log \frac{P(x,y)}{P(x)\cdot P(y)}\right]$$



(sit's symmetrical:

$$I(X,Y) = I(Y,X) = H(Y) - H(Y|X) = H(X) - H(X|Y)$$

→ zároveň:

$$I(X;Y) = D_{KL}(P(X,Y) \,\|\, P(X)P(Y))$$

⇒ therefore:
- $I(X,Y) \geq 0$ (DKL je taky)
- $I(X,Y) = 0$ iff $P(X,Y) = P(X)\cdot P(Y)$ (independ.)
  ↳ z vlastnosti D_{KL}

18 • Show that TF-IDF terms can be considered portions of suitable mutual information. [5]

→ $\mathcal{D}$ collection of N documents and T collection of terms

→ assumption:

$$P(d) = \frac{1}{|\mathcal{D}|} \text{ and } I(d) = H(\mathcal{D}) = \log|\mathcal{D}|$$

$$\Rightarrow P(d|t) = 1/|\{d \in \mathcal{D} : t \in d\}| \text{ and } I(d|t) = H(\mathcal{D}|T=t) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|}$$

$$\Rightarrow I(d) - I(d|t) = H(\mathcal{D}) - H(\mathcal{D}|T=t) = \log \frac{|\mathcal{D}|}{|\{d \in \mathcal{D} : t \in d\}|} = IDF(t)$$

↳ mutual info $I(\mathcal{D},T)$:

$$I(\mathcal{D},T) = \sum P(d)\,P(t|d)\,(I(d) - I(d|t)) = \frac{1}{|\mathcal{D}|}\sum_{d,t} TF(t,d)\cdot IDF(t)$$

→ summing all TF-IDF terms recovers mutual info between $\mathcal{D}$ and T

→ TF-IDF carries a "bit of info" attached to document-term pair

8 • Show that L^2-regularization can be obtained from a suitable prior by Bayesian inference (from the MAP estimate). [5]

MAP: $w_{MAP} = \arg\max_w p(w|X) = \arg\max_w p(X|w)\,p(w)$

Frequently, the mean is assumed to be zero, and the variance is assumed to be σ^2. Given that we have no further information, we employ the maximum entropy principle, which provides us with $p(w_i) = \mathcal{N}(w_i; 0, \sigma^2)$, so that $p(\boldsymbol{w}) = \prod_{i=1}^{D} \mathcal{N}(w_i; 0, \sigma^2) = \mathcal{N}(\boldsymbol{w}; \boldsymbol{0}, \sigma^2 \boldsymbol{I})$. Then

$$\begin{aligned}\boldsymbol{w}_{\text{MAP}} &= \arg\max_{\boldsymbol{w}} p(\boldsymbol{X}|\boldsymbol{w})p(\boldsymbol{w}) \\ &= \arg\max_{\boldsymbol{w}} \prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{w})p(\boldsymbol{w}) \\ &= \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \Big(-\log p(\boldsymbol{x}_i|\boldsymbol{w}) - \log p(\boldsymbol{w}) \Big).\end{aligned}$$

By substituting the probability of the Gaussian prior, we get

$$\boldsymbol{w}_{\text{MAP}} = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{N} \Big(-\log p(\boldsymbol{x}_i|\boldsymbol{w}) - \frac{D}{2}\log(2\pi\sigma^2) + \frac{\|\boldsymbol{w}\|^2}{2\sigma^2} \Big),$$

which is in fact the L^2-regularization.

8 • Write down how $p(C_k|\boldsymbol{x})$ is approximated in a Naive Bayes classifier, explicitly state the Naive Bayes assumption, and show how is the prediction performed. [5]

assumption: all x_d are independent given C_k

↳ tedy $p(x|C_k) = \prod_{d=1}^{D} p(x_d|C_k)$

→ Bayes formula: $p(C_k|x) = \frac{p(x|C_k)\,p(C_k)}{p(x)}$

↳ můžeme přepsat klasifikaci:

$$\arg\max_k p(C_k|x) = \arg\max_k \frac{p(x|C_k)\,p(C_k)}{p(x)} =$$

$$= \arg\max_k p(x|C_k)\,p(C_k) = \arg\max \prod_{d=1}^{D} p(x_d|C_k)\cdot p(C_k)$$

8 • Considering a Gaussian naive Bayes, describe how are $p(x_d|C_k)$ modeled (what distribution and which parameters does it have) and how we estimate it during fitting. [5]

→ we expect a continuous feature to have normal dist. for a given C_k and model $p(x_d|C_k)$ as a normal dist. $N(\mu_{dk}, \sigma^2_{dk})$

→ X train data, k classes, t targets

↳ "training" = estimating $\mu_{d,k}$ and $\sigma^2_{d,k}$ for $\forall\ 1 \le d \le D$ and $\forall\ 1 \le k \le K$ using MLE

→ fix feature d and class k → $x_1 - x_{N_k}$ train data corresponding to k

MLE ⇒ $\underset{\mu_{d,k},\sigma_{d,k}}{\operatorname{argmin}} \frac{N_k}{2}\log(2\pi\sigma^2_{d,k}) + \sum_{i=1}^{N_k}\frac{(x_{i,d}-\mu_{d,k})^2}{2\sigma^2_{d,k}}$

→ setting derivative with respect to $\mu_{d,k}$ to 0:

$$0 = \sum_{i=1}^{N_k}\frac{-2(x_{i,d}-\mu_{d,k})^2}{2\sigma^2_{d,k}} \Rightarrow \mu_{d,k} = \frac{1}{N_k}\sum_{i=1}^{N_k} x_{i,d}$$

→ setting derivative with respect to $\sigma^2_{d,k}$ to 0:

$$0 = \frac{N_k}{2\sigma^2_{d,k}} - \frac{1}{2(\sigma^2_{d,k})^2}\sum_{i=1}^{N_k}(x_{i,d}-\mu_{d,k})^2$$

$$\Rightarrow \sigma^2_{d,k} = \frac{1}{N_k}\sum_{i=1}^{N_k}(x_{i,d}-\mu_{d,k})^2$$

→ variance is usually smoothed by α

8 • Considering a Multinomial naive Bayes, describe how are $p(\boldsymbol{x}|C_k)$ modeled (what distribution and which parameters does it have) and how we estimate it during fitting. [5]

$p(x|C_k)$ is modeled using multinomial dist.

