24. Generativní modely a diskriminativní přístup ke klasifikaci (gaussovský klasifikátor, logistická regrese, ...).

Ve strojovém učení říkáme, že program se učí ze zkušenosti E s ohledem na vykonávání úlohy T měřené výkonnostním kritériem P, jestliže jeho výkonnost měřená P na úlohách typu T se se zkušeností E zlepšuje.

Typickým úkolem je vytrénovat model, který na základě nějakého vstupu dá nějaký výstup. Vstupem jsou obvykle D-rozměrné vektory čísel. Pokud máme vstupních dat více (např. trénovací sadu), zapisujeme jednotlivé vektory sloupcově do matice:

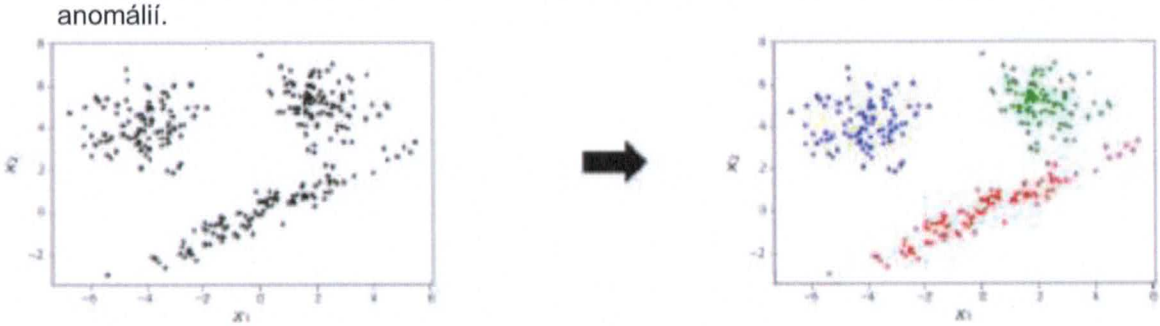
$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix}$$

A set of N input observations (e.g. set of training examples) will be then represented by a matrix:

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] = \begin{bmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & \cdots & x_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ x_{1D} & x_{2D} & \cdots & x_{ND} \end{bmatrix}$$

Rozlišujeme následující typy učících algoritmů:

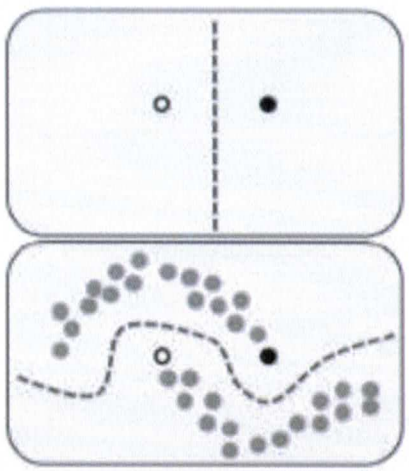
- **supervised learning (učení s učitelem)** – trénovacím vstupem jsou dvojice (vstup, očekávaný výstup) – anotovaná data. Např. klasifikace (vstupy patří do nějakých tříd, chceme pro nový vstup nalézt, do které z tříd nejspíše patří), regrese (předpovídáme nejpravděpodobnější hodnotu nějaké funkce)
- **unsupervised learning (učení bez učitele)** – trénovacím vstupem jsou neanotovaná data (bez očekávaného výstupu ke každému trénovacímu datu), např. clustering, detekce anomálií.



- **semi-supervised learning** – některá data jsou anotovaná, ale máme velké množství trénovacích dat, která nejsou vůbec anotovaná

- Unannotated examples can help to find better decision boundary between classes

- There is lots of unannotated data available on the internet
  - Text
  - Photos and other images
  - Speech and other recordings
  - ...

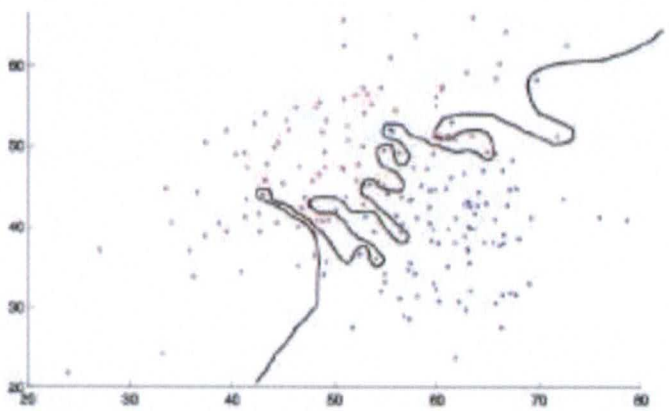


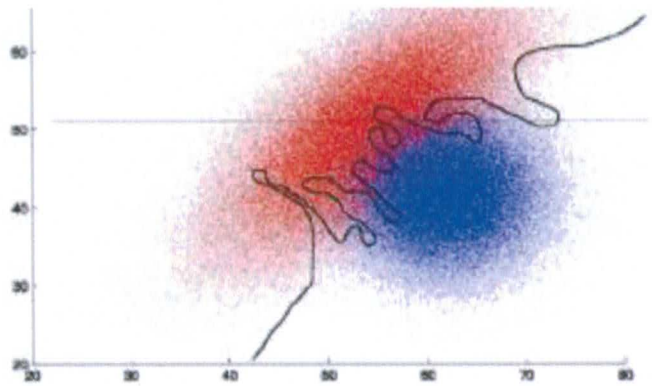
- **reinforcement learning (posilované učení)** – učení probíhá na základě pozitivních nebo negativních odměn na základě provedených akcí

Klasifikace

~ Cílem klasifikace je najít rozhodovací hranici, která odděluje data jedné třídy od ostatních tříd.

