evropský sociální fond v ČR

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

UNIVERSITAS MASARYKIANA BRUNENSIS

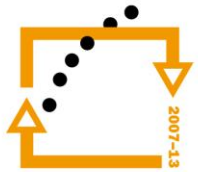
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Matematické modely ve financích

Martin Řezáč

2012

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY



OP Vzdělávání
pro konkurenceschopnost

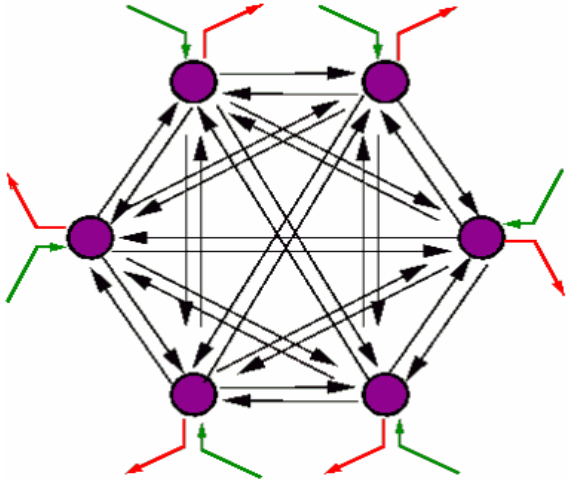
INVESTICE
DO ROZVOJE
VZDĚLÁVÁNÍ

Tento učební text vznikl
za přispění Evropského
sociálního fondu a
státního rozpočtu ČR
prostřednictvím
Operačního programu
Vzdělávání pro
konkurenceschopnost v
rámci projektu
Univerzitní výuka
matematiky v měnícím
se světě
(CZ.1.07/2.2.00/15.0203).

Obsah:

1. Vícevrstvé NN, Backpropagation, Madaline 4
2. Asociativní neuronové sítě, Hebbův zákon, Kohonenovy mapy, LVQ 32
3. RBF sítě, Modulární NN, Hammingova síť 64
4. Vyhodnocení kvality prediktivního modelu 92
5. Úvod do teorie portfolia 134

1. Vícevrstvé NN, Backpropagation, MADALINE



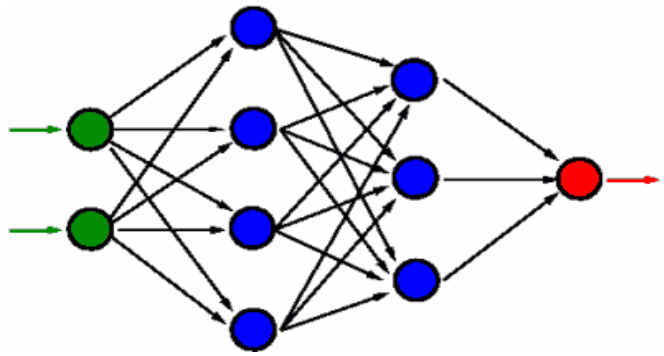
Neuronová síť (NS)

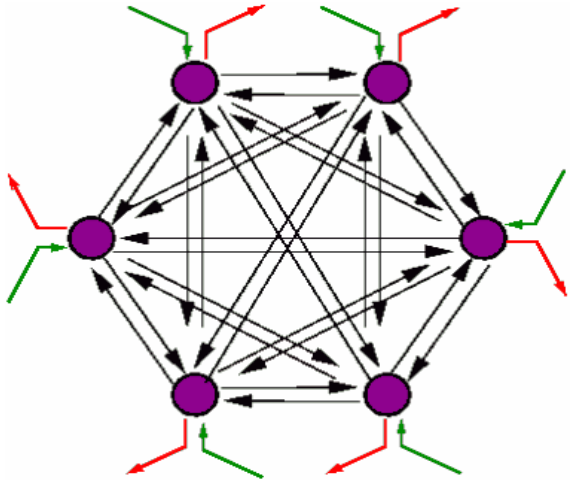
Neuronová síť se v čase vyvíjí, mění se propojení a stav neuronů a adaptují se váhy. V souvislosti se změnou těchto charakteristik v čase je účelné rozdělit celkovou dynamiku NS a pracovat v třech režimech (dynamikách):

- **Organizační** – změna topologie
- **Aktivní** – změna stavu
- **Adaptivní** – změna konfigurace

Organizační dynamika NS

- Specifikuje architekturu sítě
  - Dopředná, acyklická (feed-forward)
  - Rekurentní, cyklická





Aktívní dynamika NS

- Specifikuje **počáteční stav** NS a spůsob jeho změny v čase při pevných ostatních charakteristikách (topologie a konfigurace).
- Nastaví se stavy vstupních neuronů **(vstup sítě).**
- Po inicializaci vstupů nastává vlastní výpočet.
- Stav výstupních neuronů, který se v čase mění je tzv. **výstup** NS, který je po čase konstantní a NS tak v aktivním režimu realizuje nějakou funkci na výstupním prostoru **(funkce NS).**
- Aktivní dynamika určuje i funkci jednoho neuronu. Např.:

$$\sigma(\xi) = \begin{cases} 1 \ldots \xi \geq 0 \\ 0 \ldots \xi < 0 \end{cases} \qquad \xi = \sum_{i=0}^{n} \omega_i x_i$$

Adaptivní dynamika NS

- Specifikuje **počáteční konfiguraci** NS a způsob jakým se mění váhy v síti v čase.
- Všechny možné konfigurace tvoří tzv. **váhový prostor.**
- V adaptivním režimu se tedy nastaví váhy všech spojů a po inicializaci konfigurace probíhá vlastní adaptace (jejím cílem je najít konfiguraci, která v aktivním režimu realizuje předepsanou funkci).
- Učení s učitelem vs. bez učitele.