$p(x|C_k) \propto \prod_d p_{d,k}^{x_d}$

→ rewrite log-likelihood as:

$$\log p(C_k|x) + c = \log p(C_k) + \sum_d x_d \log p_{d,k} = b_k + x^T w_k$$

As in the previous cases, we turn to the maximum likelihood estimation in order to find out the values of $p_{d,k}$. We start with the log-likelihood

$$\sum_{i=1}^{N_k} \log\left(\prod_d p_{d,k}^{x_{i,d}}\right) = \sum_{i,d} x_{i,d} \log p_{d,k}.$$

To maximize this quantity with respect to a probability distribution $\sum_d p_{d,k} = 1$, we need to form a *Lagrangian*

$$\mathcal{L} = \sum_{i,d} x_{i,d} \log p_{d,k} + \lambda\left(1 - \sum_d p_{d,k}\right).$$

Setting the derivative with respect to $p_{d,k}$ to zero results in $0 = \sum_{i=1}^{N_k} \frac{x_{i,d}}{p_{d,k}} - \lambda$, so

$$p_{d,k} = \frac{1}{\lambda}\sum_{i=1}^{N_k} x_{i,d} = \frac{\sum_{i=1}^{N_k} x_{i,d}}{\sum_{i=1}^{N_k}\sum_{d'=1}^{D} x_{i,d'}}, \text{ where } \lambda \text{ is set to fulfill } \sum_d p_{d,k} = 1.$$

Denoting $n_{d,k}$ as the sum of features x_d for a class C_k, the probabilities $p_{d,k}$ could be therefore estimated as

$$p_{d,k} = \frac{n_{d,k}}{\sum_{d'=1}^{D} n_{d',k}}.$$

However, for the same reasons as in the Bernoulli NB case, we also use the Laplace smoothing, i.e., utilize a Dirichlet prior $\mathrm{Dir}(\alpha + 1)$, and instead use

$$p_{d,k} = \frac{n_{d,k} + \alpha}{\sum_{d'=1}^{D}(n_{d',k} + \alpha)} = \frac{n_{d,k} + \alpha}{\left(\sum_{d'=1}^{D} n_{d',k}\right) + \alpha D}$$

with pseudo-count $\alpha > 0$.

8 • Considering a Bernoulli naive Bayes, describe how are $p(x_d|C_k)$ modeled (what distribution and which parameters does it have) and how we estimate it during fitting. [5]

↳ if input features are binary → modeled as bernoulli dist.

$$p(x_d|C_k) = p_{d,k}^{x_d}(1-p_{d,k})^{(1-x_d)}$$

→ rewrite:

$$p(C_k|x) \propto \left(\prod_{d=1}^{D} p_{d,k}^{x_d}(1-p_{d,k})^{(1-x_d)}\right) \cdot p(C_k)$$

→ z logaritm.:

$$\log p(C_k|x) + c = \log p(C_k) + \sum_d \left(x_d \log \frac{p_{d,k}}{1-p_{d,k}} + \log(1-p_{d,k})\right) =$$

$$= b_k + x^T w_k \quad \rightarrow c \text{ doesn't depend on } C_k \Rightarrow \text{not}$$

needed for prediction:

$$\arg\max_k \log p(C_k|x) = \arg\max_k b_k + x^T w_k$$

- probabilities est.: → MLE

To estimate the probabilities $p_{d,k}$, we turn again to the maximum likelihood estimation. The log-likelihood of N_k samples drawn from Bernoulli distribution with parameter $p_{d,k}$ is

$$\sum_{i=1}^{N_k} \log\left(p_{d,k}^{x_{i,d}}(1-p_{d,k})^{1-x_{i,d}}\right) = \sum_{i=1}^{N_k}\left(x_{i,d}\log p_{d,k} + (1-x_{i,d})\log(1-p_{d,k})\right).$$

Setting the derivative with respect to $p_{d,k}$ to zero, we obtain

$$0 = \sum_{i=1}^{N_k}\left(\frac{x_{i,d}}{p_{d,k}} - \frac{1-x_{i,d}}{1-p_{d,k}}\right) = \frac{1}{p_{d,k}(1-p_{d,k})}\sum_{i=1}^{N_k}\Big((1-p_{d,k})x_{i,d} - p_{d,k}(1-x_{i,d})\Big),$$

giving us $p_{d,k} = \frac{1}{N_k}\sum_{i=1}^{N_k} x_{i,d}$.

We could therefore estimate the probabilities $p_{d,k}$ as

$$p_{d,k} = \frac{\text{number of documents of class } k \text{ with nonzero feature } d}{\text{number of documents of class } k}.$$

However, if a feature d is always set to one (or zero) for a given class k, then $p_{d,k} = 1$ (or 0). That is impractical because the resulting classifier would give probability zero to inputs with the opposite value of such a feature.

Therefore, **Laplace** or **additive smoothing** is used, and the probability $p_{d,k}$ estimated as

$$p_{d,k} = \frac{\text{number of documents of class } k \text{ with nonzero feature } d + \alpha}{\text{number of documents of class } k + 2\alpha}$$

for some pseudo-count $\alpha > 0$.

Note that even if this technique has a special name, it corresponds to using a *maximum a posteriori* estimate, using $\mathrm{Beta}(\alpha+1, \alpha+1)$ as a prior distribution.

8 • Describe the difference between a generative and a discriminative model, the strengths of these models, and explain why is logistic regression and multinomial/Bernoulli naive Bayes called a generative-discriminative pair. [5]

→ gen.: modelují bloby kolem toho, kde data jsou pro $\forall$ třídu
↳ odhadují sdruženou distribuci $p(t,x)$
↳ často pomocí Bayes. theoremu
↳ modelují prob. of data being generated by an outcome and only transform it to $p(t|x)$ při predikci

→ disc. : chci říct, kudy vede decision boundary
↳ modelují podmíněnou distribuci $p(t|x)$

- empiricky disc. lepší na klasifikaci
  ↳ modelovat decision boundary lehčí než prob-dist.
- gener. umí rozpoznat anomálie/outliers

gen-disc pair:

- mult. / bernoulli naive Bayes fits $\log p(C_k, x)$ as a linear model
- logistic regression also fits $\log p(C_k|x)$ as a linear model

⟹ called gen-disc. pair
↳ protože jsou tak moc podobné
↳ dělají predikce stejně, jenom se jinak trénují

Lecture 9 Questions

- Prove that independent discrete random variables are uncorrelated. [5]

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

correlation: x, y are uncorrelated if $\text{cov}(x,y) = 0$

↳ if x, y independent:

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

$$= \sum_{x,y} P(x,y)(x-\mathbb{E}(x))(y-\mathbb{E}(y)) \overset{\text{→ independence}}{=}$$

$$= \sum_{x,y} P(x)(x-\mathbb{E}[x]) \cdot P(y)(y-\mathbb{E}[y]) =$$

$$= \sum_{x} P(x)(x-\mathbb{E}[x]) \cdot \sum_{y} P(y)(y-\mathbb{E}[y]) =$$

$$= \mathbb{E}_x[x-\mathbb{E}[x]] \cdot \mathbb{E}_y[y-\mathbb{E}[y]] = 0 \quad \square$$

9 • Write down the definition of covariance and Pearson correlation coefficient ρ, including its range. [5]

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

Pearson correlation coefficient:

$$\rho = \frac{\text{cov}(x,y)}{\sqrt{\text{var}(x)} \cdot \sqrt{\text{var}(y)}}$$

range: $\rho \in [-1,1]$, $\rho^2 \leq 1$

→ population Pcc

or

$$r = \frac{\sum_i (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_i (x_i-\bar{x})^2}\sqrt{\sum_i (y_i-\bar{y})^2}}$$

$r \in [-1,1]$

↳ sample Pcc

9 • Explain how are the Spearman's rank correlation coefficient and the Kendall rank correlation coefficient computed (no need to describe the Pearson correlation coefficient). [5]

Spearman ρ

↳ for nonlinear correlation

↳ it's Pearson corr. coef. measured on ranks of the original data

→ rank of element is its index in sorted asc order

Kendall τ

↳ measures amount of concordant pairs minus the disconcordant pairs

$$\tau = \frac{|\{\text{pairs } i \neq j : x_j > x_i \wedge y_j > y_i\}| - |\{\text{pairs } i \neq j : x_j > x_i \wedge y_j < y_i\}|}{\binom{n}{2}}$$

$$= \frac{\sum_{i<j} \text{sign}(x_j - x_i)\,\text{sign}(y_j - y_i)}{\binom{n}{2}}$$