Na následujícím obrázku jsme sice nalezli rozhodovací hranici, ta velmi přesně rozděluje trénovací data. Problémem však je, že dobře negeneralizuje celková data. Data totiž byla vygenerována z 2D Gaussovských rozložení a rozhodovací hranice je tedy příliš "agresivní" a snaží se zahrnout i outliers (odchýlené hodnoty) vygenerované z rozložení. To znamená, že pokud by klasifikátor dostal nová data pro klasifikaci, neklasifikoval by je velmi dobře, protože je přetrénovaný.





k-nejbližších sousedů

Jedním z nejjednodušších algoritmů pro klasifikaci je algorithmus k-nejbližších sousedů. Jedná se o **neparametrický klasifikátor** – neučíme se žádné parametry modelu, efektivně jsou všemi parametry všechna trénovací data. To je zároveň jeho hlavním problémem – klasifikátor si musí **zapamatovat velké množství dat**. Pro přiřazení třídy vstupu **najde k nejbližších sousedů** (lze efektivně implementovat kD-stromem) a vybere nejvíce reprezentovanou třídu. Klasifikátor je **diskriminativní**, není generativní – dokážeme pouze na základě vstupních dat přiřadit třídu, nejsme však schopni generovat algoritmem nová data, jako je možné v případě třeba Gaussovského klasifikátoru (viz dále).

Problémem je, že pokud máme více-dimenzionální data, kde každá dimenze reprezentuje nějaký příznak objektu (např. jedna dimenze výška, druhá váha), nemusí být dimenze poměřitelné (např. z důvodu různých jednotek) – při výpočtu vzdálenosti (např. Euklidovské vzdálenosti) by jedna dimenze mohla druhou dominovat a mít zásadně větší vliv na výsledek klasifikátoru. Proto je typicky nutné data nějak **normalizovat** (např. do rozsahu <0, 1>).

Generativní klasifikátor

U generativního klasifikátoru **modelujeme rozložení pravděpodobnosti sledovaných dat**. Tím pádem pokud bychom chtěli, mohli bychom **vygenerovat nová data**, která se podobají trénovacím datům (od toho název generativní). **POZOR!** vygenerování dat nemá z pohledu trénování žádnou výhodu a nedostaneme vyšší přesnost.

např. gaussovský klasifikátor

Nejprve budeme tvořit jednoduchý klasifikátor nad dvěma třídami, přičemž naše pozorování budou pouze diskrétní. Např. podle váhové kategorie chceme klasifikovat, zda objekt je jablko nebo granát. Naše testovací data by mohla vypadat třeba takto:

Generativní modely se učí pravděpodobnostní distribuce jednotlivých tříd a pravděpodobnost této třídy, z čehož pomocí Bayesova pravidla odvodí $p(c|x)$.

U diskriminativních modelů se učí rozhodovací hranice mezi jednotlivými třídami, tedy buďto přímo $p(c|x)$ nebo se ani nepokouší modelovat $p(c|x)$.

např. SVM, RNN, logistická regrese

1	6	12	15	12	2	2	50
4	22	50	14	6	3	1	100
lightest 0.0 - 0.1	*lighter* 0.1 - 0.2	*light* 0.2 - 0.3	*middle* 0.3 – 0.4	*heavy* 0.4 – 0.5	*heavier* 0.5 – 0.6	*heaviest* 0.6 – 0.7	[kg]

Protože chceme modelovat rozložení pravděpodobnosti dat, je důležitý výpočet některých pravděpodobností:

- **společná (joint) probability** – pravděpodobnost, že dva jevy nastanou zároveň, typicky že objekt je nějaké třídy a zároveň má určitou vlastnost. Např. P(grenade, heavy) = 12/150
- **marginální pravděpodobnost** – **apriorní pravděpodobnost** třídy. Např. P(grenade) = 50/150
- **podmíněná pravděpodobnost** – **aposteriorní pravděpodobnost** třídy za předpokladu, že nastalo nějaké pozorování, např. P(grenade|heavy) = 12 / (12 + 6) = 12/18, protože je 18 vzorků s pozorováním heavy, ale pouze 12 z nich je zároveň granát. Základem je vzorec pro podmíněnou pravděpodobnost P(A|B) = P(A, B)/P(B). **Klasifikátor maximum aposteriori** vybírá třídu s nejvyšší aposteriorní pravděpodobností.

