1)

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

L^2-regularizace: „menší model" ~ „lepší model", takže dává pokutu za hodně vysoký váhy

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

Explicit solution of CR: – derivace podle j-té váhy

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

takže chci $\forall j \sum_i^N x_{ij}(x_i^T w - t_i) = X_{*,j}^T(Xw - t) = 0$ ~ pro všechny j

$$X^T(Xw - t) = 0$$

$$X^T X w = X^T t$$

pokud $X^T X$ reg., existuje inverze

$$w = (X^T X)^{-1} X^T t$$

2.1)

SGD:

Snažím se najít nejlepší váhy inkrementálním způsobem.

– chci minimalizovat error fci $\arg\min_w E(w)$, pak gradient descent vypadá jako: $w = w - \alpha \nabla_w E(w)$ (α → learning-rate)

Mějme $X \in \mathbb{R}^{N \times D}$, $t \in \mathbb{R}^N$ trénovací data, $\hat{p}(x,t) = \frac{|\{i : (x,t) = (x_i, t_i)\}|}{N}$,

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

a) Standard GD: využitím všechny trénovací data na výpočet $\nabla_w E(w)$

b) Stochastic GD: odhadnu $\nabla_w E(w)$ pomocí náhodného vzorku. Unbiased, noisy.

$\to \nabla_w E(w) \approx \nabla_w L(y(x, w), t)$ pro náhodné (x,t) z $\hat{p}$

c) Minibatch GD: funkce $\nabla_w E(w)$ dáším ze sampla B hodnot z trénink.

$\to \nabla_w E(w) \approx \frac{1}{B}\sum_i^B \nabla_w L(y(x_i, w), t_i)$ pro náhodné (x_i, t) z $\hat{p}$

2.2)

L^2-reg. minibatch SGD, $X \in \mathbb{R}^{N \times D}$, $t \in \mathbb{R}^N$, learning rate $\alpha \in \mathbb{R}^+$, L^2-síla $\lambda \in \mathbb{R}$

$w = 0$ nebo random

until convergence:

- sample minibatch
- $w = w - \alpha \cdot \sum_i \frac{1}{|b|} ((x_i^T w - t_i) x_i - \alpha \lambda w$

2.3)

SGD, always returns the best?

Mějme learning rates α_i, zašuměný odhad $J(w)$ reálného $\nabla_w E(w)$:

$$w_{i+1} = w_i - \alpha_i J(w_i)$$

Pokud je ztr. fce L konvexní a spojitá, SGD konverguje do unikátního optimu skoro jistě, pokud learning rates splňují:

$$\forall i\ \alpha_i > 0;\ \sum_i a_i = \infty;\ \sum_i \alpha_i^2 < \infty$$

- 3. podmínka implikuje $\alpha_i \to 0$

- pro nekonvexní L lze garantovat pouze lokální optimum.

2.4) SGD: after-train

Slyšeli jsem s: nízkou trénovací chybou, vysokou testovací chybou. Co dělat?

-> Buď začnu regularizovat nebo zastavím train dříve v „test ubliž"

2.5) Fixed train and test, report model perform.

Jeden z hyperparametrů by byla L2 regularizace.

Pokud by šlo o LR, další by byl maximální stupeň polynomu,

pokud o CD, velikost minibatche, max. počet iterací

2.6) Feature engineering:

1) MinMax scaling: $s(x) = \frac{x - min}{max - min}$

2) Standart scaling: $s(x) = \frac{x - \mu}{\sigma}$

Featury v rozdílných rozsazích by potřebovaly jiné learning rates

3.1) Binary classification:

Binární klasifikace: rozdělení vstupních dat do dvou tříd.

- lze dělat pomocí LR na základě tresholdu.