$\tau \in [-1,1]$

→ no clear consensus on which to use

q
- Considering an averaging ensemble of M models, prove the relation between the average mean squared error of the ensemble and the average error of the individual models, assuming the model errors have zero means and are uncorrelated. [10]

↳ prediction of a model y_i on (x,t) is

$y_i(x) = t + \varepsilon_i(x)$ → model error on x

→ MSE of the model is:

$$\mathbb{E}[(y_i(x)-t)^2] = \mathbb{E}[\varepsilon_i^2(x)]$$

→ considering M models → MSE of ensemble:

$$\mathbb{E}\left[\left(\frac{1}{M}\sum_i \varepsilon_i(x)\right)^2\right]$$

→ individual errors ε_i are uncorrel. and have 0 mean

↳ $\mathbb{E}(\varepsilon_i(x)\cdot\varepsilon_j(x)] = 0$ for $i \neq j$

$$\Rightarrow \mathbb{E}\left[\left(\frac{1}{M}\sum_i \varepsilon_i(x)\right)^2\right] = \mathbb{E}\left[\frac{1}{M^2}\sum_{i,j}\varepsilon_i(x)\cdot\varepsilon_j(x)\right] =$$

$$= \frac{1}{M}\mathbb{E}\left[\frac{1}{M}\sum \varepsilon_i^2(x)\right]$$

⇒ average error of ensemble is $\frac{1}{M}$ times the average error of the individual models

q
- In a regression decision tree, state what values are kept in internal nodes, define the squared error criterion and describe how is a leaf split during training (without discussing splitting constraints). [5]

↳ input dataset $X \in \mathbb{R}^{N\times D}$, $t \in \mathbb{R}^N$

→ at beginning of decision tree: single node

→ I_γ ... set of training example indices in node γ

→ for $\forall$ leaf model predicts average of training examples in the leaf: $\hat{t}_\gamma = \frac{1}{|I_\gamma|}\sum_{i\in I_\gamma} t_i$

- we use criterion c_γ : how uniform examples of γ are

→ sum of squares error between examples in the node and the predicted value in that node:

$$c_{SE}(\mathcal{I}) = \sum_{i \in I_{\mathcal{I}}} (t_i - \hat{t}_{\mathcal{I}})^2 \quad \text{where} \quad \hat{t}_{\mathcal{I}} = \frac{1}{|I_{\mathcal{I}}|} \sum_{i \in I_{\mathcal{I}}} t_i$$

- to split a node
  - ↳ find a feature and its value s.t. when splitting $\mathcal{I}$ into $\mathcal{I}_L$ and $\mathcal{I}_R$ → resulting regions decrease overall criterion value the most
  - ↳ $c_{\mathcal{I}_L} + c_{\mathcal{I}_R} - c_{\mathcal{I}}$ is the lowest

9 • In a K-class classification decision tree, state what values are kept in internal nodes, define the Gini index and describe how is a node split during training (without discussing splitting constraints). [5]

↳ input dataset $X \in \mathbb{R}^{N \times D}$, $t \in \mathbb{R}^N$

→ $I_{\mathcal{I}}$ ... set of training example indices in node $\mathcal{I}$

→ we predict class which is the most frequent in the training examples belonging to leaf $\mathcal{I}$

→ denote average probability for class k in $\mathcal{I}$ as $p_{\mathcal{I}}(k)$

Gini index (=criterion)

↳ how often a randomly chosen element would be incorrectly labeled if it was randomly labeled according to $p_{\mathcal{I}}$

$$c_{Gini}(\mathcal{I}) = |I_{\mathcal{I}}| \sum p_{\mathcal{I}}(k)(1 - p_{\mathcal{I}}(k))$$

- to split a node
  - ↳ find a feature and its value s.t. when splitting $\mathcal{I}$ into $\mathcal{I}_L$ and $\mathcal{I}_R$ → resulting regions decrease overall criterion value the most
  - ↳ $c_{Gini}(\mathcal{I}_L) + c_{Gini}(\mathcal{I}_R) - c_{Gini}(\mathcal{I})$ is the lowest

9 • In a K-class classification decision tree, state what values are kept in internal nodes, define the entropy criterion and describe how is a node split during training (without discussing splitting constraints). [5]

↳ input dataset $X \in \mathbb{R}^{N \times D}$, $t \in \mathbb{R}^N$

→ $I_\mathcal{T}$ ... set of training example indices in node $\mathcal{T}$

→ we predict class which is the most frequent in the training examples belonging to leaf $\mathcal{T}$

→ denote average probability for class k in $\mathcal{T}$ as $p_\mathcal{T}(k)$

Entropy criterion

$$C_{\text{entropy}}(\mathcal{T}) = |I_\mathcal{T}| \cdot H(p_\mathcal{T}) = -|I_\mathcal{T}| \sum_{\substack{k \\ p_\mathcal{T}(k) \neq 0}} p_\mathcal{T}(k) \log p_\mathcal{T}(k)$$

- to split a node
  ↳ find a feature and its value s.t. when splitting $\mathcal{T}$ into $\mathcal{T}_L$ and $\mathcal{T}_R$ → resulting regions decrease overall criterion value the most
  ↳ $C_{\text{entropy}}(\mathcal{T}_L) + C_{\text{entropy}}(\mathcal{T}_R) - C_{\text{entropy}}(\mathcal{T})$ is the lowest

9 • For binary classification, derive the Gini index from a squared error loss. [10]

Recall that $I_\mathcal{T}$ denotes the set of training example indices belonging to a leaf node $\mathcal{T}$, let $n_\mathcal{T}(0)$ be the number of examples with target value 0, $n_\mathcal{T}(1)$ be the number of examples with target value 1, and let $p_\mathcal{T} = \frac{1}{|I_\mathcal{T}|} \sum_{i \in I_\mathcal{T}} t_i = \frac{n_\mathcal{T}(1)}{n_\mathcal{T}(0) + n_\mathcal{T}(1)}$.

Consider sum of squares loss $L(p) = \sum_{i \in I_\mathcal{T}} (p - t_i)^2$.

By setting the derivative of the loss to zero, we get that the p minimizing the loss fulfills $|I_\mathcal{T}| p = \sum_{i \in I_\mathcal{T}} t_i$, i.e., $p = p_\mathcal{T}$.

The value of the loss is then

$$\begin{aligned} L(p_\mathcal{T}) &= \sum_{i \in I_\mathcal{T}} (p_\mathcal{T} - t_i)^2 = n_\mathcal{T}(0)(p_\mathcal{T} - 0)^2 + n_\mathcal{T}(1)(p_\mathcal{T} - 1)^2 \\ &= \frac{n_\mathcal{T}(0) n_\mathcal{T}(1)^2}{\left(n_\mathcal{T}(0) + n_\mathcal{T}(1)\right)^2} + \frac{n_\mathcal{T}(1) n_\mathcal{T}(0)^2}{\left(n_\mathcal{T}(0) + n_\mathcal{T}(1)\right)^2} = \frac{\left(n_\mathcal{T}(1) + n_\mathcal{T}(0)\right) n_\mathcal{T}(0) n_\mathcal{T}(1)}{\left(n_\mathcal{T}(0) + n_\mathcal{T}(1)\right)\left(n_\mathcal{T}(0) + n_\mathcal{T}(1)\right)} \\ &= \left(n_\mathcal{T}(0) + n_\mathcal{T}(1)\right)(1 - p_\mathcal{T}) p_\mathcal{T} = |I_\mathcal{T}| \cdot p_\mathcal{T}(1 - p_\mathcal{T}). \end{aligned}$$

Q9

- For K-class classification, derive the entropy criterion from a non-averaged NLL loss. [10]

Again let $I_\mathcal{T}$ denote the set of training example indices belonging to a leaf node $\mathcal{T}$, let $n_\mathcal{T}(k)$ be the number of examples with target value k, and let $p_\mathcal{T}(k) = \frac{1}{|I_\mathcal{T}|} \sum_{i \in I_\mathcal{T}} [t_i = k] = \frac{n_\mathcal{T}(k)}{|I_\mathcal{T}|}$.