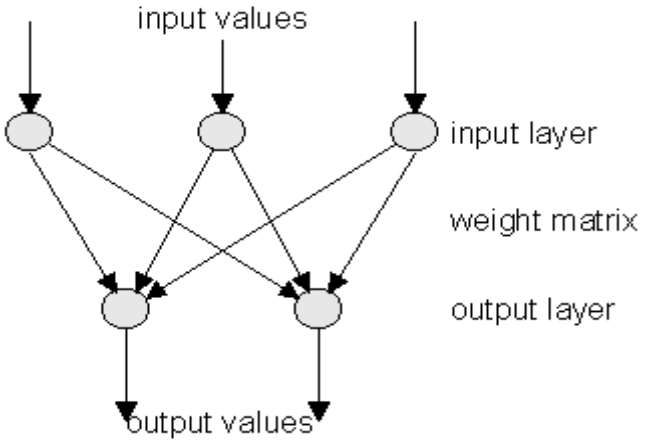
Síť perceptronů I.

- **Organizační** dynamika specifikuje pevnou architekturu jednovrstvé sítě ***n*-*m***, tedy síť se skládá z ***n*** vstupních neuronů, z nichž každý je vstupem každého z ***m*** výstupních neuronů.

$$\mathbf{x} = (x_1, \ldots, x_n) \in R^n$$

$$\mathbf{y} = (y_1, \ldots, y_m) \in \{0,1\}^m$$

$$\mathbf{w} = (\omega_{10}, \ldots, \omega_{1m}, \ldots, \omega_{m0}, \ldots, \omega_{mn})$$



Síť perceptronů II.

- **Aktivní** dynamika (určuje způsob výpočtu funkce sítě) – reálné stavy neuronů na vstupní vrstvě se nastaví na vstup a výstupní neurony počítají svůj binární stav, který určuje výstup sítě.
- Každý perceptron nejprve vypočítá svůj vnitřní potenciál jako příslušnou afinní kombinaci:

$$\xi = \sum_{i=0}^{n} \omega_{ji} x_i \qquad j = 1,\ldots,m$$

Síť perceptronů III.

- Koeficienty $w = (\omega_{10},\ldots,\omega_{1m},\ldots,\omega_{m0},\ldots,\omega_{mn})$ tvoří konfiguraci sítě.
- Stav perceptronu se potom určí z jeho vnitřního potenciálu aplikací aktivační funkce, která má tvar ostré nelinearity:

$$\sigma : R \to \{0,1\} \quad \sigma(\xi) = \begin{cases} 1 \ldots \xi \geq 0 \\ 0 \ldots \xi < 0 \end{cases}$$

Síť perceptronů IV.

- To znamená, že funkce sítě perceptronů závislá na konfiguraci **w** je daná vztahem:

$$\mathbf{y}(w): R^n \rightarrow \{0,1\}^m;$$

$$\mathbf{y}_j = \sigma(\xi_j) \quad j = 1,..,m \quad \sigma(\xi) = \begin{cases} 1 \ldots \xi \geq 0 \\ 0 \ldots \xi < 0 \end{cases}$$

Síť perceptronů V.

- V **Adaptivní** dynamice je požadovaná funkce sítě perceptronů daná tréningovou množinou:

$$\mathrm{T} = \left\{ (\mathbf{x}_k, \mathbf{d}_k) \;\middle|\; \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in R^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \{0,1\}^m \end{array} \quad k = 1, \ldots, p \right\}$$

- Kde $\mathbf{x}_k$ je reálný vstup k-tého tréningového vzoru a $\mathbf{d}_k$ je odpovídající požadovaný binární výstup.
- Cílem adaptace je, aby síť pro každý vstup $\mathbf{x}_k$ z tréningové množiny odpovídala v aktivním režimu požadovaným výstupům $\mathbf{d}_k$, tedy aby platilo:

$$\mathbf{y}(\mathbf{w}, \mathbf{x}_k) = \mathbf{d}_k \qquad k = 1, \ldots, p$$

Síť perceptronů VI.

- Na začátku adaptace v (diskrétním) čase 0 jsou váhy konfigurace nastavené náhodně z intervalu <-1,1>.
- V každém časovém kroku je síti předložen jeden vzor z tréningové množiny a síť se ho snaží naučit, tedy adaptuje podle něj svoje váhy.
- Pořadí vzorů je dané tzv. **tréningovou strategií.**
- Perceptronové učící pravidlo:

$$\omega_{ji}^{(t)} = \omega_{ji}^{(t-1)} - \varepsilon x_{ki}\left(y_j\left(\mathbf{w}^{(t-1)}, \mathbf{x}_k\right) - d_{kj}\right) \quad \Big| \quad \begin{array}{l} i = 1,\ldots,n \\ j = 1,..,m \end{array}$$

Síť perceptronů VII.

- ε z intervalu (0,1> je rychlost učení.
- $y_j(\mathbf{w}^{(t-1)}, \mathbf{x}_k) - d_{kj}$ je rozdíl mezi skutečným j-tým výstupem sítě pro vstup k-tého vzoru a požadovanou hodnotou odpovídajícího výstupu tohoto vzoru.
- Určuje tedy chybu j-tého výstupu sítě pro k-tý tréningový vzor. Pokud je tato chyba nulová, příslušné váhy se neadaptují. V opačném případě může být tato chyba buď 1 nebo -1.
- Tato adaptivní dynamika zajistí, aby síť po konečném počtu kroků adaptivního režimu našla konfiguraci, pro kterou bude správně klasifikovat všechny tréningové vzory.