Perceptron alg: $(X \in \mathbb{R}^{N \times D}, t \in \{-1, +1\}^N)$, linear-separable

Výstup: $w \in \mathbb{R}^D$, s.t. $t_i x_i^T w > 0 \ \forall i$

alg:

$w = 0$

until all items correctly qualified:

$y = x_i^T w$ - prediction

if $t_i y \leq 0$: - wrong prediction

$w = w + t_i x_i$

3.2) Entropy

Množství překvapení v celé distribuci. $I(x)$... kolik informace v tom je

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

Pro diskrétní P:

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

Cross-Entropy:

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

Gibbs-Inequality:

$H(P, Q) \geq H(P)$

$H(P) = H(P, Q) \iff P = Q$

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

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

KL-divergence:

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

Dk: Gibbs

$\log x \leq (x-1)$, $\log x = x$ pouze pro $x=1$

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

Součet přes všechny prvky musí dát 1.

$$\sum_x P(x) \log \frac{Q(x)}{P(x)} \leq \sum_x P(x) \left(\frac{Q(x)}{P(x)} - 1\right) = \sum_x Q(x) - \sum_x P(x) = 0$$

Aby rovnost platila, musí být $\frac{Q(x)}{P(x)} = 1$ $\forall x$, tedy $P = Q$

Gibbs $\Rightarrow$ KL:

$D_{KL}(P \| Q) \geq 0$, $D_{KL}(P \| Q) = 0 \iff P = Q$

3.3) likelihood in MLE:

Jde o funkci, která mi říká, s jakou pravděpodobností vidím daná data, mám-li dané parametry modelu. Tedy $p(x_i; w)$ je pravděpodobnost, že vidím data point x_i, pokud mám model parametry w. Nejde o distribuci, přesto se snažíme $L(w)$ maximalizovat.

$$L(w) = p(\mathbf{X}, w) = \prod_{i=1}^{N} p(\mathbf{x}_i, w) \quad \sim \text{likelihood, není to distr.}$$

3.4) MLE jako min NLL, CE, KL-div, určete vztahy

MLE w:

"predict t when x given"

$$w_{MLE} = \arg\max_w p(t|X, w) = \arg\max_w \prod_{i=1}^{N} p(t|x_i, w)$$

$$= \arg\min_w \sum_i^N - \log p(t|x_i, w) \quad \text{NLL}$$

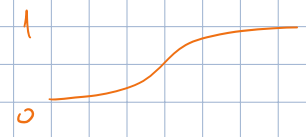
$$= \arg\min_w \mathbb{E}_{x \sim \hat{p}} [-\log p(t|x, w)]$$

$$= \arg\min_w H(\hat{p}(x), p(t|x, w)) \quad \text{CE}$$

$$= \arg\min_w D_{KL}(\hat{p}(x) \| p(t|x, w)) \quad \text{KL-div}$$

3.5) Logistic regression:

sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}} \in (0,1) \ \forall x$:



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

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

$\vdots$

$\bar{y}(x,w) = x^T w \Rightarrow y(x,w) = \sigma(\bar{y}(x,w)) = \sigma(x^T w)$ → jde o naší klasifikaci lin. predikcí

logit: $\bar{y}(x,w) = \log\left(\frac{p(C_1,w)}{p(C_0,w)}\right)$

3.6) L^2 mini-batch SGD

alg:

$w = 0$

until convergence: process a batch $\mathcal{B}$

$$g = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_w \left(-\log\left(p(C_{t_i}|x_i,w)\right)\right) + \lambda w$$

$$w = w - \alpha g$$

loss function:

$$\frac{1}{N} \sum_i -\log\left(p(C_{t_i}|x_i,w)\right) + \frac{\lambda}{2}\|w\|^2$$

gradient:

$$(y(x) - t)\,x$$

4.1) MSE jako MLE:

MSE: $\underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_i (y(x_i,w) - t_i)^2$

Vezmeme distr. s nejvyšší entropií: normální rozdělení $\mathcal{N}(t_i; y(x,w), \sigma^2)$

s použitím MLE:

$$\underset{w}{\operatorname{argmax}}\ p(t|X,w) = \underset{w}{\operatorname{argmin}} \sum_i^N -\log p(t_i|x_i,w)$$

$$= \underset{w}{\operatorname{argmin}} - \sum_i^N \log\left(\sqrt{\frac{1}{2\pi\sigma^2}} \cdot e^{-\frac{(t_i - y(x_i,w))^2}{2\sigma^2}}\right)$$

$$= \underset{w}{\operatorname{argmin}} -N\left(\log(2\pi\sigma^2)^{-\frac{1}{2}} - \sum_i^N -\frac{(t_i - y(x_i,w))^2}{2\sigma^2}\right.$$

$$= \underset{w}{\operatorname{argmin}} \sum_i^N \frac{(t_i - y(x_i,w))^2}{2\sigma^2} = \underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_i^N (y(x_i,w) - t_i)^2$$