Consider a distribution $\boldsymbol{p}$ on K classes and non-averaged NLL loss $L(\boldsymbol{p}) = \sum_{i \in I_\mathcal{T}} -\log p_{t_i}$.

By setting the derivative of the loss with respect to p_k to zero (using a Lagrangian with constraint $\sum_k p_k = 1$), we get that the $\boldsymbol{p}$ minimizing the loss fulfills $p_k = p_\mathcal{T}(k)$.

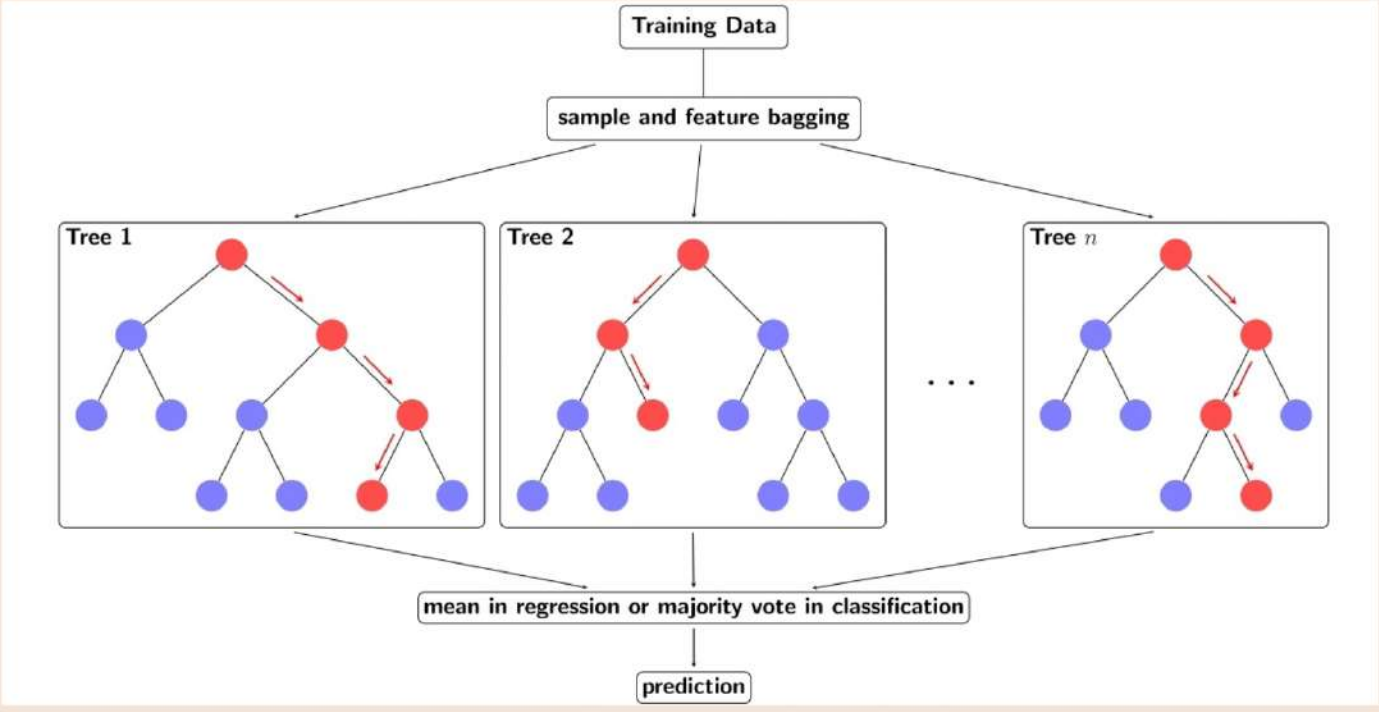
The value of the loss with respect to $\boldsymbol{p}_\mathcal{T}$ is then

$$\begin{aligned} L(\boldsymbol{p}_\mathcal{T}) &= \sum_{i \in I_\mathcal{T}} -\log p_{t_i} \\ &= -\sum_{\substack{k \\ p_\mathcal{T}(k) \neq 0}} n_\mathcal{T}(k) \log p_\mathcal{T}(k) \\ &= -|I_\mathcal{T}| \sum_{\substack{k \\ p_\mathcal{T}(k) \neq 0}} p_\mathcal{T}(k) \log p_\mathcal{T}(k) = |I_\mathcal{T}| \cdot H(\boldsymbol{p}_\mathcal{T}). \end{aligned}$$

Q9

- Describe how is a random forest trained (including bagging and a random subset of features) and how is prediction performed for regression and classification. [5]

- bagging of data combined with random subset of features



- bagging = construct diff. datasets for every model to be trained
  - using bootstrapping = sample as many training instances as the original dataset has but with replacement
    - ↳ every decision tree is trained using bagging

- **random subset of features**
  - during each node split, only a random subset of features is considered when finding the best split
  - a fresh random subset is used for every node

Lecture 10 Questions

- Write down the loss function which we optimize in gradient boosting decision tree during the construction of t^{th} tree. Then define g_i and h_i and show the value $w_{\mathcal{T}}$ of optimal prediction in node $\mathcal{T}$. [10]

loss function:

$$E^{(t)}(w_t; w_{1\_t-1}) = \sum_i \left[l\left(t_i, y^{(t-1)}(x_i; w_{1\_t-1}) + y_t(x_i, w_t)\right)\right] + \frac{\lambda}{2}\|w_t\|^2$$

- $w = (w_1 \dots w_T)$ ... parameters of the trees
- $l(t_i, y(x_i, w))$ .. per-example loss $\rightarrow (t_i - y(x_i, w))^2$ for regres.
- λ ... L^2-regularization strength

g_i and h_i

- in gradient boosting decision trees in second order method:

$$g_i = \frac{\partial l(t_i, y^{(t-1)}(x_i))}{\partial y^{(t-1)}(x_i)} \qquad h_i = \frac{\partial^2 l(t_i, y^{(t-1)}(x_i))}{\partial y^{(t-1)}(x_i)^2}$$

  - 1. derivace loss

→ expand $E^{(t)}$ using second order approx.

$$E^{(t)}(w_t; w_{1\_t-1}) \approx$$

$$\approx \sum_i \left[l(t_i, y^{(t-1)}(x_i)) + g_i y_t(x_i) + \frac{1}{2} h_i y_t^2(x_i)\right] + \frac{1}{2}\lambda \|w_t\|^2$$