Základní pravidla pro pravděpodobnosti:

Sum rule:

$$P(x) = \sum_y P(x, y)$$

Product rule:

$$P(x, y) = P(x|y)P(y) = P(y|x)P(x)$$

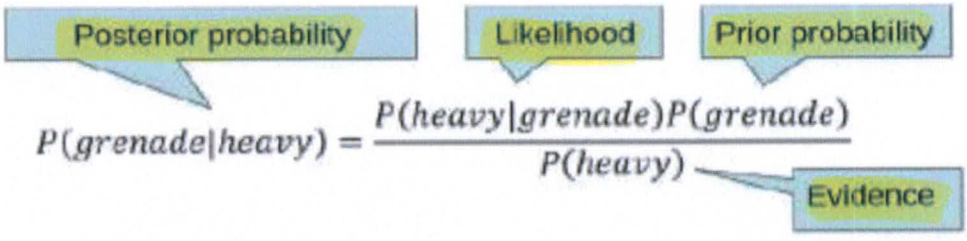
Bayes rule:

$$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$

Intuitivně:

- **sum rule** říká, že pro celkovou pravděpodobnost řádku spočítáme jednotlivé společné pravděpodobnosti v řádku
- **product rule** vychází ze vzorce pro sdílenou pravděpodobnost. Intuitivně, aplikováno např. na P(grenade, heavy) = P(grenade|heavy)*P(heavy) – s pravděpodobností P(heavy) jsem ve sloupečku heavy a chci vědět, s jakou pravděpodobností jsem v prvním řádku, tzn. v políčku (grenade, heavy) – to odpovídá podmíněné pravděpodobnosti granátu, když jsem ve sloupečku heavy.
- **Bayesovo pravidlo** vychází z product rule, z části P(x|y)P(y)=P(y|x)P(x) a je základem pro aposteriorní odhady

Pokud chceme postavit klasifikátor pro klasifikaci granátů a jablek na základě váhy, zajímá nás aposteriorní pravděpodobnost P(grenade|weight), resp. P(apple|weight) (protože klasifikujeme mezi dvěma třídami, jsou komplementem). Evidence slouží pro normalizaci výsledku, aby P(grenade|heavy) + P(apple|heavy) = 1.

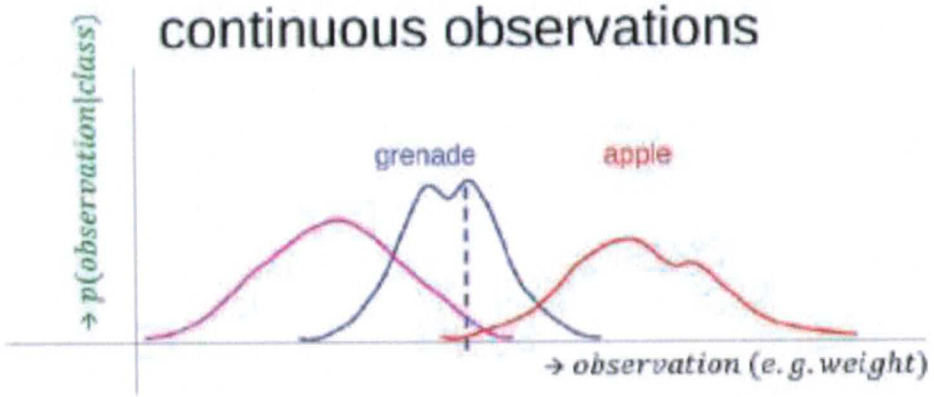


- The evidence can be evaluated using the sum and product rules in terms of likelihoods and priors:

$$P(heavy) = P(heavy|grenade)P(grenade) + P(heavy|apple)P(apple)$$

Pro diskrétní vstup by klasifikace byla velmi jednoduchá, zajímavější je spojitý vstup. Vstupem už tedy není váhová kategorie, ale např. váha a chceme opět odhadnout pravděpodobnost, že pro danou váhu je objekt granát nebo jablko. Výše uvedené vztahy stále platí, operátor sumy však nahradíme integrálem. Bayesův vzorec však normálně platí. Pro klasifikaci tedy musíme znát podmíněnou pravděpodobnost P(observation|class) a apriorní pravděpodobnosti jednotlivých tříd:

Classification - Generative model for continuous observations



Maximum a-posteriori classification rule says: "select the more likely class"

$$P(class|observation) = \frac{p(observation|class)P(class)}{p(observation)}$$

$$P(observation) = \sum_{class} p(observation|class)P(class)$$

Pokud bychom měli vícerozměrná data, modelovali bychom podmíněnou pravděpodobnost vektoru na třídě – P(**x** | class), kde **x** = [x1, x2, … , xn]

