Úvod

- Definice (Mitchell 1997): Program se zlepšuje s časem, který řeší danou úlohu.
- Úloha (Task T)
  - Klasifikace - rozdělení vstupů do tříd
    - Každému vstupu přiřadíme celé číslo -> přímo ho klasifikujeme
    - Nebo také můžeme vrátit celé rozložení pravděpodobnosti
  - Regrese - výstupem je číslo
    - Každému vstupu přiřadíme reálné číslo
- Míra (Mesuare M)
  - U klasifikace jednoduše úspešnost
- Zkušenost (Experiance E)
  - S učitelem (supervised) - porovnávám výsledek programu se správnými výsledky
  - Bez učitele (unsupervised)
  - Zpětnovazební učení - dostanu zpětnou vazbu na výsledek
- Data generating distribution
  - Zdroj dat, na kterých se chceme učit, se správným ohodnocením
- Optimalizace -> snažím se na známých datech o co nejmenší chybu
- Strojové učení -> snažím se o co nejmenší chybu na neznámých datech

Značení

- *a*, ***a***, ***A***, A ~ skalár, vektor, matice, tenzor
- a, **a**, **A** ~ náhodný skalár, vektor, matice
- Derivace, parciální derivace, gradient
- Trénovací množina X^{N*D}, kde N je počet vzorků a D je počet featur
- Target množina t^N, kde t_i je buď reálné číslo (regrese) nebo celé číslo z intervalu (u klasifikace)

Lineární regrese

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
- Nejjednodušší lineární model
- $y(x, w, b) = x^T w + b$
- $Y = Xw$ kde $w = (w_1, w_2, \dots, b)$, tedy bias dám nakonec vektoru vah, abych s ním nemusel pracovat zvlášť
- Chci postupně upravovat váhy tak, abych minimalizoval chybu
- Použiji funkci MSE pro vyjádření chyby $MSE(w) = \frac{1}{2} \sum_i^N (x_i^T w - t_i)^2 = \frac{1}{2} ||Xw - t||^2$
  - Správně je to 1/N, ale protože monotonie, tak tam dám 1/2, aby se mi to zkrátilo, až to budu derivovat
- Existuje explicitní řešení pro minimalizace MSE (což je dost neobvyklé)
  - Pro nalezení minimální chyby chceme, aby $\nabla MSE(w) = 0$, tedy minimalizuji podle w
  - $\frac{\partial}{\partial w_j} \frac{1}{2} \sum_i^N (x_i^T w - t_i)^2 = \frac{1}{2} \sum_i^N 2(x_i^T w - t_i) x_{ij} = \sum_i^N (x_i^T w - t_i) x_{ij} = 0$

    - $X^T(Xw - t) = 0$
    - Řešení metodou nejmenších čtverců $w = (X^TX)^{-1}X^Tt$
    - Předpokládá, že matice X^TX je regulární. Pokud by byla singulární, je nutno použít SVD
- **Overfitting** - malá chyba na testovacích datech, ale velká chyba na trénovacích datech
- **Underfitting** - velká chyba všude
- Pokud mám jen jednu vstupní featuru, můžu na ní udělat nějaké nelineární operace (mocnění) a tím vytvořit více featur (tzv. **polynomial features**)
- Naopak pokud mám vstupní featuru, která mi reprezentuje den v týdnu, je vhodné ji převést do **one-hot**

Algoritmus

- Vstup: X^{N*D}, t^N
- Výstup: w^{D+1}, +1 je za bias
- Kroky:
    - $w = (X^TX)^{-1}X^Tt$

Regularizace

- Obecně jde o jakoukoliv úpravu modelu (nejen lineární regrese), jejímž cílem je snížení chyby
- Například omezení kapacity modelu (zabraňuje overfittingu)