→ $I_{\mathcal{T}}$ ... set of training example indices in node $\mathcal{T}$

- denote prediction of $\mathcal{T}$ as $w_{\mathcal{T}}$

↳ $E^{(t)}(w_t; w_{1..t-1}) \approx \sum_i \left[g_i y_t(x_i) + \frac{1}{2} h_i y_t^2(x_i) \right] + \frac{1}{2}\lambda \|w_t\|^2 + \text{const}$

$\approx \sum_j \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\lambda + \sum_{i \in I_j} h_i \right) w_j^2 \right] + \text{const}$

↳ jdu přes listy místo všech příkladů

→ setting derivative with respect to w_j to 0:

$$0 = \frac{\partial E^{(t)}}{\partial w_j} = \sum_{i \in I_j} g_i + \left(\lambda + \sum_{i \in I_j} h_i \right) w_j$$

⇒ optimal w_j^* for node j is:

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\lambda + \sum_{i \in I_j} h_i}$$

10 • Write down the loss function which we optimize in gradient boosting decision tree during the construction of t^{th} tree. Then define g_i and h_i and the criterion used during node splitting. [10]

loss function:

$$E^{(t)}(w_t; w_{1..t-1}) = \sum_i \left[l\left(t_i, y^{(t-1)}(x_i; w_{1..t-1}) + y_t(x_i, w_t)\right) \right] + \frac{\lambda}{2}\|w_t\|^2$$

- $w = (w_1 \dots w_T)$ … parameters of the trees
- $l(t_i, y(x_i, w))$ … per-example loss → $(t_i - y(x_i, w))^2$ for regres.
- λ … L^2-regularization strength

g_i and h_i

↳ in gradient boosting decision trees in second order method:

$$g_i = \frac{\partial l(t_i, y^{(t-1)}(x_i))}{\partial y^{(t-1)}(x_i)} \qquad h_i = \frac{\partial^2 l(t_i, y^{(t-1)}(x_i))}{\partial y^{(t-1)}(x_i)^2}$$

↳ 1. derivace loss

→ optimum weights:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\lambda + \sum_{i \in I_j} h_i}$$

→ I_j ... set of training example indices in node j

→ substitute opt. weights to the loss:

$$E^{(t)}(w^*) \approx -\frac{1}{2} \sum_j \frac{\left(\sum_{i \in I_j} g_i\right)^2}{\lambda + \sum_{i \in I_j} h_i} + \text{const.}$$

= splitting criterion

- How is the learning rate used during training and prediction of a gradient boosting decision tree? [5]

10

shrinkage:

↳ multiply each trained tree by a learning rate α, which reduces the influence of each individual tree and leaves space for future optimization

→ if one model is responsible for most of the work → when we get unseen data → it can be responsible for most of the mistakes too

↳ so we limit it to only eliminate a certain amount of errors

↳ we train it to correct as much as possible and then we take it and scale it

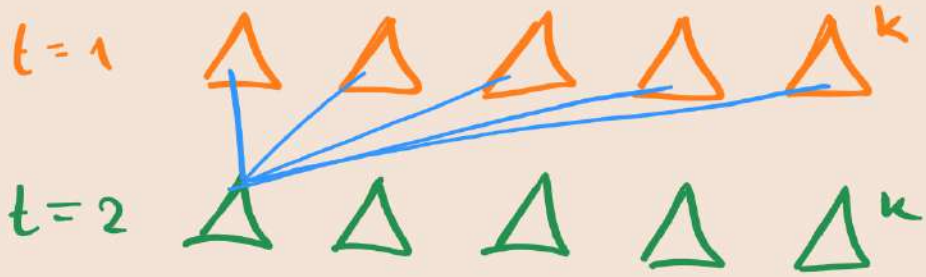
$$y(t) = \sum \alpha \cdot y_i$$

↳ learning rate

→ pomáhá s generalizací

- For a K-class classification, describe how to perform prediction with a gradient boosting decision tree trained for T timestamps (how the individual trees perform prediction and how are the $K \cdot T$ trees combined to produce the predicted categorical distribution). [5]

10



→ we need to model the full categorical output dist.

→ ∀ timestap t we train k trees w_{tk} each predicting a single value of the linear part of a generalized linear model

→ each tree in a timestamp is responsible for saying whether the example belongs to the class or not

→ our trees form a 2D matrix → final prediction is softmax of the sums of the columns

- **prediction:**

$$\text{softmax}(y(x_i)) = \text{softmax}\left(\sum_{t=1}^{T} y_{t,1}(x_i, w_{t,1}) \dots \sum_{t=1}^{T} y_{tk}(x_i, w_{tk})\right)$$

- per-example loss for ∀ k trees is defined as:

$$l(t_i, y(x_i)) = -\log(\text{softmax}(y(x_i))_{t_i})$$

→ so for a tree k at time t:

$$\frac{\partial l(t_i, y^{(t-1)}(x_i))}{\partial y^{(t-1)}(x_i)_k} = \left(\text{softmax}(y^{(t-1)}(x_i)) - 1_{t_i}\right)_k$$

one-hot

- Considering a K-class classification, describe which individual trees (and in which order) are created during gradient boosted decision tree training, and what per-example loss is used for training every one of them (expressed using predictions of the already trained trees). You do not need to describe the training process of the individual trees themselves. [10]

10

→ nevidím rozdíl mezi touhle a předchozí otázkou moc

Lecture 11 Questions

- When deriving the first principal component, write the value of the variance we aim to maximize, both without and with the covariance matrix (and define the covariance matrix). [5]

covariance matrix S:

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})(x_i-\bar{x})^T \qquad \text{where: } \bar{x} = \sum_i \frac{x_i}{N}$$

- matrix form: $S = \frac{1}{N}(X-\bar{x})^T(X-\bar{x})$

→ projection of x_i: $u_1^T x_i$, mean of proj. data: $u_1^T \bar{x}$

→ the variance to maximize:

$$\frac{1}{N}\sum_{i=1}^{N}(u_1^T x_i - u_1^T \bar{x})^2 = u_1^T S u_1$$

u_1 = first principal component = eigenvector

- assume $u_1^T u_1 = 1$

- When deriving the first M principal components, write the value of the reconstruction loss we aim to minimize using all but the first M principal components, both without and with the covariance matrix (and define the covariance matrix). [10]

covariance matrix S:

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})(x_i-\bar{x})^T \qquad \text{where: } \bar{x} = \sum_i \frac{x_i}{N}$$

- matrix form: $S = \frac{1}{N}(X-\bar{x})^T(X-\bar{x})$

- assume $u_1 - u_D$ orthonormal $\Rightarrow u_i^T u_j = [i==j] = \delta_{ij}$

using this basis: $x_i = \sum_j (x_i^T u_j) u_j$

→ approx. data using first M basis vectors

$$\tilde{x}_i = \sum_{j=1}^{M} z_{i,j} u_j + \sum_{j=M+1}^{D} b_j u_j$$

→ minimize approx. error which we measure as loss:

$$L = \frac{1}{N} \sum_{i=1}^{N} \| x_i - \tilde{x}_i \|^2$$

→ derivation + orthogonality:

$$z_{i,j} = x^T u_j \qquad b_j = \bar{x}^T u_j$$

$$\Rightarrow L = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=M+1}^{D} (\bar{x}_i^T u_j - \bar{x}^T u_j)^2 = \sum_{j=M+1}^{D} u_j^T S u_j = \sum_j \lambda_j$$