Odhad rozložení pravděpodobnosti p(observation|class) z testovacích dat

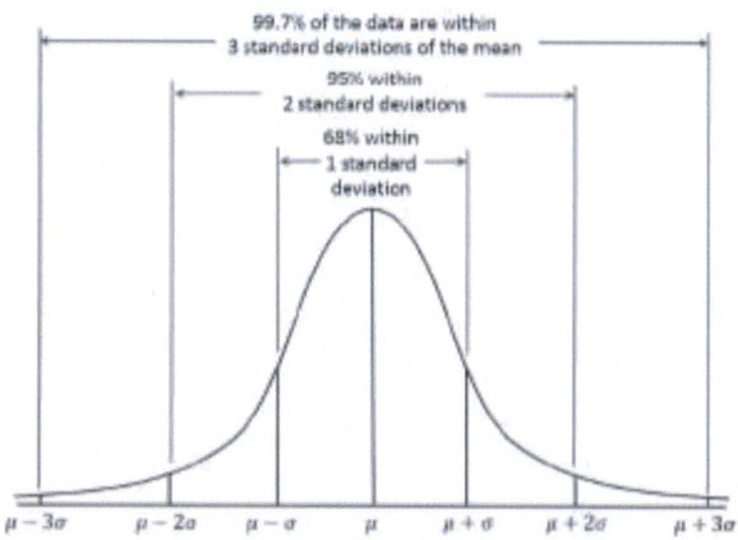
Jak bylo naznačeno výše, pro aposteriorní klasifikaci pomocí Bayesova vzorce musíme znát rozložení pravděpodobnosti p(observation|class) a apriorní pravděpodobnosti. Rozložení pravděpodobnosti však typicky přímo neznáme – máme pouze nějaká trénovací data, ze kterých jej musíme odhadnout.

Nejprve si vybereme nějaké rozložení, kterému si myslíme, že data odpovídají a následně se na základě dat pokusíme odhadnout jeho parametry. Jedním z nejtypičtější příkladů je normální (Gaussovské rozložení) – od toho je odvozen Gaussovský klasifikátor. Důvodem je, že jevy v přírodě často odpovídají tomuto rozložení. Zároveň dle Centrální Limitní Věty pokud sečteme hodnoty mnoha nezávislých náhodných proměnných, dostáváme Gaussovské rozložení. Abychom mohli využít Gaussovské rozložení, musíme z trénovacích dat odhadnout střední hodnotu a rozptyl.

Jednorozměrné normální rozložení

(Je dobré alespoň vědět, že vzorec má nějakou exponenciální část, která tvoří typický tvar Gaussovské funkce a normalizační část, která tento "klobouk" normalizuje aby jeho integrál byla 1 - rozdělení pravděpodobnosti.)

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



ML estimates of parameters

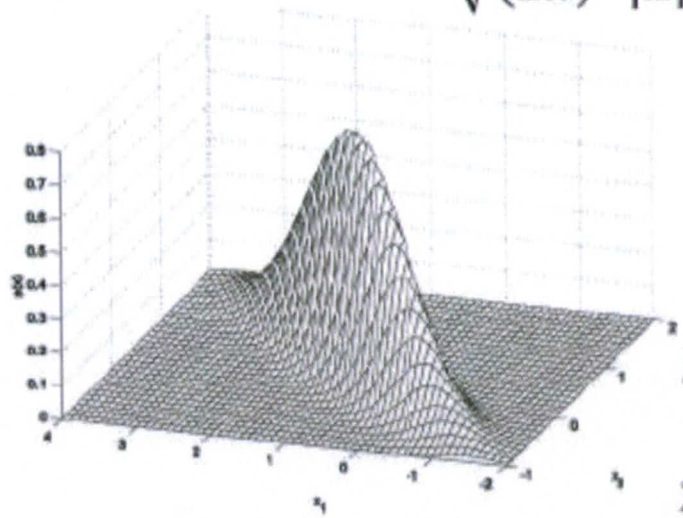
$$\mu = \frac{1}{N}\sum_n x_n$$

$$\sigma^2 = \frac{1}{N}\sum_n (x_n - \mu)^2$$

Vícerozměrné normální rozložení

Vícerozměrné normální rozložení je zobecněním základního normálního rozložení. Tentokrát však namísto skalární střední hodnoty máme vektor a namísto rozpytlu kovarianční matici sigma. V ní jsou na diagonále odchylky v jednotlivých směrech a v ostatních prvcích korelace (matice je symetrická dle diagonály)