Vícevrstvá síť a Backpropagation

- Najznámnější a najpoužívanejší model NS, který se používá přibližně v 80% všech aplikací NS.
- Zobecnění sítě perceptronů – tzv. **vícevrstvý perceptron.**
- Algoritmus zpětného šíření chyby – **Backpropagation.**

Organizační a aktivní dynamika

Organizační dynamika:

- obecně se používá dvou- nebo třívrstvá síť
- X – množina *n* vstupních neurónů
- Y – množina *m* výstupních neurónů
- ε_j - reálný vnitřní potenciál neuronu j
- y_j - reálný stav (výstup) neuronu j
- ω_{ji} - reálná synaptická váha spoje od neuronu i k nevstupnímu neuronu j
- $\omega_{j0} = -h_j$ - bias nevstupního neuronu j odpovedající formálnímu jednotkovému vstupu $y_0 = 1$
- $j_{\leftarrow}$ - množina neuronů, které jsou vstupem neuronu j
- $j_{\rightarrow}$ - množina neuronů, kterým je neuron j vstupem

Organizační a aktivní dynamika

Aktivní dynamika:

- Výpočet funkce $\mathbf{y}(w): R^n \to (0,1)^m$ probíhá podle diskrétní aktivní dynamiky.
- V čase 0 jsou odpovídající stavy vstupních neuronů nastavené na vstup sítě a ostatní neurony nemají určený stav.
- V čase t>0 jsou vypočtené reálné hodnoty vnitřních potenciálů všech neuronů, které už mají určený stav (v čase t se aktualizují neurony v t-té vrstvě):

$$\xi_j = \sum_{i \in j_{\leftarrow}} \omega_{ji} y_i$$

- Dále je stanoven reálný stav $\mathbf{y}_j = \sigma(\xi_j)$ neuronu j pomocí diferencovatelné aktivační funkce :

$$\sigma : R \to (0,1) \quad \sigma(\xi) = \frac{1}{1+e^{-\lambda\xi}}$$

Organizační a aktivní dynamika

- Diferencovatelnost použité funkce a z ní plynoucí diferencovatelnost funkce sítě je podstatná pro učící algoritmus backpropagation.
- λ - parametr strmosti (gain)– v základním modelu je rovný 1, ale obecně může být strmost různá pro každý nevstupní neuron j. Stav neuronu se potom počítá:

$$\mathbf{y}_j = \sigma_j(\xi_j), \quad kde \quad \sigma_j(\xi) = \frac{1}{1+e^{-\lambda_j \xi}}$$

- Takto se vypočtou výstupy všech neuronů, hlavně výstupních, které určují výstup sítě a tedy i hodnotu sítě funkce pro daný vstup.

Adaptivní dynamika

- Podobně jako u sítě perceptronů je požadovaná funkce zadaná tréningovou množinou:

$$\mathrm{T} = \left\{ (\mathbf{x}_k, \mathbf{d}_k) \;\middle|\; \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in R^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \{0,1\}^m \end{array} \quad k = 1, \ldots, p \right\}$$

- **Chyba sítě** E(**w**) vzhledem k této tréningové množině je definovaná jako součet parciálních chyb sítě vzhledem k jednotlivým tréningovým vzorům, přičemž závisí na konfiguraci sítě **w**:

$$E(\mathbf{w}) = \sum_{k=1}^{p} E_k(\mathbf{w})$$

kde $$E_k(\mathbf{w}) = \frac{1}{2} \sum_{j \in Y} (y_j(\mathbf{w}, \mathbf{x}_k) - d_{kj})^2$$

- Cílem adaptace je minimalizace chyby sítě ve váhovém prostoru – používá se gradientní metoda vyžadující diferencovatelnost chybové funkce.

Adaptivní dynamika

- V čase 0 jsou váhy konfigurace nastavené náhodně, blízko nuly.
- Adaptace probíhá v diskrétních časových krocích, které odpovídají tréningovým cyklům.
- Nová konfigurácia $\mathbf{w}^{(t)}$ v čase t>o se vypočítá:

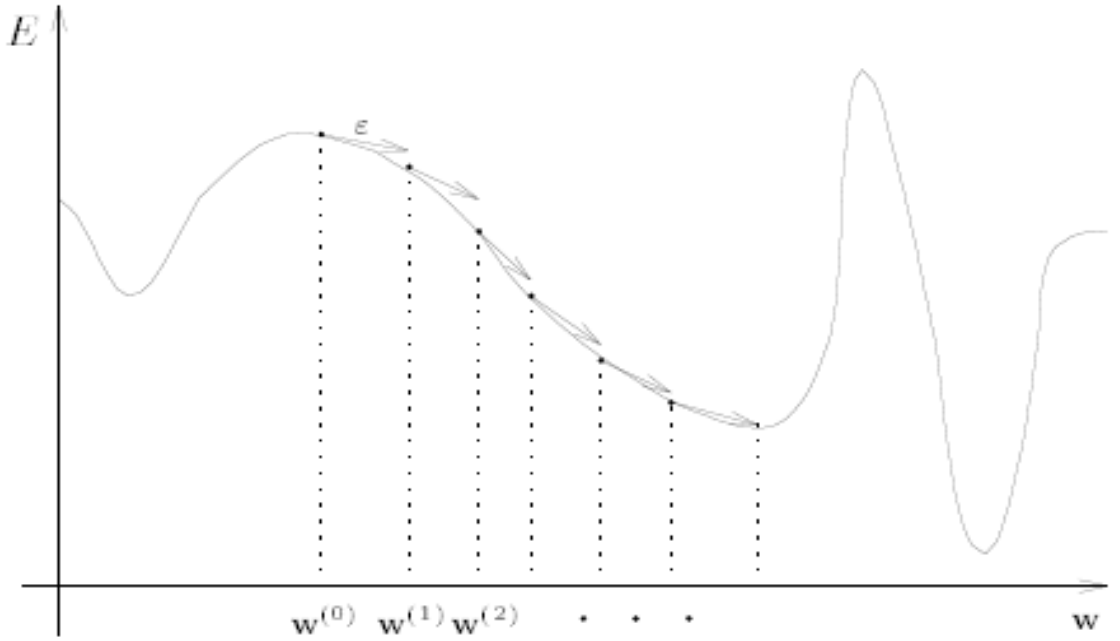
$$\omega_{ji}^{(t)} = \omega_{ji}^{(t-1)} + \Delta\omega_{ji}^{(t)}$$

- Kde změna vah v čase t je úměrná zápornému gradientu chybové funkce v čase t-1:

$$\Delta\omega_{ji}^{(t)} = -\varepsilon \frac{\partial E}{\partial \omega_{ji}}(\mathbf{w}^{(t-1)})$$

- ε z (0,1) je rychlost učení

Adaptivní dynamika



Obr. 2.2: Gradientní metoda.

Při adaptaci sestrojíme v bodě současné konfigurace tečný vektor – gradient a posuneme se ve směru tohoto vektoru.