L2 regularizace

- Snaží se, aby w byly co nejmenší
- Neaplikuje se na bias, musím ho tedy uvažovat zvlášť
    - Pak je invariantní vůči tomu, když např. ke všem vstupům přičtu konstantu
- $MSE(w) = \frac{1}{2}\sum_i^N(x_i^Tw - t_i)^2 + \frac{\lambda}{2} \mid\mid w \mid\mid^2 = \frac{1}{2}||Xw - t||^2 + \frac{\lambda}{2} \mid\mid w \mid\mid^2$
- Explicitní se pak liší jen přičtením jednotkové matice
    - $w = (X^TX + \lambda I)^{-1}X^Tt$
    - Mimochodem, matice $(X^TX + \lambda I)$ pro $\lambda > 0$ je vždy regulární

Hyperparametry

- Jde o hodnoty, které nejsou součástí vstupních dat, ale upravují chování modelu
- Právě například regularizační parametr λ, stupeň polynomial features atd...

SGD

Náhodné veličny

- **Střední hodnota** (expectantion) funkce $f(x)$ je v podstatě vážený průměr
    - $\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x)$, kde $P(x)$ je pravděpodobnostní rozložení
- **Rozptyl** (variance) udává, jak se průměrně hodnoty liší od průměru
    - $Var(f(x)) = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2]$
- **Bias** (jiný bias, než ten, který se přičítá v LR)
    - Máme **estimator** (odhadce), pak $bias = \mathbb{E}[estimate] - true\ estimated\ value$
    - Pokud je $bias = 0$ je estimator **unbiased**, jinak **biased**

Gradient descent

- Obecná metoda pro hledání "minim" funkcí
- $w \leftarrow w - \alpha \nabla_w E(w)$, kde α je tzv. **learing rate** (další hyperparametr) a $E(w)$ je obecně chybová funkce
- Definuji $\nabla_w E(w) = \nabla_w \mathbb{E}_{(x,t)}[L(y(x,t),t)]$
- **Standart gradient descent**
  - K výpočtu využiju všechna vstupní data, pak dostanu $\nabla_w E(w)$ přesně
- Stochastic gradient descent
  - K výpočtu gradientu použiju náhodné vstupní dato, tedy
  - $\nabla_w \mathbb{E}(w) \approx \nabla_w L(y(x,w),t)$ pro náhodně vybrané (x,t)
- **Minibatch SGD**
  - K výpočtu gradientu použiju několik (batch) náhodně vybraných dat
  - $\nabla_w \mathbb{E}(w) \approx \frac{1}{B} \sum_i^B \nabla_w L(y(x_i,w),t_i)$
- Řešení bude **konvergovat** k optimu za těchto podmínek
  - Chybová funkce $L(y(x,w),t)$ je **konvexní** a **spojitá**
  - Posloupnost learning rates $(\alpha_i)_i$ splňuje:
    - $\forall i : \alpha_i > 0$, $\sum_i \alpha_i = \infty$, $\sum_i \alpha_i^2 < \infty$, tedy $\alpha_i \to 0$
  - Pokud by byla L nekonvexní, konverguje SGD pouze do lokálního minima

SGD pro lineární regresi

- https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDRegressor.html
- Pro standartní GD: $E(w) = \mathbb{E}_{(x,t)}[\frac{1}{2}(x^T w - t)^2] + \frac{\lambda}{2} \| w \|^2$
- Pro Minibatch SGD: $E(w) = \frac{1}{b} \sum_i^b \frac{1}{2}(x_i^T w - t_i)^2 + \frac{\lambda}{2} \| w \|^2$
  - $\nabla_w E(w) = \frac{1}{b} \sum_i^b ((x_i^T w - t_i) x_i) + \lambda w$