$$p(x_1, \ldots, x_D) =$$
$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

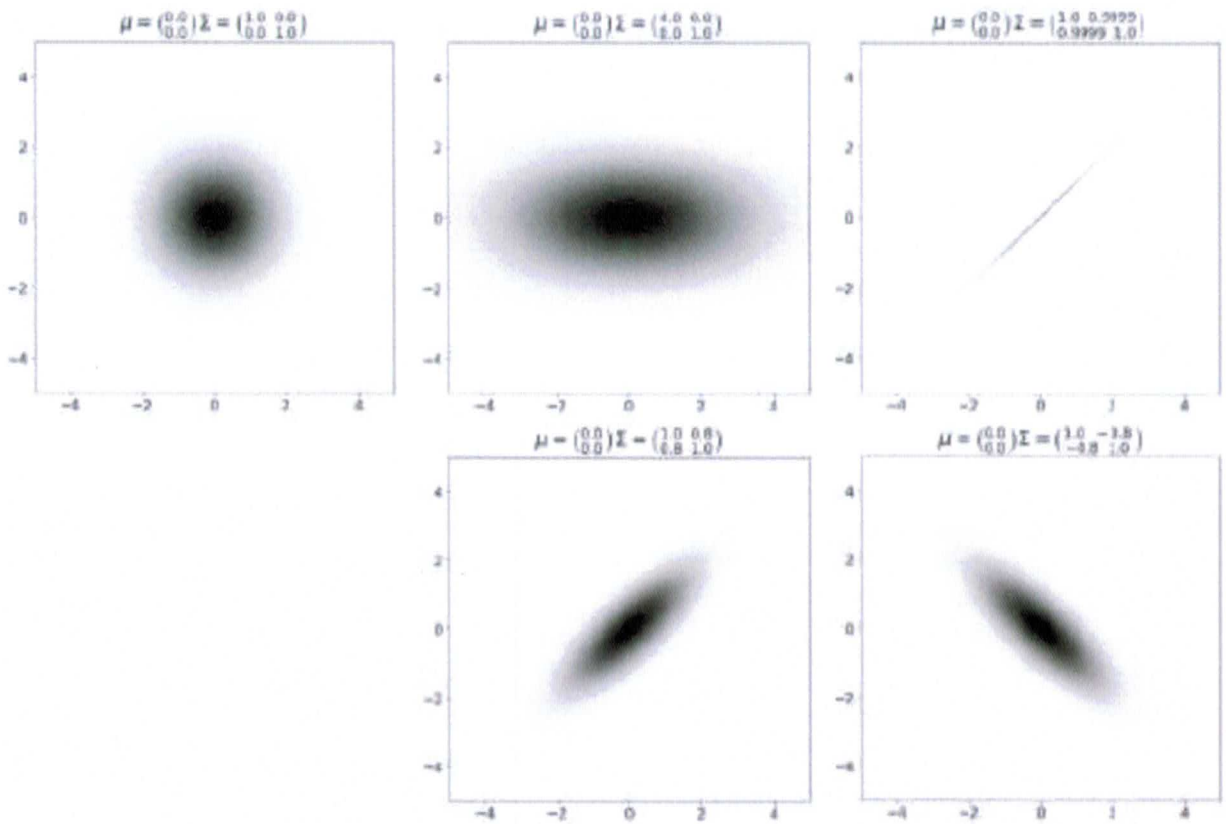


ML estimates of parameters

$$\boldsymbol{\mu} = \frac{1}{N}\sum_n \mathbf{x}_n$$

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_n (\mathbf{x}_n - \boldsymbol{\mu})(\mathbf{x}_n - \boldsymbol{\mu})^T$$

Příklady různých rozložení. V prvním řádku a druhém sloupci můžeme vidět, že na ose x je větší rozptyl, tedy rozložení je více rozpláclé. Ve druhém řádku a druhém sloupci vidíme, že x a y mají pozitivní korelaci.



Maximum Likelihood Estimation (ML)

Ukažme si nyní, jak se odvodí např. vzorec pro střední hodnotu Gaussovského rozložení. Odvození Maximum Likelihood (ML) odhadů parametrů rozdělení vychází z nezávislosti výběru dat z náhodného rozložení pravděpodobnosti, tzn. pokud máme trénovací data xi z nějakého (v našem případě Gaussovského) rozložení, pak p(x1, x2, x3, …) = p(x1)p(x2)p(x3)... . S násobením se však pracuje obtížně, proto budeme pracovat s logaritmy pravděpodobností – výsledek zůstane stejný, namísto násobení však budeme pracovat se součty logaritmů.

Mějme tedy rozložení p(**x** | η) s parametry η (střední hodnota a rozptyl pro normální rozložení) a nějaká testovací data **X** = [**x1, x2, …**]. Maximum-likelihood odhad parametrů η můžeme určit jako:

$$\widehat{\boldsymbol{\eta}}^{ML} = \arg\max_{\boldsymbol{\eta}} p(\mathbf{X}|\boldsymbol{\eta}) = \arg\max_{\boldsymbol{\eta}} \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\eta})$$

Jinými slovy hledáme parametry, pro které trénovací data maximalizují funkci hustoty pravděpodobnosti. Úprava na násobení pravděpodobností vychází z nezávislosti, jak bylo

vysvětleno výše. Celý výraz můžeme zlogaritmovat (logaritmus zachovává maximum) a logaritmus součinu převést na sumu logaritmů. Když zlogaritmujeme funkci hustoty pravděpodobnosti normálního rozdělení, můžeme navíc přesunout exponent. Určeme nyní logaritmus normálního rozdělení log p(x): $ln\,(2\pi\sigma)^{-1/2}e^{-\frac{(x-\mu)^2}{2\sigma^2}} = ln(2\pi\sigma)^{-1/2} + ln\,e^{-\frac{(x-\mu)^2}{2\sigma^2}} =$