Strategie zpětného šíření

- Potřebujeme vypočítat gradient chybové funkce.
- Podle pravidla o deriváci součtu:

$$\frac{\partial E}{\partial \omega_{ji}} = \sum_{k=1}^{p} \frac{\partial E_k}{\partial \omega_{ji}}$$

$$\frac{\partial E_k}{\partial \omega_{ji}} = \frac{\partial E_k}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial \xi_j}{\partial \omega_{ji}}$$

kde $\frac{\partial \xi_j}{\partial \omega_{ji}} = y_i$

a $\frac{\partial y_j}{\partial \xi_j} = \frac{\lambda_j e^{-\lambda_j \xi_j}}{(1+e^{-\lambda_j \xi_j})^2} = \frac{\lambda_j}{1+e^{-\lambda_j \xi_j}} \left(1 - \frac{1}{1+e^{-\lambda_j \xi_j}}\right) = \lambda_j y_j (1-y_j)$

- Po dosazení

$$\frac{\partial E_k}{\partial \omega_{ji}} = \frac{\partial E_k}{\partial y_j} \lambda_j y_j (1-y_j) y_i$$

Strategie zpětného šíření

- Pro výpočet $\frac{\partial E_k}{\partial y_j}$ se používá strategie zpětného šíření:

1. Je-li j je z Y (výstupní neuron): $\frac{\partial E_k}{\partial y_j} = y_j - d_{kj}$
   což odpovídá chybě výstupního neuronu j pro k-tý tréningový vzor.
2. Pro skrytý neuron uplatníme pravidlo o derivování složené funkce:

$$\frac{\partial E_k}{\partial y_j} = \sum_{r \in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r} \frac{\partial y_r}{\partial \xi_r} \frac{\partial \xi_r}{\partial \omega_{ji}} = \sum_{r \in j^{\rightarrow}} \frac{\partial E_k}{\partial y_r} \lambda_r y_r (1 - y_r) \omega_{rj} \qquad j \notin X \cup Y$$

- Tedy výpočet derivace pro skrytý neuron j jsme převedli na výpočet parciálních derivací u neuronů r, do kterých vede vstup z neuronu j.

MADALINE I.

- **M**ultiple **ADALINE**
- Základním prvkem je neuron ADALINE, který je velmi podobný perceptronu.
- **Organizační** dynamika je totožná jako u sítě perceptronů, ale namísto perceptronu je použitý ADALINE.

MADALINE II.

- **Aktivní** dynamika se liší tím, že výstupy sítě můžou být obecně reálné a jednotlivé ADELINE realizují lineární funkci (chybí nelineární aktivační funkce).

$$\mathbf{y}(w) : R^n \rightarrow R^m \qquad y_j = \sum_{i=1}^{n} \omega_{ji} x_i \qquad j = 1,\ldots,m$$

MADALINE III.

- V **Adaptivním** režimu je požadovaná funkce MADALINE zadaná tréningovou posloupností, kde reálné vstupy tréningových vzorov $\mathbf{x}_k$ jsou generované náhodně s daným rozdělením pravděpodobnosti a u každého je daný požadovaný výstup $\mathbf{d}_k$:

$$\mathrm{T} = \left\{ \left(\mathbf{x}_k, \mathbf{d}_k \right) \middle| \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in R^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in R^m \end{array} \quad k = 1,2,.. \right\}$$

MADALINE IV.

- Chyba j-tého ADALINE vzhledom k tréningové posloupnosti v závislosti na části konfigurace $\mathbf{w}_j$ je definovaná:

$$E_j(\mathbf{w}_j) = \lim_{p\to\infty} \frac{\frac{1}{2}\sum_{k=1}^{p}(y_j(\mathbf{w}_j,\mathbf{x}_k) - d_{kj})^2}{p} = \mathbf{E}\left[\frac{1}{2}(y_j(\mathbf{w}_j,\mathbf{x}_k) - d_{kj})^2\right] \quad j = 1,..,m$$

- Je to tedy (podle zákona velkých čísel) střední hodnota poloviny mocniny rozdílu skutečného stavu j-tého ADELINE a odpovedajíceho požadovaného výstupu vzhledem k tréningové posloupnosti.

MADALINE V.

- Cílem adaptace je minimalizace chyby $E_j(\mathbf{w}_j)$.
- Vypočítáme gradient této chybové funkce záměnou limity a derivace a s využitím pravidla o derivaci složené funkce:

$$\frac{\partial E_j}{\partial \omega_{ji}} = \lim_{p\to\infty} \frac{1}{p} \sum_{k=1}^{p} x_{ki}\left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right) \quad i = 0,\ldots,n$$

- Vyjádříme jako střední hodnotu:

$$\frac{\partial E_j}{\partial \omega_{ji}} = E\left[x_{ki}\left(y_j(\mathbf{w}_j, \mathbf{x}_k) - d_{kj}\right)\right] \quad i = 0,\ldots,n$$

- Dosadíme za funkci y_j:

$$\frac{\partial E_j}{\partial \omega_{ji}} = -E\left[x_{ki} d_{kj}\right] + \sum_{r=0}^{n} \omega_{jr} E\left[x_{kr} x_{ki}\right] \quad i = 0,\ldots,n$$

MADALINE VI.

- 2 možné postupy minimalizace chybové funkce:

1. Položíme parciální derivace rovny 0: $\frac{\partial E_j}{\partial \omega_{ji}} = 0$

   Odhadnem stredné hodnoty: $E\left[x_{ki} d_{kj}\right]; E\left[x_{kr} x_{ki}\right]$

   Dostanem sústavu:

$$\sum_{r=0}^{n} \omega_{jr} E\left[x_{kr} x_{ki}\right] = E\left[x_{ki} d_{kj}\right] \quad i = 0,\ldots,n$$

   Řešením této soustavy je konfigurace $\mathbf{w}_j^*$ pro j-tý ADALINE, která minimalizuje chybovou funkci.

MADALINE VII.