Algoritmus

- Vstup: $X^{N*D}, t^N, \alpha \in \mathbb{R}^+, \lambda \in \mathbb{R}$
- Výstup: w^D, které minimalizují MSE (doufejme)
- Kroky:
  - $w \leftarrow 0$
  - Dokud jsem nezkonvergoval
    - Vyberu náhodně minibatch z dat velikosti b
    - $w \leftarrow w - \alpha \nabla_w E(w)$, kde $\nabla_w E(w) = \frac{1}{b} \sum_i^b ((x_i^T w - t_i) x_i) + \lambda w$

Cross-validace

- https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
- Pro vhodné nastavení hyperparametrů, preprocessing nebo výběr modelu potřebuji průběžně ověřovat, jak je můj model dobrý na datech, která nezná
- Také chci trénovat vždy na trochu jiných datech, aby byl model "objektivnější"
- K tomu použiju tzv. **k-fold cross validation**, kdy si trénovací množinu rozsekám na k částí
- Na k-1 částech natrénuji model s nějakým nastavením a na té poslední udělám validaci (tj. predikci)

Klasifikace

Perceptron

- Klasifikace do dvou tříd
- **Accuracy** (přesnost) budeme měřit jako poměr správně/špatně klasifikovaných vzorků
- Pro převedení lineární regrese na binární klasifikaci se určí **treshold**, který odděluje dvě třídy (obvykle 0)
- Předpokládáme, že data jsou lineárně separabilní (pokud ne, nebude existovat vhodný vektor w)
- Vektor w je kolmý na dělící nadrovinu (protože pro dva body na ní platí $(x_1 - x_2)^T w = 0$)
- **Vzdálenost vzorku x** od dělící nadroviny
  - Buď $x_\perp$ projekcí na nadrovinu
  - Pak $x = x_\perp + \frac{rw}{||w||}$
  - Tedy $r = \frac{x^T w}{||w||} = \frac{y(x)}{||w||}$
- **Nevýhoda**: najde nějakou dělící přímku, tedy muže být "blízko" jedné množiny a "daleko" od druhé

Algoritmus

- Vstup: $X^{N*D}, t \in -1; 1^N$, kde X jsou lineárně separabilní
- Výstup: w^D, že $t_i x_i w > 0$ (resp. $sign(x_i^T w) = t_i$)
- Kroky:
  - $w \leftarrow 0$
  - Dokud nejsou všechny vzorky správně klasifikované
    - Pokud $t_i x_i^T w \leq 0$
      - $w \leftarrow w + t_i x_i$
- Vlastně jde o **online GD** algoritmus, kde chybová funkce je $L(y(x, w), t) = ReLU(-tx^T w) = max(0, -tx^T w)$
  - Není nutno používat learing rate, neboť násobení vah konstantou nezmění predikce
- Algoritmus **konverguje**, tedy pro lineárně separabilní data vždy najde dělící přímku

Logistická regrese

Distribuce

Bernouliho distribuce

- $x \in \{0, 1\}, \phi \in (0, 1)$
- $P(x) = \phi^x (1 - \phi)^x$ ... tj. buď ϕ nebo $(1 - \phi)$

Kategorická distribuce

- $x \in \{0, 1\}^k, p \in (0, 1)^k, \sum_i^k p_i = 1$
- $P(x) = \prod_i^k p_i^{x_i}$

Normální distribuce

- Suma náhodných pokusů s konečným rozptylem k ním konverguje
- Jde o distribuci s maximální entropií