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

$$= -\frac{1}{2}ln\,(2\pi\sigma) - \frac{1}{2\sigma^2}(x^2 - 2x\mu + \mu^2)$$

Pokud dosadíme do vzorce pro ML odhad, dostáváme:

$$\arg\max_{\mu,\sigma^2} p(\mathbf{x}|\mu,\sigma^2) = \arg\max_{\mu,\sigma^2} \log p(\mathbf{x}|\mu,\sigma^2) = \arg\max_{\mu,\sigma^2} \sum_n \log \mathcal{N}(x_n;\mu,\sigma^2)$$

$$= \arg\max_{\mu,\sigma^2}\left(-\frac{1}{2\sigma^2}\sum_n x_n^2 + \frac{\mu}{\sigma^2}\sum_n x_n - N\frac{\mu^2}{2\sigma^2} - \frac{N\log(2\pi)}{2}\right)$$

Chceme určit, pro které parametry je výraz maximální – spočítáme parciální derivaci dle parametrů a položíme rovno nule. Například pro odhad střední hodnoty (pro rozptyl by postup byl podobný, ovšem derivovali bychom dle rozptylu):

$$\frac{\partial}{\partial\mu}\log p(\mathbf{x}|\mu,\sigma^2) = \frac{\partial}{\partial\mu}\left(-\frac{1}{2\sigma^2}\sum_n x_n^2 + \frac{\mu}{\sigma^2}\sum_n x_n - N\frac{\mu^2}{2\sigma^2} - \frac{N\log(2\pi)}{2}\right)$$

$$= \frac{1}{\sigma^2}\left(\sum_n x_n - N\mu\right) = 0 \;\Rightarrow\; \hat{\mu}^{ML} = \frac{1}{N}\sum_n x_n$$

Kategorické rozložení

Dalším možným rozložením, kterým se data mohou řídit, je tzv. kategorické (diskrétní) rozložení, kdy data spadají do nějaké kategorie. Z trénovacích dat tedy můžeme odvodit pravděpodobnost jednotlivých kategorií, které definují chování (parametry) rozložení.

4	22	50	14	6	3	1	100
lightest	*lighter*	*light*	*middle*	*heavy*	*heavier*	*heaviest*	
0.0 - 0.1	0.1 - 0.2	0.2 - 0.3	0.3 – 0.4	0.4 – 0.5	0.5 – 0.6	0.6 – 0.7	[kg]

$$p(x|\boldsymbol{\pi}) = \mathrm{Cat}(x|\boldsymbol{\pi}) = \pi_x$$

Pravděpodobnost daných trénovacích dat můžeme spočítat jako:

$$P(\mathbf{x}|\boldsymbol{\pi}) = \prod_n \text{Cat}(\mathbf{x}_n|\boldsymbol{\pi}) = \prod_n \pi_{x_n} = \prod_c \pi_c^{m_c}$$

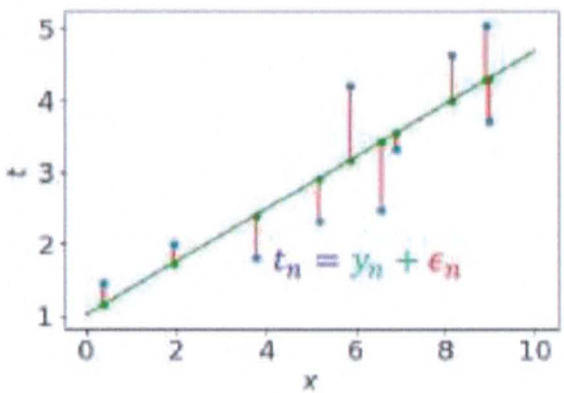
Např. pro výše uvedený příklad bychom dostali $\pi_{lightest}{}^4 * \pi_{lighter}{}^{22} *...$
Maximální věrohodný odhad parametrů π je velmi jednoduchý – podělíme počet trénovacích dat každé kategorie celkovým počtem trénovacích dat. Pro zadaný příklad bychom tedy měli $\pi = [4/100, 22/100, 50/100, 14/100, 6/100, 3/100, 1/100]$

Gaussovský klasifikátor

Spojme nyní vše výše uvedené do Gaussovského klasifikátoru. Pro Gaussovský klasifikátor musíme znát **apriorní pravděpodobnost jednotlivých tříd** (tu můžeme odhadnout z trénovacích dat dle poměru tříd v testovacích datech). Dále musíme pomocí **Maximum Likelihood Estimation** pro každou třídu určit **rozložení hustoty pravděpodobnosti** – pro Gaussovský klasifikátor určíme **pro každou třídu střední hodnotu a rozptyl**. Na základě Bayesova vzorce pak z těchto informací vypočítáme pro nové vstupní dato **hodnotu funkce hustoty pravděpodobnosti** jednotlivých tříd, dle ní dopočítáme **aposteriorní pravděpodobnosti** a vybereme **třídu s maximální aposteriorní pravděpodobností** (příp. necháme výstup jako pravděpodobnost – výstupem může být pravděpodobnost nebo třída).