4.2) Extenze logist. regrese:

váhy pro každou třídu: $W \in \mathbb{R}^{D \times K}$

softmax:

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

alg: opět je predikce lin. část pohmnná softmaxem.

$$p(C_i | x_i, W) = y(x_i, W) = \text{Softmax}(\bar{y}(x_i, W))_i = \text{Softmax}(x^T W)_i$$

$$\bar{y}(x_i, w) = \log(p(C_i | x_i, w)) + c$$

4.3) Sigmoid vs. Softmax

Sigmoid je speciální případ softmaxu:

$$\sigma(x) = \text{softmax}([x, 0])_0 = \frac{e^x}{e^x + e^0} = \frac{1}{1+e^{-x}}$$

4.4) L^2 minibatch SGD pro K-label

alg: $X \in \mathbb{R}^{N \times D}$, $t \in \{0 \dots K-1\}^N$, learning rate $\alpha \in \mathbb{R}^+$

$w := W$

$w = 0$

until convergence: process minibatch B

$$g = \frac{1}{|B|} \sum_{i \in B} \nabla_w \left(-\log(p(C_{t_i} | x_i, w)) \right)$$

$$w = w - \alpha g$$

gradient:

$$(y(x) - 1_t)\, x^T$$

loss function:

$$\frac{1}{N} \sum_i -\log(p(C_{t_i} | x_i, w))$$

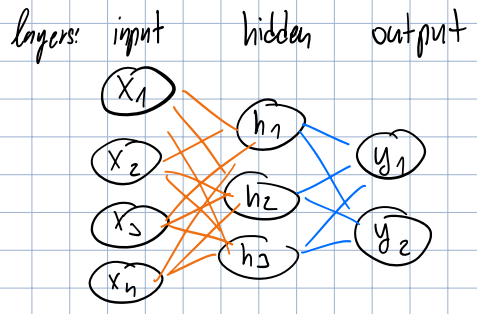
4.5) multilabel, convex decision regions

Rozhodovací regiony jsou konvexní. Uvažme dva body stejné kategorie x_a a x_b. Libovolný bod ležící na přímce $x = \lambda x_a + (1-\lambda) x_b$ je jejich konvexní kombinací díky linearitě:

$$\bar{y}(x) = x^T W \Rightarrow \bar{y}(x) = \lambda \bar{y}(x_a) + (1-\lambda) \bar{y}(x_b)$$

Tím, že $\bar{y}(x_a)$ i $\bar{y}(x_b)$ měli největší k-tou složku, bude ji mít nejvíce i $\bar{y}(x)$.

4.6) MLP



$$h_i = f\left(\sum_j x_j w_{j,i}^{(h)} + b_i^{(h)}\right)$$

$$y_i = a\left(\sum_j h_j w_{j,i}^{(y)} + b_i^{(y)}\right)$$

což je maticově:

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

$$y = a\left(h^T W^{(y)} + b^{(y)}\right)$$ -> to jsou zároveň predikce

4.7) MLP output activations:

Binary:

$\sigma(x)$ – Bernoulliho distribuce : $\frac{1}{1+e^{-x}}$

K-class:

$softmax(x)$ – kategorická distr.: $softmax(x) \propto e^x$, $softmax(x)_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$

Hidden-layer:

$\sigma(x)$

$\tanh(x) = 2\sigma(2x) - 1$ – dělí 0 symetrické a rovinaci v nule

$ReLU(x) = \max(0, x)$ – nejčastější nelineární

5.1) Single-layer MLP, ReLU hidden, Softmax output

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

3) $\frac{\partial y^{(in)}}{\partial W^{(y)}}$ 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 stejně $\frac{\partial L}{\partial b^{(y)}}$