Entropie

- **Surprise**, **Self-information** (míra překvapení)
  - Pro jisté jevy (Pr=1) je nulová
  - Roste pro méně pravděpodobné jevy
  - Pro nezávislé jevy musí být aditivní (tj. když mě překvapí dvě věci, tak se překvapení sečtou)
  - $I(x) = -\log P(x) = \log \frac{1}{P(x)}$
- **Entropie** je pak střední hodnota překvapení v celém rozložení (distribuci)
  - $H(P) = \mathbb{E}_{x \sim P}[I(x)] = -\mathbb{E}_{X \sim P}[\log P(x)]$
  - $H(P) = -\sum_x P(x) \log P(x)$
  - **Motivace**
    - Chci posílat zprávy kanálem, kudy chodí 0,1
    - Mám pravděpodobnosti jednotlivých znaků a místo, které zabírají
    - Pak průměrný počet bitů na znak je $\sum_x P(x)S(x)$, kde P je pravděpodobnost a S je velikost
    - Tedy entropie mi říká, kolik bitů potřebuji, abych mohl reprezentovat výsledek náhodného pokusu (nejen to, platí pro $\log_2$)
      - Je to střední hodnota "self-information"
- **Corss-Entropie**
  - Mám dvě distribuce, P, Q
    - K motivační příkladu: znaky mi chodí z P, ale kód jsem postavil na znacích z Q
  - $H(P, Q) = -\mathbb{E}_{X \sim P}[\log Q(x)]$
- **Gibbsova nerovnost**
  - $H(P, Q) \geq H(P)$
  - $H(P, Q) = H(P) \iff P = Q$
  - ... vypadá to skoro jako metrika, ale není to metrika, protože obecně neplatí $H(P, Q) = H(Q, P)$
- Pro kategorickou distribuci o n prvcích platí $H(P) = \log n$ (u motivačního příkladu bych každému prvku přiřadil sekvenci o $\log n$ bitech)
- **K čemu to a co chceme?**
  - Máme definovanou cross-entropii $H(P, Q)$
  - Klasifikační model vrátí distribuci ... a já chci, aby se co nejvíc blížila distribuci v datech
  - Takže chci spočítat cross-entropy (nejlépe spojitá funkce, kterou pak zderivuji)
- Obecně je rozložení s **největší entropií** to nejobecnější → a to je právě normální rozložení

KL divergence

- $D_{KL}(P||Q) = H(P, Q) - H(P) = \mathbb{E}_{x \sim P}[\log P(x) - \log Q(x)]$
- Budeme na ní měřit, jak je distribuce Q daleko od P ... tedy jak moc jsem k ním blízko
- Nenulová
- Nesymetrická, protože $H(P, Q) \neq H(Q, P)$

- Záleží, přes co kterou distribuci se počítá střední hodnota
- V praxi pak P budou data a Q predikce
  - → chceme, aby se predikce s daty shodovala tam, kde data máme … nezajímá nás, co predikce predikuje tam, kde data nemáme

MLE

- Maximum likelihood estimation
- **Empirické rozložení dat** z množiny $X = \{x_1, \dots x_N\}$ je $\hat{p}_{data}(x) = \frac{|\{i: x_i = x\}|}{N}$
- **Likelihood** je pak $L(w) = p_{model}(X, w) = \prod_i^N p_{model}(x_i, w)$, kde $p_{model}(x, w)$ je množina různých rozložení
  - Zafixoval jsem x - to je trénovací množina
  - Nabývá hodnoty z <0,1> (to je likelihood)
- **Maximum likelihood estimation** pro w je pak
  - $w_{MLE} = \arg\max_w p_{model}(X, w) = \max_w \prod_i^N p_{model}(x_i, w)$ (je to součin pravděpodobností, protože data jsou nezávislá)
  - $w_{MLE} = \arg\min_w \sum_i^N - \log p_{model}(x_i, w)$ → **Negative Log Likelihood**
  - $w_{MLE} = \arg\min_w \mathbb{E}_{x \in p_{data}}[- \log p_{model}(x_i, w)]$ (přidám si $1/N$, minimum to neovlivní) → to je **cross-entropy**
  - $w_{MLE} = \arg\min_w H(p_{data}(x), p_{model}(x, w))$ ($\pm H(p_{data}(x))$)
  - $w_{MLE} = \arg\min_w D_{KL}(p_{data}(x) || p_{model}(x, w))$ (navíc ještě $+H(p_{data}(x))$, ale to je konstanta, takže nemá na minimum vliv)
- MLE pak lze zobecnit pro případ, kdy sis pro dané x předpovídat t
  - $w_{MLE} = \arg\max_w p_{model}(t|X, w) = \max_w \prod_i^N p_{model}(t_i|x_i, w)$
  - $w_{MLE} = \arg\min_w \sum_i^N - \log p_{model}(t_i|x_i, w)$ … NLL (negative log likelihood)
  - $w_{MLE} = \arg\min_w H(\hat{p}_{data}, p_{model}(t|x, w))$
- MLE je **konzistentní**, tj. s rostoucím počtem dat konverguje ke skutečnému rozložení dat
- MLE má také **nejmenší MSE**