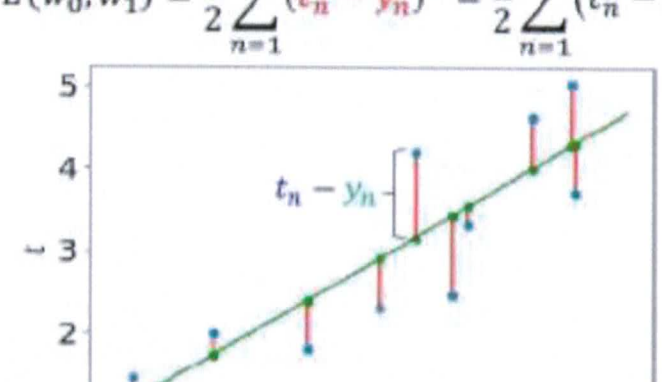
Lineární regrese

V lineární regresi je **cílem najít lineární funkci** y = f(x) = $w_1 x + w_0$, která co nejlépe odpovídá trénovacím datům. ***POZOR**: Nejedná se o algoritmus pro klasifikaci, nýbrž regresi. Ovšem zavádíme ji zde kvůli návaznosti na lineární logistickou regresi a protože tato návaznost byla i v SUI přednáškách (bližší info k regresi v otázce 51).* Základním předpokladem je, že trénovací data mají **lineární trend**, až na nějaký **Gaussovský šum** ϵ, trénovací data t tedy mají formu $t = y + \epsilon$:



Hledáme parametry w_1 a w_0, které **minimalizují součet objektivní funkce**, pro Gaussovský šum využíváme jako objektivní funkci střední kvadratickou chybu (sum-of-squares error), tzn. suma čtverců odchylek:

$$E(w_0, w_1) = \frac{1}{2}\sum_{n=1}^{N}(t_n - y_n)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w}\right)^2$$



Pro odvození ideálních vah w musíme minimalizovat chybu, musíme tedy určit derivaci dle vektoru vah a položit ji rovnu 0:

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} E(\mathbf{w}) &= \frac{\partial}{\partial \mathbf{w}} \frac{1}{2}\sum_{n=1}^{N}\left(t_n - \hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w}\right)^2 \\
&= \frac{1}{2}\sum_{n=1}^{N}\frac{\partial}{\partial \mathbf{w}}\left(t_n - \hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w}\right)^2 \\
&= \sum_{n=1}^{N}\left(t_n - \hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w}\right)\frac{\partial}{\partial \mathbf{w}}\left(t_n - \hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w}\right) \\
&= \sum_{n=1}^{N}\left(\hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w} - t_n\right)\hat{\mathbf{x}}_n \\
&= \sum_{n=1}^{N}\hat{\mathbf{x}}_n\hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w} - \sum_{n=1}^{N} t_n\hat{\mathbf{x}}_n = 0
\end{aligned}$$

$$\begin{aligned}
\left(\sum_{n=1}^{N}\hat{\mathbf{x}}_n\hat{\mathbf{x}}_n^{\mathrm{T}}\right)\mathbf{w}^* &= \sum_{n=1}^{N} t_n\hat{\mathbf{x}}_n \\
\mathbf{w}^* &= \left(\sum_{n=1}^{N}\hat{\mathbf{x}}_n\hat{\mathbf{x}}_n^{\mathrm{T}}\right)^{-1}\sum_{n=1}^{N} t_n\hat{\mathbf{x}}_n \\
&= \left(\hat{\mathbf{X}}\hat{\mathbf{X}}^{\mathrm{T}}\right)^{-1}\hat{\mathbf{X}}\mathbf{t} \\
&= \hat{\mathbf{X}}^{\dagger}\mathbf{t}
\end{aligned}$$

Where † means *Moore-Penrose pseudo-inverse*, $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_N]$ is matrix of all extended inputs and $\mathbf{t} = [t_1, t_2, \ldots, t_N]^{\mathrm{T}}$ is vector of all desired outputs

Lineární regresi můžeme využít i k naučení se složitějších funkcí – **polynomiální regrese** je pouze speciálním případem lineární regrese (název lineární vychází z toho, že funkce je lineární vzhledem ke koeficientům – jedná se o lineární kombinaci funkcí s váhami). Obecně se tedy lineární regresí můžeme naučit nějakou polynomiální funkci, příp. i složitější funkci, třeba y = w1log x + w0:

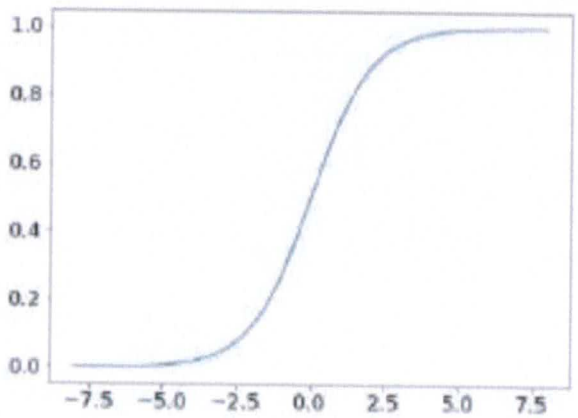
$$y = \hat{\mathbf{x}}^{\mathrm{T}}\mathbf{w} = w_0 + w_1 x + w_2 x^2 + \cdots + w_K x^K$$