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

5) $\frac{\partial h^{(in)}}{\partial W^{(h)}}$ a $\frac{\partial h^{(in)}}{\partial b^{(h)}}$ → z čehož získáme: $\frac{\partial L}{\partial W^{(h)}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial y^{(in)}} \cdot \frac{\partial y^{(in)}}{\partial h} \cdot \frac{\partial h}{\partial h^{(in)}} \cdot \frac{\partial h^{(in)}}{\partial W^{(h)}}$

a stejně $\frac{\partial L}{\partial b^{(h)}}$

5.2) Universal approx. theorem

Mějme $\varphi(x): \mathbb{R} \to \mathbb{R}$ nekonstantní, omezenou a neklesající spojitou funkci.
Pro libovolné $\varepsilon > 0$ a libovolnou spojitou $f: [0,1]^D \to \mathbb{R}$, existují
$H \in \mathbb{N}$, $v \in \mathbb{R}^H$, $b \in \mathbb{R}^H$ a $W \in \mathbb{R}^{D \times H}$ takové, že pokud

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

kde φ je použita „elementwise",

pak $\forall x \in [0,1]^D: |F(x) - f(x)| < \varepsilon$

5.3) Minimum search in $f(x): \mathbb{R}^D \to \mathbb{R}$

Mějme $f(x): \mathbb{R}^D \to \mathbb{R}$ s minimem v x s podmínkami
$g_1(x) = 0, \dots, g_n(x) = 0$. $f, g_1, \dots, g_n$ nechť mají spoj. parc. der.
a nechť $\nabla_x g_1(x), \dots, \nabla_x g_n(x)$ jsou lin. nez.
Pak existují $\lambda_1, \dots, \lambda_m \in \mathbb{R}$ t.ž.

$$\mathcal{L}(x, \lambda) = f(x) - \sum_i^m \lambda_i g_i(x)$$ má nulový gradient v x i λ.

5.4) Categorical distr. and maximum entropy proof:

Mějme $p = (p_1 \dots p_n)$, u kterých chceme maximalizovat entropii.

Tedy chceme minimalizovat $-H(p)$ za podmínek:

$\forall i : p_i \geq 0$ → tu budeme ignorovat

$\sum_i p_i = 1$

$L = \left(\sum_i p_i \log p_i\right) - \lambda \cdot \left(\sum_i p_i - 1\right)$, z čehož lze odvodit:

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

tedy $p_i = e^{\lambda - 1}$ bude stejné pro všechna i a $\sum_i^n p_i = 1 \Rightarrow p_i = \frac{1}{n}$

5.5) Softmax derivation using max. entropy principle

Chceme minimalizovat $-\sum_i^N H(\pi(x_i))$ za podmínek:

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

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

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

Máme tedy:

$$L = \sum_i^N \sum_k^K \pi(x_i)_k \log(\pi(x_i)_k)$$
$$- \sum_j^D \sum_k^K \lambda_{jk} \left(\sum_i^N \pi(x_i)_k \, x_{ij} - [t_i = k] \, x_{ij} \right)$$
$$- \sum_i^N \beta_i \left(\sum_k^K \pi(x_i)_k - 1 \right)$$

5.6) Precision, Recall, F_1 a F_β score

Precision = $TP / (TP + FP)$, Recall = $TP / (TP + FN)$, Accuracy = $(T*) / (T* + F*)$

$F_1 = 2 \cdot \text{precision} \cdot \text{recall} / (\text{precision} + \text{recall})$, $F_\beta = (1 + \beta^2) \cdot \text{precision} \cdot \text{recall} / ((\beta^2 \cdot \text{precision}) + \text{recall})$

5.7) micro F_1 vs macro F_1

Micro F_1: Nejdříve sečtu všechny TP, FP, FN všech samostatných binárních klasifikací a vypočítám jedno F_1-score.

Macro F_1: Nejdříve vypočítám všechna jednotlivá F_1-score a pak vezmu jejich průměr.

5.8) Why accuracy is not always good

Accuracy není vhodná, pokud se ve vzorku přirozeně vyskytuje hodně správných bodů. Když mám velmi ojedinělou nemoc, test může jen fixně vždy říkat „negativní" a bude mít velmi vysokou přesnost.