Binární klasifikace

- Funkce **sigmoid** $\sigma(x) = \frac{1}{1+e^{-x}} : \mathbb{R} \to [0, 1]$
  - Chápeme ji jako pravděpodobnost pozitivního výsledku
  - **Derivace sigmoid** $\sigma'(x) = \sigma(x)(1 - \sigma(x))$
- $P(C_1|x) = \sigma(x^T w + b)$
- $P(C_0|x) = 1 - P(C_1|x)$
- Pokud má platit $y(x, w) = P(C_1|x) = \frac{1}{1+e^{\hat{y}(x,w)}}$, tak musí $\hat{y}(x, w) = \log\left(\frac{P(C_1|x)}{1-P(C_1|x)}\right) = \log\left(\frac{P(C_1|x)}{P(C_0|x)}\right)$
  - To se jmenuje **logit** - ta část, která je parametr nelineární funkce (tj. lineární část modelu)
    - Co je logit? Logaritmus poměru pravděpodobnost. také inverz sigmoidu
- Použiji chybovou funkci **NLL** $E(w) = \frac{1}{N} \sum_i - \log\left(p_{model}(C_{t_i}|x_i, w)\right)$
- Algoritmus opět používá SGD

Algoritmus

- Vstup: $X^{N\times D}, t \in \{0,1\}^N$, learing rate $\alpha \in \mathbb{R}^+$
- Kroky
  - $w \leftarrow 0$
  - Dokud jsem nezkonvergoval
    - $g \leftarrow \frac{1}{b}\sum_i^b \nabla_w - \log\left(P(C_{t_i}|x_i, w)\right) = \frac{1}{b}\sum_i^b((x_i^T w - t_i)x_i)$$
    - $w \leftarrow w - \alpha g$

Odvození MSE z MLE

- Pokud náš model generuje normální rozložení, pak lze z ML odvodit MSE

Klasifikace do více tříd

- Klasifikátor se jmenuje multinomial logistic regression nebo maximum entropy classifier nebo softmax regression
- Místo do dvou chceme klasifikovat do *K* tříd
- Budeme generovat *K* výstupních parametrů, pro každý bude existovat sada vah → $W \in \mathbb{R}^{D\times K}$
  - $\hat{y}(x, W) = x^T W$, tedy $\hat{y}(x, W)_i = x^T W_{*i}$
- Zobecním funkci sigmoid na **softmax**
  - $softmax(z)_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$
  - Pak
    $P(C_i|x, W) = y(x, W)_i = softmax(\hat{y}(x, W))_i = softmax(x^T W) = \frac{e^{(x^T W)_i}}{\sum_j e^{(x^T W)_j}}$
  - Kde $\hat{y}(x, W)_i = \log\left(P(C_i|x, W)\right) + c$
    - Na konstantě *c* nezáleží, protože ze softmaxu můžu vytknout $\frac{e^c}{e^c} = 1$, tedy $softmax(z + c)_i = softmax(z)_i$

Algoritmus

- Vstup: $X^{N\times D}, t \in \{0,1,\dots K-1\}^N$, learing rate $\alpha \in \mathbb{R}^+$
  - w reprezentuje jak matici vah W, tak eventuálně bias b
- Kroky
  - $w \leftarrow 0$
  - Dokud jsem nezkonvergoval
    - $g \leftarrow \frac{1}{b}\sum_i^b \nabla_w - \log\left(P(C_{t_i}|x_i, w)\right) = \frac{1}{b}\sum_i^b((x_i^T w - 1_t)x_i^T)$$
    - $w \leftarrow w - \alpha g$