↳ zahodíme M nejmenších vlastních vektorů a čísel

- Write down the formula for whitening (sphering) the data matrix X, and state what mean and covariance does the result have. [5]

= linear transformation so that resulting dataset has zero mean and identity covariance matrix

- U ... eigenvectors of S (covariance matrix)
- Λ ... diagonal matrix of corresp. eigenvalues

↳ $SU = U\Lambda$ → transformed data:

$$z_i = \Lambda^{-\frac{1}{2}} U^T (x_i - \bar{x}) \qquad \text{where: } \bar{x} = \sum_i \frac{x_i}{N}$$

⇒ mean is zero and covariance is:

$$\frac{1}{N} \sum_i z_i z_i^T = \frac{1}{N} \sum_i \Lambda^{-\frac{1}{2}} (x_i - \bar{x})(x_i - \bar{x})^T U \Lambda^{-\frac{1}{2}} =$$

$$= \Lambda^{-\frac{1}{2}} U^T S U \Lambda^{-\frac{1}{2}} = \Lambda^{-\frac{1}{2}} \Lambda \Lambda \Lambda^{-\frac{1}{2}} = I$$

- Explain how to compute the first M principal components using the SVD decomposition of the data matrix X, and why it works. [5]

→ avoid computing covariance matrix

↳ SVD (singular value decomposition)

$$(X - \bar{x}) = UDV^T \qquad \text{where: } \bar{x} = \sum_i \frac{x_i}{N}$$

↳ $(X-\bar{x})^T(X-\bar{x}) = VDU^T UDV^T = VD^2V^T$

⇒ $(X-\bar{x})^T(X-\bar{x})V = VD^2$

↳ V... eigenvectors of $(X-\bar{x})^T(X-\bar{x})$ and therefore the data covariance matrix S.

↳ eigenvalues of S are squares of the singular values of $(X-\bar{x})$ divided by N

- Write down the algorithm for computing the first M principal components of the data matrix $\boldsymbol{X}$ using the power iteration algorithm. [10]

Input: Matrix $\boldsymbol{X}$, desired number of dimensions M.

- Compute the mean $\boldsymbol{\mu}$ of the examples (the rows of $\boldsymbol{X}$).
- Compute the covariance matrix $S \leftarrow \frac{1}{N}(\boldsymbol{X} - \boldsymbol{\mu})^T(\boldsymbol{X} - \boldsymbol{\mu})$.
- for i in $\{1, 2, \ldots, M\}$:
  - Initialize $\boldsymbol{v}_i$ randomly.
  - Repeat until convergence (or for a fixed number of iterations):
    - $\boldsymbol{v}_i \leftarrow \boldsymbol{S}\boldsymbol{v}_i$
    - $\lambda_i \leftarrow \|\boldsymbol{v}_i\|$
    - $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i / \lambda_i$
  - $\boldsymbol{S} \leftarrow \boldsymbol{S} - \lambda_i \boldsymbol{v}_i \boldsymbol{v}_i^T$
- Return $\boldsymbol{X}\boldsymbol{V}$, where the columns of $\boldsymbol{V}$ are $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_M$.

- Describe the K-means algorithm, including the `kmeans++` initialization. [10]

Input: Input points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, number of clusters K.

- Initialize $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ as K random input points.
- Repeat until convergence (or until patience runs out):
  - Compute the best possible $z_{i,k}$. It is easy to see that the smallest J is achieved by

$$z_{i,k} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2, \\ 0 & \text{otherwise.} \end{cases}$$

  - Compute the best possible $\boldsymbol{\mu}_k = \arg\min_{\boldsymbol{\mu}} \sum_i z_{i,k} \|\boldsymbol{x}_i - \boldsymbol{\mu}\|^2$. By computing a derivative with respect to $\boldsymbol{\mu}$, we get

$$\boldsymbol{\mu}_k = \frac{\sum_i z_{i,k} \boldsymbol{x}_i}{\sum_i z_{i,k}}.$$

where: $J = \sum_{i=1}^{N} \sum_{k=1}^{K} z_{i,k} \| x_i - \mu_k \|^2$

k-means++

↳ init not random
↳ first cluster center randomly
  ↳ others proportionally to the square of their distance to the nearest cluster center

→ solution has O(log k) approx. ratio in expect.

Lecture 12 Questions

- Define the multivariate Gaussian distribution of dimension D. [5]

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

For D-dimensional vector $\boldsymbol{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \stackrel{\text{def}}{=} \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right).$$

Σ ... covariance matrix .. symmetric pos-definite DxD

- Show how to sample from a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ with a full covariance matrix, by using random samples from $\mathcal{N}(0, I)$ distribution. [5]

12

↳ $\Sigma = U \Lambda U^T = (U \Lambda^{1/2})(\Lambda^{1/2} U)^T$

$\Rightarrow x \sim N(\mu, \Sigma) \iff x \sim \mu + U\Lambda^{1/2} N(0, I)$

↳ applying scaling by the eigenvalues
↳ rotating according to eigenvectors
↳ shifting by mean μ

- Describe the constant surfaces of a multivariate Gaussian distribution with (1) $\sigma^2 I$ covariation, (2) a diagonal covariation matrix, (3) a full covariation matrix. [5]

12

1) $\sigma^2 I$ .. single parameter
↳ surfaces are cocentric hyperspheres
(2D: circles, 3D: spheres)

2) Λ ... D parameters
↳ surfaces are axis-aligned hyperellipsoids centered at μ with axes depending on digonal entries
(2D: ellipses, 3D: hyperellipses)

3) Σ ... $\binom{D}{2}$ parameters .. $\theta(D^2)$
↳ surfaces are hyperellipsoids centered a μ but also rotated so that their axes are eigenvectors u_i with sizes $\lambda_i^{1/2}$

12 • Considering a Gaussian mixture with K clusters, explain how we represent the individual clusters and write down the likelihood of an example $\boldsymbol{x}$ for a given Gaussian mixture. [5]

→ we represent data as Gaussian mixture
= combination of k Gaussians

$$p(x) = \sum_{k=1}^{k} \pi_k N(x; \mu_k, \Sigma_k)$$

↳ each cluster is parametrized as $N(x, \mu_k, \Sigma_k)$
→ π_k ... priors representing "fertility" of the cluster
→ let $z \in \{0,1\}^k$ ... k dimensional rand. var. s.t.
↳ to which cluster example belongs
↳ $p(z_k = 1) = \pi_k \rightarrow p(z) = \pi_k \pi_k^{z_k}$

$$\Rightarrow p(x) = \sum_z p(x, z) = \sum_z p(z) \cdot p(x|z) = \sum_{k=1}^{k} \pi_k N(x; \mu_k, \Sigma_k)$$

12 • Write down the log-likelihood of an N-element dataset for a given Gaussian mixture model with K components. [5]

-> log-probability of clustering:

$$\log p(X \mid \pi, \mu, \Sigma) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i, \mu_k, \Sigma_k) \right)$$

distribuce * vektor μ_i clusteru * kovariancni matice

* X... input data

π... $\pi_k = \frac{1}{N} \sum_i r(z_{ik})$

μ ... $\mu_k = \frac{\sum_i r(z_{ik}) x_i}{\sum_i r(z_{ik})}$

* could be unnecessary for this question

Σ ... $\Sigma_k = \frac{\sum_i r(z_{ik})(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r(z_{ik})}$

$r(z_{ik})$ ... responsibility ... $= \frac{\pi_k \mathcal{N}(x_i, \mu_k, \Sigma_k)}{\sum_l \pi_l \mathcal{N}(x_i, \mu_l, \Sigma_l)}$ *

12 • Considering the algorithm for Gaussian mixture clustering, write down the E step (how to compute the responsibilities) and the M step (how to update the means, covariances and priors of the individual clusters). [10]

Input: Input points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, number of clusters K.

- Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ and π_k. It is common to start by running the K-Means algorithm to obtain $z_{i,k}$, set $r(z_{i,k}) \leftarrow z_{i,k}$ and use the **M step** below.
- Repeat until convergence (or until patience runs out):
  - **E step**. Evaluate the responsibilities as

$$r(z_{i,k}) = \frac{\pi_k \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)} = p(z_k = 1 | \boldsymbol{x}_i, \mu, \Sigma, \pi).$$

  - **M step**. Maximize the log-likelihood by setting

$$\boldsymbol{\mu}_k = \frac{\sum_i r(z_{i,k}) \boldsymbol{x}_i}{\sum_i r(z_{i,k})}, \quad \boldsymbol{\Sigma}_k = \frac{\sum_i r(z_{i,k})(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T}{\sum_i r(z_{i,k})}, \quad \pi_k = \frac{\sum_i r(z_{i,k})}{N}.$$

12 • Write down the MSE loss of a regression problem, and formulate the bias-variance trade-off, i.e., the decomposition of expected MSE loss (with respect to a randomly sampled test set) into bias, variance and irreducible error terms. [10]

MSE:

$$L = \mathop{\mathbb{E}}_{x,t}\left[(y(x)-t)^2\right]$$

- denote $g(x) = \mathop{\mathbb{E}}_{t|x}[t]$

↳ $(y(x)-t)^2 = (y(x) - g(x) + g(x) - t)^2 =$

$= (y(x)-g(x))^2 + 2(y(x)-g(x))\underbrace{(g(x)-t)}_{0} + (g(x)-t)^2 =$

$= (y(x)-g(x))^2 + (g(x)-t)^2$

irreducible error

$\rightarrow L = \mathop{\mathbb{E}}_{x,t}\left[(y(x)-g(x))^2\right] + \mathop{\mathbb{E}}_{x,t}\left[(g(x)-t)^2\right]$

→ decompose ⌊ → $\mathcal{D}$ dataset from data-generating dist. → prediction of a model $y(x;\mathcal{D})$

↳ $(y(x;\mathcal{D}) - g(x))^2 = (y(x;\mathcal{D}) - \mathop{\mathbb{E}}_{\mathcal{D}}[y(x;\mathcal{D})] + \mathop{\mathbb{E}}_{\mathcal{D}}[y(x;\mathcal{D})] - g(x))^2 =$

$= (y(x;\mathcal{D}) - \mathop{\mathbb{E}}_{\mathcal{D}}[y(x;\mathcal{D})])^2 +$

$+ 2\underbrace{(y(x;\mathcal{D}) - \mathop{\mathbb{E}}_{\mathcal{D}}[y(x;\mathcal{D})])}_{\mathbb{E}=0}(\mathop{\mathbb{E}}_{\mathcal{D}}[y(x;\mathcal{D})] - g(x)) +$

$+ (\mathbb{E}[y(x;\mathcal{D}) - g(x)])^2$

$\Rightarrow \mathbb{E}[(y(x;\mathcal{D}) - g(x))^2] = \mathbb{E}[(y(x;\mathcal{D}) - \mathop{\mathbb{E}}_{\mathcal{D}}[y(x;\mathcal{D})])^2] +$

$+ (\mathbb{E}[y(x;\mathcal{D}) - g(x)])^2$

$\Rightarrow \mathop{\mathbb{E}}_{\mathcal{D}}[L] = \mathop{\mathbb{E}}_{\mathcal{D}}\left[\mathop{\mathbb{E}}_{x,t}\left[(y(x)-t)^2\right]\right] =$

bias-variance tradeoff ↙

$$= \mathbb{E}_{x,t}\Big[\underbrace{\left(\mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x};\mathcal{D})] - g(\boldsymbol{x})\right)^2}_{\text{bias}^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(y(\boldsymbol{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[y(\boldsymbol{x};\mathcal{D})]\right)^2\right]}_{\text{variance}} + \underbrace{\left(g(\boldsymbol{x}) - t\right)^2}_{\text{irreducible error}}\Big].$$

Lecture 13 Questions

- Considering statistical hypothesis testing, define type I errors and type II errors (in terms of the null hypothesis). Finally, define what a significance level is. [5]

In this setting, H_0 is "not guilty" and H_1 is "guilty".

	H_0 **is true** **Truly not guilty**	H_1 **is true** **Truly guilty**
Not proven guilty Not rejecting H_0	Correct decision True negative	Wrong decision False negative **Type II Error**
Proven guilty Rejecting H_0	Wrong decision False positive **Type I Error**	Correct decision True positive

Our goal is to limit the Type 1 errors – the test **significance level** is the type 1 error rate.

13. Explain what a test statistic and a p-value are. [5]

test statistic:

↳ some summary of the observed data
→ very often single value (like μ)
→ can be used to distinguish H_0 and alt. hypoth.

p-value:

↳ the prob. of obtaining test statistic value at least as extreme as the one actually observed → assuming validity of H_0
→ very small p-value ⇒ observed data are very unlikely under the null hypothesis

13. Write down the steps of a statistical hypothesis test, including a definition of a p-value. [5]

p-value:

↳ the prob. of obtaining test statistic value at least as extreme as the one actually observed → assuming validity of H_0
→ very small p-value ⇒ observed data are very unlikely under the null hypothesis

0. určit úroveň signifikance α
1. zformuluj H_0, volitelně H_1
2. vyber test statistic
3. vypočítej observed value of the test statistic
4. vypočítej p-value
5. zamítni H_0 (in favor of H_1) if p-value is less than α ($\approx$ standard 5%)

13 • Explain the differences between a one-sample test, two-sample test, and a paired test. [5]

There are several kinds of test statistics:

- **one-sample tests**, where we sample values from one distribution.

  Common one-sample tests usually check for
  - the mean of the distribution to be greater than/lower than/equal to zero;
  - the goodness of fit (that the data comes from a normal or categorical distribution of given parameters).
- **two-sample tests**, where we sample independently from two distributions.
- **paired tests**, in which case we also sample from two distributions, but the samples are paired (i.e., evaluating several models on the same data).

  In paired tests, we usually compute the difference between the paired members and perform a one-sample test on the mean of the differences.

↳ když má někdo velkou ruku → má i velkou nohu (spolu souvisí)
↳ bereme je ale vždy od jednoho člověka

13 • When considering multiple comparison problem, define the family-wise error rate, and prove the Bonferroni correction, which allows limiting the family-wise error rate by a given α. [5]

$$FWER = P\left(\bigcup_i (p_i \leq \alpha)\right)$$

B. correction:
↳ rejects H_0 of a test in the family of size m when $p_i \leq \frac{\alpha}{m}$

→ Boole's inequality $P(\bigcup_i A_i) \leq \sum_i P(A_i)$

↳ FWER $= P(\bigcup_i (p_i \leq \frac{\alpha}{m})) \leq \sum_i P(p_i \leq \frac{\alpha}{m}) = m \cdot \frac{\alpha}{m} = \alpha$

→ there exist better corrections

- For a trained model and a given test set with N examples and metric E, write how to estimate 95% confidence intervals using bootstrap resampling. [5]

Bootstrap resampling

Input: Test set $\{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, model predictions $\{y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N)\}$, metric E, number of resamplings R.