6.1) TF, IDF

$$TF(t,d) = \frac{\#\text{výskytů } t \text{ v } d}{\#\text{termů v } d}, \quad IDF(t) = \log\left(\frac{\#\text{dokumentů } d}{\#\text{dokumentů s } t}\right)$$

$$TF\text{-}IDF = TF \cdot IDF \quad \rightarrow \text{jak moc je důležitý term pro dokument v korpusu}$$

6.2) Conditional entropy:

vars $x, y : x \sim X,\ y \sim Y$ → v diskrétním případě

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

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

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

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

$$\Rightarrow I(X,Y) \geq 0$$

$I(X,Y) = 0 \iff P(X,Y) = P(X)\cdot P(Y)$, tedy jsou nezávislé.

6.3) TF-IDF as mutual info

Mějme D dokumenty a T termy. Uděláváme vytahujeme dokument, děláme to uniformně:

$P(d) = 1/|D|$ a $I(d) = H(D) = \log |D|$

$P(d|t \in d) = 1 / |\{d \in D : t \in d\}|$

$I(d|t \in d) = H(D|t) = \log |\{d \in D : t \in d\}|$ $\quad 0 \cdot \log 0 = 0$

$I(d) - I(d|t \in d) = H(D) - H(D|t) = \log \frac{|D|}{|\{d \in D, t \in d\}|} = IDF(t)$

Tedy $I(D,T)$:

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

6.4) Word embedding:

Místo ohyzdných slov reprezentovaných sparse one-hot vectory, reprezentujeme je vektory reál. čísel, které ve vektorovém prostoru svou vzdáleností reprezentují podobnost/spřízněnost mezi slovy.

V MLP learningu dané vektory slouží jako vstupy pro trénování. Zároveň umožňuje pracovat s contextem slov, což dovoluje MLP lépe pochopit význam. Zároveň to dovoluje zachytit komplexní syntaktické a sémantické konstrukce.

6.5) SkipGram: Negative sampling

Rozdělíme output matici W na output embeddingy V.

Pro vstupní slovo w s emb. e_w a kontextovým slovem c s v_c provedeme logistickou regresi a zavedeme loss fci

$$-\log \sigma(e_w^T v_c)$$

Pak nasamplujeme k negativních samplů c_i, které nejsou v kontextovém okně a negativně je přidáme do loss fce:

$$-\sum_i^k \log \sigma(-e_w^T v_{c_i})$$

Distribuce, ze které samplujeme je heuristicky upravená kotev. distr. založená na frekv. slov.

6.6) Pre-trained word-embeddings

Použil bych sliding okno embeddingu a klasifikoval bych podle prostředního.

7.1) k-nearest neighbours

Regression: k nejbližších sousedů má hodnotu t_i a váhy w_i

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

Classification: nejčastější třída je moje predikce

$t_i \in \mathbb{R}^K$: $t = \sum_i \frac{w_i}{\sum_j w_j} t_i$, predicted je: $\operatorname{argmax}_k \sum_i w_i t_{i,k}$

L_p norm: $x, y \in \mathbb{R}^D$, vzdálenost je $\|x-y\|_p$, kde

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

Váhy: - uniformní

- inverzní -> proporcionální k $1/$vzdálenost
- softmax -> proporcionální k softmax(-distance)

7.2) L^2 jako Bayesian interference:

$$w_{MAP} = \operatorname*{argmax}_w p(X|w) \cdot p(w)$$

$$= \operatorname*{argmax}_w \prod_{i=1}^{N} p(x_i|w)\, p(w)$$

$$= \operatorname*{argmin}_w \sum_i^N -\log p(x_i|w) - \log p(w))$$

po substituci Gausian prior

$$w_{MAP} = \operatorname*{argmin}_w \sum_i^N \left(-\log p(x_i|w) + \frac{D}{2}\log(2\pi\sigma^2) + \frac{\|w\|^2}{2\sigma^2}\right)$$

to je L^2 regularizace

7.3) $p(C_k|x)$ under Naive bayes

Modelování distribuce $p(x|C_k)$ je složité, protože x může být vysoko-dimenzionální.