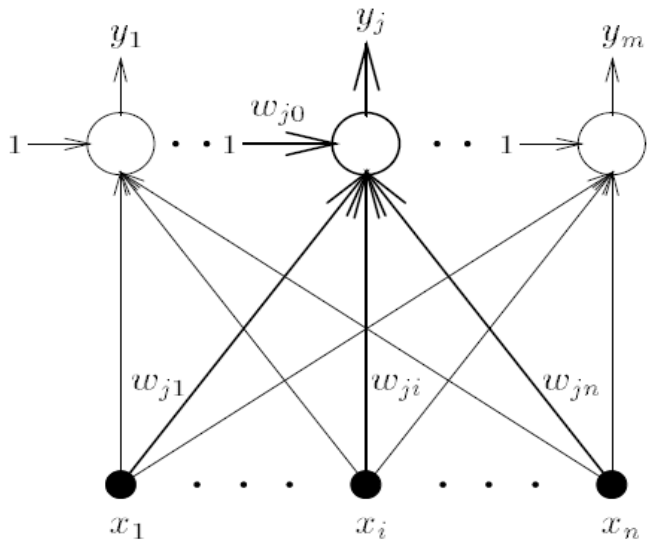
- 2 možné postupy minimalizace chybové funkce:

2. Použití gradientní metody s využitím (Widrow-Hoff) pravidla **LMS** (**L**ast-**M**ean-**S**quare), podle kterého je změna konfigurace v čase t daná:

$$\omega_{ji}^{(t)} = \omega_{ji}^{(t-1)} - \varepsilon x_{ki}\left(y_j\left(\mathbf{w}_j^{(t-1)}, \mathbf{x}_k\right) - d_{kj}\right) \qquad \begin{matrix} j = 1,\ldots,m \\ i = 0,\ldots,n \end{matrix}$$

Tento adaptivní proces konverguje z libovolné počáteční konfigurace $\mathbf{w}^{(0)}$ ke konfiguraci $\mathbf{w}^*$, která minimalizuje chybové funkce $E_j\left(\mathbf{w}_j\right) \quad j = 1,\ldots,m$.

2. Asociativní neuronové sítě, Hebbův zákon, Kohonenovy mapy, LVQ

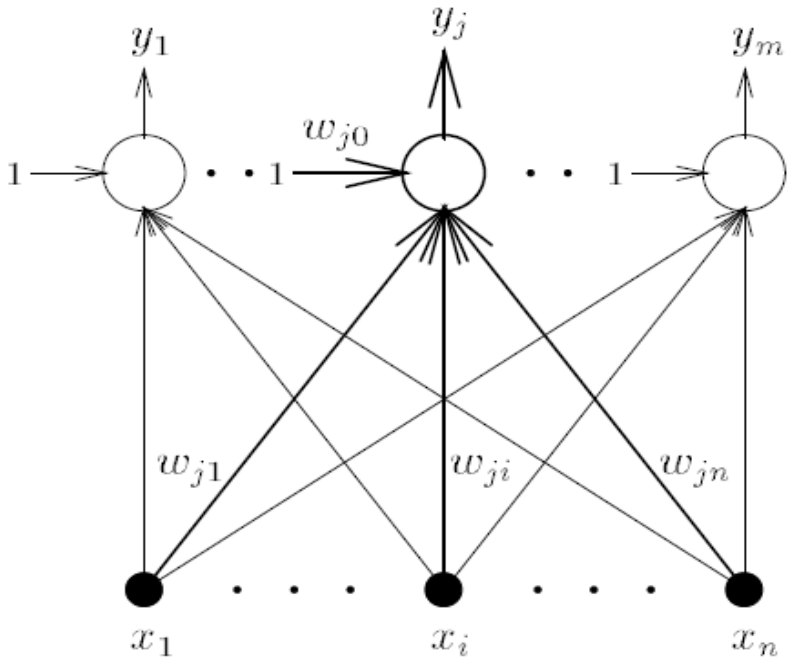


Lineární asociativní síť

- Model neuronové sítě, při kterém sa využívá asociativní paměť.
- Rozdíl proti klasickým počítačům – na vyhledání položky neslouží adresa v paměti, ale částečná znalost informace.
- Příklad: č-b foto připomene barvu vlasů, očí, jméno.
- 2 typy asociativní paměti:
  - Autoasociativní – zpřesnění vstupní informace (vybavení si barevného obrazu).
  - Heteroasociativní – vybavení si združené informace (vybavení si jména).

Lineární asociativní síť

- Organizační dynamika:
  - Skládá se z n vstupních neuronů, kde každý je vstupem každého z m výstupních neuronů.



Lineární asociativní síť

- Aktívní dynamika:
  - Určuje spůsob výpočtu funkce sítě.
  - Počítá se jako lineární kombinace vstupů.
  - Formálně se její funkce $\mathbf{y}(\mathbf{w}) : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ zapisuje: $y_j = \sum_{i=1}^{n} w_{ji} x_i \quad j = 1, \ldots, m\,.$
  - Vyjádření maticovým zápisem: $\mathbf{x} = (x_1, \ldots, x_n)$ $\mathbf{y} = (y_1, \ldots, y_m)$ vstupy/výstupy jsou sloupcové vektory konfigurace sítě je daná váhovou maticí **W** typu m × n, jejíž řádky odpovídají synaptickým váhám vstupů.
  - Maticový součin:

$$\mathbf{y} = \mathbf{W}\mathbf{x} \qquad \mathbf{W} = \begin{pmatrix} w_{11} & \ldots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \ldots & w_{mn} \end{pmatrix}$$

Lineární asociativní síť

- Adaptivní dynamika:
  - V adaptivním režimu je požadovaná funkce zadaná tréningovou množinou

$$\mathcal{T} = \left\{ (\mathbf{x}_k, \mathbf{d}_k) \;\middle|\; \begin{array}{l} \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \mathbb{R}^n \\ \mathbf{d}_k = (d_{k1}, \ldots, d_{km}) \in \mathbb{R}^m \end{array} \quad k = 1, \ldots, p \right\}$$