Lineární logistická regrese

Logistická regrese je **diskriminativní** model, to znamená, že se snažíme přímo **naučit funkci p(class|x)**. To je rozdíl od generativních modelů, kde odhadujeme pravděpodobnosti p(x | class) a p(class) a až na základě toho p(class | x). Pokud bychom chtěli srovnat diskriminativní a generativní modely:

- generativní mají menší tendenci overfitovat při malém množství dat
- diskriminativní fungují lépe v případě dostatečného množství dat

V rámci lineární logistické regrese aplikujeme na vstup ke klasifikaci funkci, která je podobně jako v lineární regresi vytvořena lineární kombinací natrénovaných vah a funkcí. Výstup této funkce poté za účelem binární klasifikace převedeme do rozsahu <0, 1>, abychom modelovali funkci p(class|x), např. funkcí logistické sigmoidy:



$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

$$P(c = 1|\mathbf{x}) = \sigma(\hat{\mathbf{x}}^{\mathrm{T}}\mathbf{w})$$

Pravděpodobnost druhé třídy je pak komplementem P(c = 1 | x). Podobně jako u lineární regrese musíme natrénovat parametry **w** , tak abychom minimalizovali chybu. Můžeme opět použít střední kvadratickou chybu, alternativně maximum likelihood estimation (t_n je reálný label data – 0 nebo 1, sigmoida nám vrací předpovídaný label. Pravděpodobnost správného data je v případě $t_n = 1$ výstup sigmoidy, v případě $t_n = 0$ je to komplement – o to se stará umocnění na t_n):

$$P(\mathbf{t}|\mathbf{X}) = P(t_1, \ldots, t_N|\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_n P(t_n|\mathbf{x}_n) = \prod_n \sigma(\hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w})^{t_n} \left(1 - \sigma(\hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w})\right)^{(1-t_n)}$$

Alternativně je možné minimalizovat ekvivalentní chybu, zlogaritmováním a znegováním dostáváme tzv. cross-entropii:

$$E(\mathbf{w}) = -\ln P(\mathbf{t}|\mathbf{X}) = \sum_{n=1}^{N} t_n \ln \sigma(\hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w}) + (1 - t_n) \ln\left(1 - \sigma(\hat{\mathbf{x}}_n^{\mathrm{T}}\mathbf{w})\right)$$

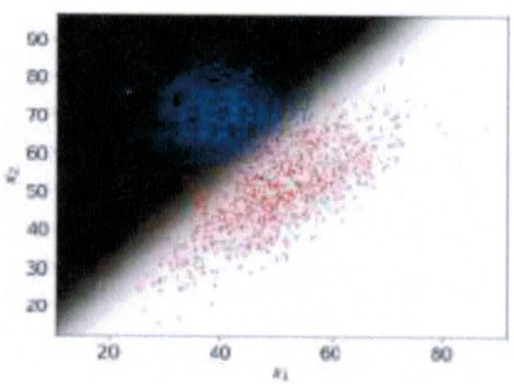
Pro tuto rovnici ovšem není možné nalézt optimální váhy w podobně jako v případě lineární regrese, kdy pro výpočet ideálních parametrů máme vzorec. Je nutné využít numerickou

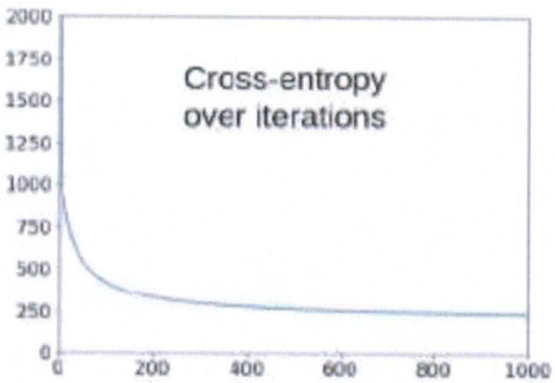
optimalizaci, např. pomocí metody gradientního sestupu (viz otázka 25). Využívá se následující vzorec, kde η je learning rate a ΔE je derivace cross-entropie:

$$\mathbf{w}^{\tau+1} = \mathbf{w}^{\tau} - \eta \nabla E(\mathbf{w}^{\tau})$$

Příklad iterativního trénování logistické regrese s jednoduchým polynomem, který má 3 parametry:

$$P(c = 1|\mathbf{x}) = \sigma(\hat{\mathbf{x}}^{\mathrm{T}}\mathbf{w}) = \sigma(w_1 x_1 + w_2 x_2 + w_0)$$





Pokud v textu najdete chybu, nebudete něčemu rozumět nebo budete mít dojem, že by bylo vhodné něco doplnit, kontaktujte na discordu uživatele Fifinas.