Předpoklad: Všechny x_d jsou nezávislé se zadaným C_k.

Pak lze psát: $p(x|C_k) = p(x_1|C_k) \cdot p(x_2|C_k, x_1) \ldots$ Predikujeme: $\operatorname{argmax}_k p(x|C_k) \cdot p(C_k)$

tzn: $p(x|C_k) = \prod_{d=1}^{D} p(x_d|C_k)$.

8.1) Independent discrete vars are uncorrelated

Nekorelované := $Cov(X,Y) = 0$

$$Cov(X,Y) = \mathbb{E}\left[(x - \mathbb{E}(x)) \cdot (y - \mathbb{E}(y))\right]$$

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

protože nezávislé $= \sum_x P(x) \cdot (x - \mathbb{E}(x)) \cdot \sum_y P(y)(y - \mathbb{E}(y))$

$$= \mathbb{E}_x[x - \mathbb{E}[x]]\, \mathbb{E}_y[y - \mathbb{E}[y]] = \underline{\underline{0}}$$

8.2) Kovariance a pearsonův koef.

Kovariance: $Cov(X,Y) = \mathbb{E}\left[(x - \mathbb{E}(x)) \cdot (y - \mathbb{E}(y))\right]$

Pearson:

$$\rho = \frac{Cov(X,Y)}{\sqrt{Var(X)} \cdot \sqrt{Var(Y)}} \in \langle -1, 1 \rangle$$

8.3) Spearman korelace a Kendall rank

Spearman ρ je počítán je měřen na rankích, kde rank elementu je jeho index ve vzestupně setříděné posloupnosti.

Kendall τ měří objem konkordantních párů (y ↑/↓ když to stejné dělá x)

$$= \frac{\sum_{i<j} sign(x_j - x_i) \cdot sign(x_j - x_i)}{\binom{n}{2}} \in \langle -1, 1 \rangle$$

8.4) Correlation as evaluation metric

Vhodné využít na:

- vyhodnocovací metriku tasku
- měření kvality autoencoderů
- měření kvality automatických metrik v porovnání s lidskou metrikou.

8.5) Assess validity of evaluation metrix

Mám-li lidsky hodnocené data, chci měřit metriku, která bude nejvíce „korelovat" s lidským hodnocením. Tam korelaci rozhodně použiju.

8.6)

Cohen's k

Cohen's k vyjadřuje Inter-annotater agreement (IAA)

$$k = \frac{p_0 - p_E}{1 - p_E}$$

, kde p_0 je observed agreement,
p_E je expected agreement

V machine-learningu se používá pro filtraci nalezených dat a chybujících anotátorů. Důvěřuj ne všichni outliers jsou šum...

8.7) Ensembling MSE:

Mějme $\varepsilon_i(x)$:= chyba i-tého modelu od skutečnosti:

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

Jelikož mají ε_i průměr 0 a jsou nekorelované, musí platit:

$$\mathbb{E}[\varepsilon_i(x)\,\varepsilon_j(y)] = 0 \;:\; i \neq j$$

a tedy

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

takže chyba ensemblu je $\frac{1}{M}$ hlavní mínus.

8.8) Knowledge-distillation

Velký ensemble může být moc velký nebo pomalý. Proto pak natrénuji model, co mimikuje ten velký.

1) Natrénuju velký $p_{teacher}(y|x,w)$

2) Natrénuju malý model s $H(p_{student}(y|x,w), p_{teacher}(y|x,w))$ jako trénovacím cílem.

9.1) Decision-tree:

Ve vnitřních uzlech jsou držený podmínky, které určují další dělení.
V listech jsou reprezentanti výsledných tříd. Z = jako průměr podle, co tam spadly.

Criterion popisuje, jak moc shodní jsou data ve vrcholu.

$$c_{SE} := \sum_{i \in I_T} (t_i - \hat{t}_T)^2, \text{ kde } \hat{t}_T = \frac{1}{|I_T|} \sum_{i \in I_T} t_i.$$

Splitují tak, abych každým uzlem reprezentoval nejjímavější solution markery, které na vstupu mám. Snažím se minimalizovat $c_R + c_L - c_T$

9.2)
9.3) Multiclass:

Ve vnitřních vrcholech držím podmínku, podle které splituju.
V listech držím listy, co mi tam spadly.
Predikuju v listu takovou třídu, která je v něm nejčastější:

$p_T(k) :=$ prob. třídy k v regionu T.