  - Při autoasociativní paměti – výstup odpovídá vstupu ($m=n$ a $x_k=d_k$)
  - 2 možnosti adaptace:
    - Adaptace podle Hebbova zákona
    - Pseudohebbovská adaptace

Lineární asociativní síť

- Adaptace podle Hebbova zákona:
  - Vysvětlená adaptivní dynamika pro případ heteroasociativní paměti.
  - Tvrdí, že změna synaptické váhy spoje mezi dvěma neurony je úměrná jejich souhlasné aktivitě, tedy součinu jejich stavů.
  - Na začátku adaptace (t=0) jsou všechny váhy konfigurace nulové, tedy $w_{ji}^{(0)} = 0 \ (j = 1, \ldots, m, \ i = 1, \ldots, n)$.
  - V čase t=1,…,p je síti předložený k-tý tréningový vzor, váhy se adaptují: $w_{ji}^{(t)} = w_{ji}^{(t-1)} + d_{kj} x_{ki} \qquad \begin{matrix} j = 1, \ldots, m \\ i = 1, \ldots, n \, . \end{matrix}$
  - Adaptace končí po p krocích – všechny tréningové vzory jsou naučené.
  - Výsledná konfigurace: $w_{ji} = \sum_{k=1}^{p} d_{kj} x_{ki} \qquad \begin{matrix} j = 1, \ldots, m \\ i = 1, \ldots, n \, . \end{matrix}$

Lineární asociativní síť

- Adaptace podle Hebbova zákona:
  - Vyjádření pomocí matic:

$$\mathbf{W}^{(0)} = \mathbf{0}, \quad \mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} + \mathbf{d}_k \mathbf{x}_k^{\mathsf{T}}, \quad k = 1, \ldots, p\,,$$

    kde T je transpozice matice, **o** je nulová matice a váhová matice $\mathbf{W}^{(k)}$ určuje konfiguraci sítě v čase t=k.
  - Výsledná konfigurace: $\mathbf{W} = \mathbf{W}^{(p)} = \sum_{k=1}^{p} \mathbf{d}_k \mathbf{x}_k^{\mathsf{T}} = \mathbf{D}\mathbf{X}^{\mathsf{T}}$

    kde sloupce matic **X**, resp. **D** jsou vstupy $\mathbf{x}_k$, resp. požadované výstupy $\mathbf{d}_k$ tréningových vzorů.
  - V případě autoasociativní paměti (**X**=**D**) $\mathbf{W} = \mathbf{X}\mathbf{X}^{\mathsf{T}}$

Lineární asociativní síť

- Adaptace podle Hebbova zákona:
  - Předpokládáme, že množina vstupních vektorů je ortonormální – vzájemně kolmé jednotkové vektory (vstupy se tedy dostatečně liší a jsou porovnatelné).
  - Síť má schopnost reprodukce – ze vstupu $\mathbf{x}_r$ dostaneme příslušný výstup $\mathbf{d}_r$.

$$\mathbf{y}(\mathbf{x}_r) = \mathbf{W}\mathbf{x}_r = \left(\sum_{k=1}^{p} \mathbf{d}_k \mathbf{x}_k^{\mathrm{T}}\right) \mathbf{x}_r = \sum_{k=1}^{p} \mathbf{d}_k \left(\mathbf{x}_k^{\mathrm{T}} \mathbf{x}_r\right) = \mathbf{d}_r$$

  - Síť by pro vstup $\mathbf{x}_r+\delta$, který je blízko $\mathbf{x}_r$ měla dát požadovaný výstup $\mathbf{d}_r$.
  - Odpovídající chyba je norma rozdílu skutečného výstupu pro vstup $\mathbf{x}_r+\delta$ a požadovaného výstupu $\mathbf{d}r$

$$E_r(\boldsymbol{\delta}) = \left\|\mathbf{y}(\mathbf{x}_r + \boldsymbol{\delta}) - \mathbf{d}_r\right\| = \left\|\mathbf{W}\mathbf{x}_r + \mathbf{W}\boldsymbol{\delta} - \mathbf{d}_r\right\| = \left\|\mathbf{W}\boldsymbol{\delta}\right\|$$

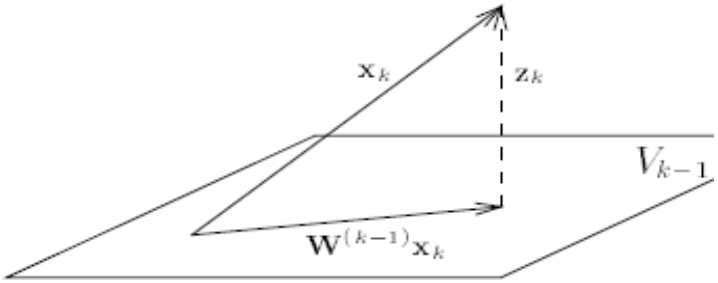
Lineární asociativní síť

- Pseudohebbovská adaptace:
  - Zeslabuje předpoklad reprodukce na ortonormalitu vstupů tréningových vzorů.
  - Předpokládejme LN množinu tréningových vzorů $\{\mathbf{x}_1,\ldots,\mathbf{x}_p\}$ – tvoří bázi vekt. prostoru V_p .
  - Vytvoříme z nich ortogonální bázi $\{\mathbf{z}_1,\ldots,\mathbf{z}_p\}$ V_p .
  - V čase t=0 je $\mathbf{W}^{(0)} = \mathbf{0}$.
  - Po předložení k-tého tréningového vzoru určíme

$$\mathbf{z}_k = \mathbf{x}_k - \mathbf{W}^{(k-1)}\mathbf{x}_k \quad , \mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} + \frac{\mathbf{z}_k \mathbf{z}_k^{\mathsf{T}}}{\mathbf{z}_k^{\mathsf{T}} \mathbf{z}_k} \quad .$$

  - Výsledná váhová matice bude $\mathbf{W} = \mathbf{W}^{(p)} = \sum_{k=1}^{p} \frac{\mathbf{z}_k \mathbf{z}_k^{\mathsf{T}}}{\mathbf{z}_k^{\mathsf{T}} \mathbf{z}_k} = \mathbf{X}\mathbf{X}^{+}$ kde $\mathbf{X}^{+} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}$.
  - $\mathbf{X}^{+}$ je pseudoinverzní matice k matici $\mathbf{X}$.