Regiony

- Všechny regiony tříd při klasifikaci do k-tříd jsou **konvexní**

F1-score

- **accuracy** $= \frac{TP+TN}{TP+TN+FP+FN}$
- **precision** $= \frac{TP}{TP+FP}$

- **recall** $= \frac{TP}{TP+FN}$
- **F1** $= \frac{2}{precision^{-1}+recall^{-1}} = \frac{TP+TP}{TP+FP+TP+FN}$
- Také je nutné použít vhodný průměr
  - Aritmetický $AM(p,r) = \frac{p+r}{2}$
    - $AM(100,1) = 50.5$
  - Geometrický $GM(p,r) = \sqrt{p*r}$
    - $AM(100,1) \approx 10$
  - Harmonický $HM(p,r) = \frac{2}{\frac{1}{p}+\frac{1}{r}}$
    - $AM(100,1) \approx 2$
- F1 skóre může být zobecněno na F_β, kde recall je β-krát důležitější než precision
  - $F_\beta = \frac{1+\beta^2}{precision^{-1}+\beta^2*recall^{-1}}$

MLP

- Obecně reprezentuje lineární modely
- Multilayer perceptron (vícevrstvý perceptron)

Bez skryté vrstvy

- Mám *D* vstupních vrcholů pro featury
- Mám *K* výstupních vrcholů (v případě regrese $K = 1$, jinak počet tříd-1)
- Každý vstupní vrchol je propojený se všemi výstupními a hrany jsou ohodnoceny váhami
- Hodnota výstupního vrcholu $y_i = a(\sum_j x_j w_{ij} + b_i)$, kde a je aktivační funkce

Se skrytou vrstvou

- Navíc uvažuji skrytou vrstvu *h* s aktivační funkcí *f*
- $h = f(x^T W^{(h)} + b^{(h)})$
- $y = a(h^T W^{(y)} + b^{(y)})$

Aktivační funkce výstupní vrstvy

- Lineární regrese - identita nebo $exp(x)$ pro Posionovksou regresi
- Binární klasifikace - sigmoid $\sigma(x)$
- Klasifikace do více tříd - $softmax(x)$

Aktivační funkce skryté vrstvy

- Žádná - pak skrytá vrstva nepomůže, model zůstane pořád stejný (zachová se linearita)
- $tanh(x)$ - symetrický sigmoid
- $ReLU(x) = max(0,x)$ - používá se dnes

Algoritmus trénovaní MLP pomocí SGD

- Vstup: $X^{N \times D}, t^N$, learing rate $\alpha \in \mathbb{R}^+$
- w reprezentuje jak váhy W, tak bias b
- Kroky:
  - Inicializace w

        - Váhy nastavím náhodně z $\left\langle -1/\sqrt{M}; 1/\sqrt{M} \right\rangle$, kde M je velikost vrstvy
        - Biasy nastavím na 0
    - Dokud jsem nezkonvergoval
        - $g \leftarrow \frac{1}{b} \sum_i^b \nabla_w - \log\left(P(t_i|x_i, w)\right)$$
        - $w \leftarrow w - \alpha g$

Universal approximation theorem

Buď $\phi(x) : \mathbb{R} \to \mathbb{R}$ rostoucí, spojitá a omezená. Pak $\forall \epsilon > 0$ a spojitou funkci $f : \langle 0, 1 \rangle^D \to \mathbb{R}$ existuje $H \in \mathbb{N}, v \in \mathbb{R}^H, b \in \mathbb{R}^H, W \in \mathbb{R}^{D \times H}$, že pro funkci $F(x) = v^T \phi(x^T W + b)$ platí $\forall x \in \langle 0, 1 \rangle^D : |F(x) - f(x)| < \epsilon$.