Opět splitují podle $c_R + c_L - c_T$.

Gini index: jak často by byl náhodně zvolený element náhodně ohodnocený podle $p_T(k)$ špatně ohodnocen.

$$c_{gini}(T) = |I_T| \cdot \sum_k p_T(k)(1 - p_T(k))$$

Entropy criterion

$$c_{entropy}(T) = |I_T| \cdot H(p_T) = -|I_T| \cdot \sum_{\substack{k \\ p_T(k) \neq 0}} p_T(k) \cdot \log p_T(k)$$

9.4) Binary Gini

$n_T(0)$ = # elementů se značkou 0, $n_T(1) = \dots$

$$p_T = \frac{1}{|I_T|} \sum_{i \in I_T} t_i = \frac{n_T(1)}{n_T(0) + n_T(1)}$$

Sum of squares: $L(p) = \sum_{i \in I_T} (p - t_i)^2$

$\Leftrightarrow$ p, které minimalizuje loss, je $p = p_T$ $\qquad p_T^2 - 2p_T + 1$

$$L(p_T) = \sum_{i \in I_T} (p_T - t_i)^2 = n_T(0) \cdot (p_T - 0)^2 + n_T(1) \cdot (p_T - 1)^2$$

$$= \frac{n_T(0)\, n_T(1)^2}{(n_T(0) + n_T(1))^2} + \frac{n_T(1) \cdot n_T(0)^2}{(n_T(0) + n_T(1))^2} = \frac{(n_T(1) + n_T(0))\, n_T(0)\, n_T(1)}{(n_T(0) + n_T(1)) \cdot (n_T(0) + n_T(1))} =$$

$$= (n_T(0) + n_T(1))(1 - p_T) \cdot p_T = |I_T| \cdot p_T \cdot (1 - p_T)$$

9.5) Entropy criterion derivation:

$n_T(k)$:= # elementů s targetem k v T.

$$p_T(k) = \frac{1}{|I_T|} \sum_{i \in I_T} [t_i = k] = \frac{n_T(k)}{|I_T|}$$

Uvažme distr. p na U a NLL loss $L(p) = \sum_{i \in I_T} -\log p_{t_i}$

Minimalizující $p = p_T(k)$.

$$L(p_T) = \sum_{i \in I_T} -\log p_{t_i} = -\sum_{\substack{k \\ p_T(k) \neq 0}} n_T(k) \log p_T(k)$$

$$= -|I_T| \sum_{\substack{k \\ p_T(k) \neq 0}} p_T(k) \log p_T(k) = |I_T| \cdot H(p_T)$$

9.6)

Stromy se trénují oddělené, jejich trénovací data nejsou stejná.

a) Bagging

- vytvořím si ze vstupní množiny více množin tím, že do nich data uniformě náhodně vybírám s opakováním.

b) Random subset

- v každém splitu náhodně zahodím některá data, když vyhodnocuji split.

Volím path softem pro regresi, hardem pro klasifikaci.

10.1)

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

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

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

Criterion: $-\frac{1}{2}\sum_T \frac{\left(\sum_{i\in I_T} g_i\right)^2}{\lambda + \sum_{i\in I_T} h_i} + \text{const}$

10.2)

Pro každý timestamp trénuji u stromů $w_{t,k}$, kde každý predikuje single lin. část.

Predikce je pak prováděna jako:

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

a loss je tedy per-example:

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

10.3)

Vhodné pro low-dimensional data, kde každá feature nese samou o sobě významu.

- protože dává smysl rozdělovat data podle jejich features, pokud feature nese význam

11.1)

Libovolnou matici X velikosti $m \times n$, hodnosti r, lze faktorizovat jako:

$$X = U \Sigma V^T, \text{ kde}$$

U je ortonormální $m \times m$

Σ je diagonální $m \times n$ s nezápornými hodnotami, v sestupném pořadí

V je ortonormální $n \times n$

U jsou komponenty řádků, V jsou komponenty sloupců.