Lineární asociativní síť



- Pseudohebbovská adaptace – geometrický význam:
  - $\mathbf{W}^{(k-1)}\mathbf{x}_k$ – ortogonální projekce $\mathbf{x}_k$ do V_{k-1}, kt. je určený bází $\{\mathbf{x}_1,\ldots,\mathbf{x}_p\}$, resp. $\{\mathbf{z}_1,\ldots,\mathbf{z}_p\}$.
  - Chceme ověřit, že $\mathbf{z}_k$ je kolmý na všechny $\mathbf{z}_r$, $\mathbf{z}_k\mathbf{z}_r$=0.
  - Dosadíme za $(\mathbf{W}^{(k-1)})^{\mathsf{T}} = \sum_{s=1}^{k-1} \frac{\mathbf{z}_s \mathbf{z}_s^{\mathsf{T}}}{\mathbf{z}_s^{\mathsf{T}} \mathbf{z}_s}$.
  - Dostaneme $\mathbf{z}_k^{\mathsf{T}}\mathbf{z}_r = \mathbf{x}_k^{\mathsf{T}}\mathbf{z}_r - \sum_{s=1}^{k-1} \frac{\mathbf{x}_k^{\mathsf{T}}\mathbf{z}_s \mathbf{z}_s^{\mathsf{T}}\mathbf{z}_r}{\mathbf{z}_s^{\mathsf{T}}\mathbf{z}_s} = \mathbf{x}_k^{\mathsf{T}}\mathbf{z}_r - \frac{\mathbf{x}_k^{\mathsf{T}}\mathbf{z}_r \mathbf{z}_r^{\mathsf{T}}\mathbf{z}_r}{\mathbf{z}_r^{\mathsf{T}}\mathbf{z}_r} = 0$.
  - Pokud $\mathbf{x}$ leží ve V_p, pak splývá se svojí ortogonální projekcí $\mathbf{Wx}=\mathbf{x}$, speciálně pro vstupní bázické vektory $\{\mathbf{x}_1,\ldots,\mathbf{x}_p\}$ dostaneme
$$\mathbf{y}(\mathbf{x}_r) = \mathbf{W}\mathbf{x}_r = \mathbf{x}_r$$
  - Tedy lineární autoasociativní síť vzniknuvší pseudohebbovskou adaptací má schopnost reprodukce.

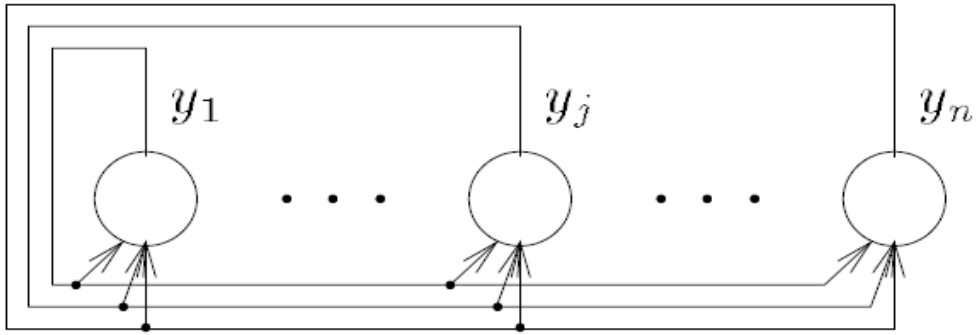
Lineární asociativní síť

- Pseudohebbovská adaptace – zobecnění pro heteroasociativní paměť:
  - Rekurzivní zápis výpočtu váhové matice v případě heteroasociativní paměti určuje Grevilleova věta: $$\mathbf{W}^{(k)} = \mathbf{W}^{(k-1)} + \frac{\left(\mathbf{d}_k - \mathbf{W}^{(k-1)}\mathbf{x}_k\right)\mathbf{z}_k^{\mathrm{T}}}{\mathbf{z}_k^{\mathrm{T}}\mathbf{z}_k}$$

    kde $\mathbf{z}_k$ je stejný sloupcový vektor jako v případě autoasociativní paměti.
  - Pomocí pseudoinverze dostaneme $\mathbf{z}_k = \mathbf{x}_k - \mathbf{X}^{(k-1)}\left(\mathbf{X}^{(k-1)}\right)^{+}\mathbf{x}_k$, kde $\mathbf{X}^{(k-1)}$ je matice $n\times(k-1)$ – sloupce jsou vstupní vektory prvních k-1 tréningových vzorů.
  - Pseudohebbovská adaptivní dynamika zaručuje schopnost heteroasociativní sítě reprodukovat tréningové vzory: $$\mathbf{y}(\mathbf{x}_r) = \mathbf{W}\mathbf{x}_r = \mathbf{D}\mathbf{X}^{+}[\mathbf{X}]_r = \mathbf{D}[(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}(\mathbf{X}^{\mathrm{T}}\mathbf{X})]_r = \mathbf{d}_r$$ kde $[\mathbf{X}]_r$ je r-tý sloupec matice $\mathbf{X}$.

Hopfieldova síť

- Používá se jako autoasociativní paměť.
- Organizační dynamika:
  - Cyklická síť s *n* neurony.
  - Každý je spojený s každým.
  - Všechny neurony jsou vstupní a zároveň výstupní.
  - Dva opačně orientované spoje se dají chápat jako jeden neorientovaný.