K-nearest neigbors

- Výsledek určí podle "nejbližších" známých prvků v trénovací množině
- Jednoduchý, ale často efektivní algoritmus jako pro klasifikaci, tak pro regresi
- V algoritmu v podstatě není trénovací fáze, vše probíhá až při predikci
- Hyperparametry
    - **k** - hledaný počet sousedů
    - **metrika** - jak najít nejbližší sousedy (obvykle L_p norma, kde $d(x, y) = \|x - y\|_p = \sqrt[p]{\sum_i |x_i|^p}$)
    - **váhy**
        - Uniformní - všech k sousedů má stejnou váhu
        - Inversní - $\frac{1}{d(x,y)}$
        - Softmax
- Výsledek $t = \sum_i^k \frac{w_i}{\sum_j^k w_j} t_i$

Kernel metody

Kernel lineární regrese (SGD)

- Cíl: zrychlení SGD (i lineární regrese) pro data, kde potřebujeme polynomial features
- Vyjádřím w jako lineární kombinaci vstupních vektorů s polynomial features
    - $w = \sum_i \beta_i \phi(x_i)$, kde $\phi : \mathbb{R}^D \to \mathbb{R}^{D^p}$
- Upravím SGD update
    - $w \leftarrow w - \frac{\alpha}{b} \sum_i^b (\phi(x_i)^T w - t_i)\phi(x_i) \leftarrow \sum_i^b (\beta_i - [i \in b] \frac{\alpha}{b} (\phi(x_i)^T w - t_i))\phi(x_i)$
    - Tedy pro $i \in \{1, \dots, b\}$ se upraví $\beta_i = \beta_i - \frac{\alpha}{b}(\sum_j^n (\beta_j \phi(x_i)^T \phi(x_j)) - t_i)$

Algoritmus

- Vstup: $X^{N \times D}, t^N$, learing rate $\alpha \in \mathbb{R}^+$
- Kroky:
    - $\beta_i = 0$
    - $K_{ij} = \phi(x_i)^T \phi(x_j)$
    - Dokud jsem nezkonvergoval:

        - Tedy pro $i \in \{1, \dots, b\}$ se upraví $\beta_i = \beta_i - \frac{\alpha}{b}(\sum_j^n (\beta_j K_{ij}) - t_i)$
- Predikce: $y(z) = \phi(z)^T w = \sum_i \beta_i \phi(z)^T \phi(x_i)$
- **Bias**: průměr přes trénovací data

Kernel trick

- $\phi(x)^T \phi(y) = 1 + x^T y + (x^T y)^2 + \dots + (x^T y)^p$

Kernely

- Polynomiální kernel stupně alespoň d $K(x, y) = (\gamma x^T y)^d$
- Polynomiální kernel stupně alespoň d $K(x, y) = (\gamma x^T y + 1)^d$
- Gaussian radial basis kernel (RBF) $K(x, y) = e^{-\gamma \|x-y\|^2}$
    - Odpovídá "nekonečným" polynomial features

SVN

- Support vector machine
- Cílem je při binární klasifikaci najít takovou dělící nadrovinu, která je maximálně vzdálená od obou skupin (**maximum margin**)
    - Vzdáleností rozumíme normu rozdílu prvku skupiny a jeho projekce na dělící nadrovinu → chceme ji maximalizovat pro obě skupiny
- Vzdálenost bodu od dělící nadroviny $r = \frac{|y(x_i)|}{\| w \|}$
    - Chceme tedy $\arg\max_{w,b} \frac{1}{\| w \|} \min_i [t_i(\phi(x_i)^T w + b)]$ za předpokladu $\frac{|y(x_i)|}{\| w \|} = \frac{t_i y(x_i)}{\| w \|}$, což platí, pokud jsou všechny vzorky správně klasifikovány
- Protože model je invariantní k násobení w, b, můžu předpokládat, že nejbližší bod dělící nadroviny bude splňovat $t_i y(x_i) = 1$
    - S předpokladem $t_i y(x_i) \geq 1$ můžu upravit $\arg\min_{w,b} \frac{1}{2} \| w \|^2$
        - Použiju větu o Lagrangeových multiplikátorech
- K řešení SVM se obvykle používá **SMO** (Sequential Minimal Optimization) agoritmus