$X = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_r u_r v_r^T$ (X má hodnost r)

Redukovaná verze SVD: vyhodíme všechna $\sigma_k : k > r$ a použijeme menší U, V.

11.2)

$X \in \mathbb{R}^{m \times n}$, $X_k = \sigma_1 u_1 v_1^T + \dots + \sigma_k u_k v_k^T$ je aproximace pomocí SVD.

$\forall B \in \mathbb{R}^{m \times n}$ s $\text{rank} = k$:

$$\|X - X_k\|_F \leq \|X - B\|_F$$

11.3)

Nechť $S = \frac{1}{N}(X - \bar{x})^T (X - \bar{x})$, pak PCA X-ka jsou vlastní vektory S,

mean-centered

tedy V matice SVD dekompozice $X - \bar{x}$,

pouze přeškálované $1/N$

11.4)

- spočítat mean μ řádků X
- spočítat kovariační matici: $S = \frac{1}{N}(X-\mu)^T(X-\mu)$
- for i in $\{1 \dots M\}$
  - v_i = náhodné
  - repeat until convergence
    - $v_i = S v_i$
    - $\lambda_i = \|v_i\|$
    - $v_i = v_i / \lambda_i$
  - $S = S - \lambda_i v_i v_i^T$
- return XV, kde sloupce V jsou $v_1, v_2, \dots, v_m$

11.5)

- Initialize $\mu_1, \dots, \mu_k$ randomly
- repeat until convergence:
  - compute best possible $z_{i,k} = \begin{cases} 1 & \text{if } k = \arg\min_j \|x_i - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$ → pokud k-tý bod je nejbližší
  - compute best possible $\mu_k = \arg\min_\mu \sum_i z_{i,k} \|x_i - \mu\|^2$
    - což jako derivací podle μ dává: $\mu_k = \frac{\sum_i z_{i,k} x_i}{\sum_i z_{i,k}}$ → per-feature avg.

- k-means++ → $\mu_1 - \mu_k$ jsou inicializovány tak, že μ_1 náhodně, ostatní proporcionálně k 2. mocnině vzdálenosti od středu nejbližšího clusteru

12.1)

	H_0	H_1	
H_0	true negative	false negative	type II
$\neg H_0$	false positive	true positive	
	type I		

significance level = type 1 error rate

12.2)

Test-statistics: je to jako „shrnutí" viděných dat, dovolí rozlišovat nulovou a alt. hypotézu.

p-values: past. získání test-stat. hodnoty alespoň tak extrémní jako té aktuálně viděné, předpokládající platící nulovou hypotézu.

– low p-value := velmi malá past. při nulové hypotéze

12.3)

1) Formulace H_0, alt. i H_1
2) zvolení test. statistiky
3) Spočtání hodnoty test-statistiky
4) Spočítání p-value
5) rejecting H_0 pokud p-value je méně než zvolená α (ať most 5%)

12.4)

One-sample samplujeme z jedné distribuce,
Two-sample samplujeme ze dvou distribucí,
Paired-test samplujeme ze dvou distribucí, ale jejich hodnoty párujeme

12.5)

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

Bonferroni correction: rejects H_0 pokud v rodině velikosti m $p_i \leq \frac{\alpha}{m}$

jelikož $P(\bigcup_i A_i) \leq \sum_i P(A_i)$

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

12.6)

perf = [] —> získáme jako výsledek 2,5-97,5 percentilu

repeat R times:

- sample N test sets with replacement, together with predictions
- measure performance using metric E and append to perf

12.7)

diffs = []

repeat R times:

- sample N test sets with replacement, together with predictions
- measure performance of the models y and z on the sampled data using metrics E, append their diff to diffs

return the ratio of the diffs that are ≤ 0

12.8) Myšlenka je:

Jsou-li modely stejně dobré, je jedno jestli rozdíl predikcí z prvního nebo druhého. Pak uvidím-li všechny možné výběry z dvojí predikcí, mám dist. perform. za předpokladu, že jsou modely stejně dobré.

Potom, p-value je pravděpodobnost performance dotazovaného modelu.