Output: R samples of model performance.

- $\mathrm{performances} \leftarrow [\,]$
- repeat R times:
  - sample N test set examples with replacements, together with corresponding model predictions
  - measure the performance of the sampled data using the metric E, and append the result to $\mathrm{performances}$

→ when given the empiracal distribution of model performances produced by
↳ we can estimate 95% conf. int. as a range from the 2.5th percentile and 97.5 percentile of the empirical dist.

- For two trained models and a given test set with N examples and metric E, explain how to perform a paired bootstrap test that the first model is better than the other. [5]

Paired Bootstrap Resampling

Input: Test set $\{(\boldsymbol{x}_1, t_1), \ldots, (\boldsymbol{x}_N, t_N)\}$, model predictions $\{y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_N)\}$, model predictions $\{z(\boldsymbol{x}_1), \ldots, z(\boldsymbol{x}_N)\}$, metric E, number of resamplings R.

Output: Estimated probability of the model y performing worse or equal to z (beware that such a quantity is not a p-value).

- $\mathrm{differences} \leftarrow [\,]$
- repeat R times:
  - sample N test set examples with replacements, together with the corresponding predictions of the models
  - measure the performances of the models y and z on the sampled data using the metric E, and append their difference to $\mathrm{differences}$
- return the ratio of the $\mathrm{differences}$ which are less than or equal to zero

Even if a two-sample test could be used, such a test does not consider the fact that some of the inputs might be more difficult than others, and takes into account cases when a weaker model achieves higher performance on a simpler test set than a stronger model on a more difficult test set. Therefore, we perform a **paired** test.

Our alternative hypothesis is that the mean of the model performance differences is larger than zero, and the null hypothesis is that it is less than or equal to zero. We then repeatedly sample a test set with repetition, and compute the difference of the model performances on the sampled test set. Finally, we compute the quantile of the performance difference 0.

Unfortunately, the value returned by the algorithm is not really a p-value.

The reason is that the distribution of differences was obtained **under the true distribution**.

However, to perform the statistical test, we require the distribution of the test statistic **under the null hypothesis**.

Nevertheless, you can encounter such paired bootstrap tests "in the wild".

13

- For two trained models and a given test set with N examples and metric E, explain how to perform a random permutation test that the first model is better than the other with a significance level α. [5]

To obtain a principled p-value for a model comparison, we can turn to a **permutation test**.

The main idea is that

> *If the models are equally good, it does not matter if we utilize predictions from the first or the second one.*

Therefore, if we consider all possible choices of prediction origins, we obtain a distribution of performances under the hypothesis that the models are equally good.

Finally, the p-value is the quantile of the performance of the model in question.

Of course, enumerating all assignments is not feasible. Therefore, we sample only some number of random assignments, resulting in a **random** or **Monte Carlo** or **approximate** permutation test.

useful

For $y(\boldsymbol{x}; \boldsymbol{w}, b) \stackrel{\text{def}}{=} \boldsymbol{\varphi}(\boldsymbol{x})^T \boldsymbol{w} + b$, we have seen the following losses:

Model	Objective	Per-Instance Loss
Linear Regression	$\arg\min_{\boldsymbol{w},b} \sum_i \mathcal{L}_{\text{MSE}}\big(t_i, y(\boldsymbol{x}_i)\big) + \frac{1}{2}\lambda\lvert\boldsymbol{w}\rvert^2$	$\mathcal{L}_{\text{MSE}}(t,y) = \frac{1}{2}(t-y)^2$
Logistic regression	$\arg\min_{\boldsymbol{w},b} \sum_i \mathcal{L}_{\sigma\text{-NLL}}\big(t_i, y(\boldsymbol{x}_i)\big) + \frac{1}{2}\lambda\lvert\boldsymbol{w}\rvert^2$	$\mathcal{L}_{\sigma\text{-NLL}}(t,y) = -\log \begin{pmatrix} \sigma(y)^t \cdot \\ \cdot \big(1-\sigma(y)\big)^{1-t} \end{pmatrix}$
Softmax regression	$\arg\min_{\boldsymbol{W},\boldsymbol{b}} \sum_i \mathcal{L}_{\text{s-NLL}}\big(t_i, \boldsymbol{y}(\boldsymbol{x}_i)\big) + \frac{1}{2}\lambda\lvert\boldsymbol{w}\rvert^2$	$\mathcal{L}_{\text{s-NLL}}(t,\boldsymbol{y}) = -\log \operatorname{softmax}(\boldsymbol{y})_t$
SVM	$\arg\min_{\boldsymbol{w},b} C\sum_i \mathcal{L}_{\text{hinge}}\big(t_i, y(\boldsymbol{x}_i)\big) + \frac{1}{2}\lvert\boldsymbol{w}\rvert^2$	$\mathcal{L}_{\text{hinge}}(t,y) = \max(0, 1-ty)$

Note that $\mathcal{L}_{\text{MSE}}(t,y) \propto -\log\big(\mathcal{N}(t; \mu = y, \sigma^2 = \text{const})\big)$ and $\mathcal{L}_{\sigma\text{-NLL}}(t,y) = \mathcal{L}_{\text{s-NLL}}(t,[y,0])$.

Name	Activation	Distribution	Loss	Gradient
linear regression	identity	Normal	$\text{NLL} \propto \text{MSE}$	$\big(y(\boldsymbol{x}) - t\big)\boldsymbol{x}$
logistic regression	$\sigma(\bar{y})$	Bernoulli	$\text{NLL} \propto \mathbb{E} - \log(p(t\lvert\boldsymbol{x}))$	$\big(y(\boldsymbol{x}) - t\big)\boldsymbol{x}$
multiclass logistic regression	$\operatorname{softmax}(\bar{\boldsymbol{y}})$	categorical	$\text{NLL} \propto \mathbb{E} - \log(p(t\lvert\boldsymbol{x}))$	$\big(\boldsymbol{y}(\boldsymbol{x}) - \mathbf{1}_t\big)\boldsymbol{x}^T$ wrt $\boldsymbol{W}^T \in \mathbb{R}^{K\times D}$
Poisson regression	$\exp(\bar{y})$	Poisson	$\text{NLL} \propto \mathbb{E} - \log(p(t\lvert\boldsymbol{x}))$	$\big(y(\boldsymbol{x}) - t\big)\boldsymbol{x}$

navigation

1) linear regression
2) regularization | SGD | lin. reg. SGD
3) perceptron | MLE | logistic regression
4) gen. lin. models | MSE as MLE | multiclass log. reg |
   MLP | Universal aprox
5) Lagrange | softmax | F-score | KNN
6) Kernel lin. reg. | SVM | KKT
7) soft SVM | Hinge | SMO | multi SVM
8) TF-IDF | Mutuall info | Bayes | MAP |
   Naive Bayes | gen. vs. disc.
9) Covariance | Correlation | model combin. |
   Decision Tree | Gini & Entropy | Random 
10) Gradient boosting | GB training + classification
11) PCA | Whitening | PCA-SVD | power iter. |
    Clustering | K-means
12) Multivariate Gaussian | Gaussian Mixture |
    Bias-var tradeoff
13) statistics | multiple comparisons | FWER |
    resampling | permutation test