Coordinate descent agloritmus

- Chceme řešit $\arg\min_w L(w_1, \dots, w_D)$
- Kroky
    - Dokud jsem nezkonvergoval
        - $\forall i \in \{1, \dots, D\}$ nastav $w_i \leftarrow \arg\min_{w_i} L(w_1, \dots, w_D)$

SMO algoritmus

- https://en.wikipedia.org/wiki/Sequential_minimal_optimization

Strojové zpracování textu

- **Term** - slovo, obecně sekvence znaků
- Obvykle máme featuru pro každý term. Její hodnotu lze reprezentovat různě.
    - Binárně - 1/0, zda se term v dokumentu vyskytuje

- Frekvence (TF) - $TF(t,d) = \frac{|\{t:t\in d\}|}{|d|}$
- Inverse document frequency (IDF) - $IDF(t) = \log \frac{\#\ docs}{\#\ docs\ containing\ t} = I(P(t \in d))$
  - Vyjadřuje, jak "unikátní" slovo je
    - Například "a" bude téměř v každém a často, zatímco "mastodont" je unikátnější
- Nejčastěji se používá **TF-IDF** $= TF \cdot IDF$

TD-IDF

- Podmíněná entropie
  $H(Y|X) = \mathbb{E}_{x,y}[I(y|x)] == \mathbb{E}_{x,y}[-\log P(y|x)] \sum_{x,y} P(x,y)P(y|x)$
- Pak $I(x,y) = H(Y) - H(Y|X) = \mathbb{E}_{x,y}[\frac{P(x,y)}{P(x)P(y)}]$ je **mutal information**
  - Je symetrická $H(Y) - H(Y|X) = H(X) - H(X|Y)$
  - Je nezáporná $I(X,Y) \geq 0$
  - Pokud jsou X, Y jsou stejné (tj. náhodné veličiny jsou nezávislé $P(X,Y) = P(X)P(Y)$), pak je nulová
- Odvození TD-IDF z mutal information
  - Buď D množina dokumentů velikost N a T množina termů. Dokumenty vybírám uniformě.
  - Pak $P(d) = 1/N$ a $I(d) = H(d) = \log N$
  - $P(d|t) = 1/\left|\{d \in D : t \in d\}\right|$ a $I(D|T=t) = \log\left|\{d \in D : t \in d\}\right|$
  - $I(d) - I(d|t) = H(D) - H(D|T=t) = \log \frac{N}{|\{d\in D: t\in d\}|} = IDF(t)$
    - Aneb kolik self-ifnormation se dozvím o dokumentu, když v něm je nějaký term - relevance mezi dokumentem a termem

Bayesovská pravděpodobnost

- Pravděpodobnost uvažuje jako míru nejistoty
- $P(B|A) = \frac{P(A|B)P(A)}{P(B)}$
- **Vsuvka:** Franta je spořádaný a klidný člověk, která má rád pořádek. Je spíš knihovník, nebo farmář?
  - Ačkoliv popis sedí spíš na knihovníka, musíme uvážit, že farmářů je zhruba 30krát více
- $p(w|X) \propto p(X|w) \cdot p(w)$, kde:
  - $p(w|X)$ - posterior
  - $p(X|w)$ - likelihood
  - $p(w)$ - prior
- Pak **Maximum a posterior** $w_{MAP} = \arg\max_w p(w|X) = \arg\max_w p(X|w)p(w)$