Hopfieldova síť

- Adaptivní dynamika:
  - Řídí se hebbovým zákonem.
  - Funkce sítě je specifikovaná tréningovou množinou:

$$\mathcal{T} = \{\mathbf{x}_k \mid \mathbf{x}_k = (x_{k1}, \ldots, x_{kn}) \in \{-1, 1\}^n,\ k = 1, \ldots, p\}$$

  - Tréningové vzory nejsou uložené přímo, ale jsou reprezentované pomocí vztahů mezi stavy neuronů.
  - Probíhá v p diskrétních krocích, kde jsou předkládány tréningové vzory, podle kterých se adaptují synaptické váhy a výsledná konfigurace se zapisuje:

$$w_{ji} = \sum_{k=1}^{p} x_{kj} x_{ki} \quad 1 \leq j \neq i \leq n$$

Hopfieldova síť

- Aktivní dynamika – pro případ sekvenčního synchronního výpočtu:
  - V čase 0 jsou stavy nastavené na vstup sítě **x**=(x_1,…,x_n), t.j.

$$y_i^{(0)} = x_i \ (i = 1, \ldots, n)$$

  - V čase t>0 je aktualizovaný neuron j, vybraný např. systematicky: t=τn+j, kde τ je makroskopický čas – počet period, v kterých jsou aktualizované všechny neurony.
  - Celočíselný potenciál neuronu j: $\xi_j^{(t-1)} = \sum_{i=1}^{n} w_{ji} y_i^{(t-1)}$
  - Znamínko určuje nový bipolární stav:

$$y_j^{(t)} = \begin{cases} 1 & \xi_j^{(t-1)} > 0 \\ y_j^{(t-1)} & \xi_j^{(t-1)} = 0 \\ -1 & \xi_j^{(t-1)} < 0 \end{cases}$$

  - Výpočet končí v čase t*, kdy se síť nachází v tzv. stabilním stavu.
  - Stavy výstupních neuronů určují výstup sítě **y**=(y_1,…,y_n), kde

$$y_j = y_j^{(t^*)} \ (j = 1, \ldots, n)$$

Samoorganizace

- Modely neuronových sítí, které využívají soutěžní strategii učení.
- Výstupní neurony soutěží, který bude aktivní – na rozdíl od Hebbovských sítí je v určitém čase aktivní jen jeden neuron.
- Nejdůležitější/nejznámější architektura soutěžní strategie je Kohonennova samoorganizační mapa.

Vektorová kvantizace (VQ)

- Úlohou je přiblížit hustotu pravděpodobnosti reálných vstupních vektorů **x** pomocí konečného počtu reprezentantů $\mathbf{w}_i$.
- Jedním ze způsobů nalezení reprezentantů je minimalizovat chybu VQ definovanou jako:

$$E = \int ||\mathbf{x} - \mathbf{w}_c||^2 p(\mathbf{x})\, d\mathbf{x} \quad \text{kde } c = \arg \min_{l=1,\ldots,h} \{||\mathbf{x} - \mathbf{w}_l||\}.$$

- Pokud hustotu neznáme a problém je zadaný konečnou tréningovou množinou vzorů, chybu vypočítáme jako

$$E = \frac{1}{k} \sum_{t=1}^{k} ||\mathbf{x}^{(t)} - \mathbf{w}_c||^2$$

- Index c funkčně závisí na vzorech **x** a reprezentantech **w**.

Vektorová kvantizace (VQ)

- Lloydův algoritmus:
  - Nechť je problém zadaný tréningovou množinou a parametrem h, který určuje počet reprezentantů.
  - Projdeme tréningovou množinu a ke každému vstupu $\mathbf{x}^{(t)}$ určíme příslušné $\mathbf{w}_c$, pro každé $\mathbf{w}_j$ zjistíme

$$T_j = \left\{ \mathbf{x}^{(t)}; j = \arg \min_{l=1,\ldots,h} \{ ||\mathbf{x}^{(t)} - \mathbf{w}_l|| \} \right\}$$

vypočítáme $$\mathbf{t}_j = \frac{1}{|T_j|} \sum_{\mathbf{x}_j \in T_j} \mathbf{x}_j$$

a $\mathbf{w}_j$ nahradíme hodnotou $\mathbf{t}_j$.

Kohonennovo učení

- Lloydův algoritmus je nevýhodný v tom, že ke změnám reprezentantů dochází až po průchodu celou tréningovou množinou.

- Proto byla vyvinutá jeho on-line varianta – jednoduchá samoorganizační síť, jejíž algoritmus se nazývá Kohonennovo učení.

Kohonennovo učení

- Organizační dynamika:
  - Dvojvrstvá síť s úplným propojením jednotek mezi vrstvami.
  - Vstupní vrstva – n neuronů – slouží k distribuci vstupních hodnot **x**.
  - Výstupní vrstva – jednotky, které odhadují hustotu pravděpodobnosti vstupů.
  - Váhy $\mathbf{w}_j$ příslušné dané výstupní jednotce j určují její polohu ve výstupním prostoru.

Kohonennovo učení

- Aktivní dynamika:
  - Vstupy – reálná čísla, výstupy – hodnoty 0, 1, přičemž jen jeden neuron je aktivní.
  - Výstup neuronu v závislosti na jeho vzdálenosti od vstupního vektoru se počítá $$y_j^{(t)} = \begin{cases} 1 & j = \arg\min_{l=1,\ldots,h} \{||\mathbf{x}^{(t)} - \mathbf{w}_l||\} \\ 0 & \text{jinak.} \end{cases}$$
  - Popsaný princip – „vítěz bere vše“ – je to jeden z mechanizmů pro realizaci tzv. laterální inhibice.
  - Každý neuron se snaží oslabit ostatní silou úměrnou jeho potenciálu, který je tím větší, čím je neuron blíže vstupu.
  - Výstupní neuron s největším potenciálem zůstane aktivní.

Kohonennovo učení

- Adaptivní dynamika:
  - Procházíme celou tréningovou množinou.
  - Po předložení tréningového vzoru proběhne mezi jednotkami sítě soutěž.
  - Vítěz změní svoje váhy podle vzorce:

$$w_{ji}^{(t)} = \begin{cases} w_{ji}^{(t-1)} + \theta(x_i^{(t-1)} - w_{ji}^{(t-1)}) & j = \arg\min_l\{||\mathbf{x}^{(t)} - \mathbf{w}_l||\} \\ w_{ji} & \text{jinak.} \end{cases}$$

  - Reálný parametr $0<\theta\leq 1$ určuje míru změny vah, na začátku je těsně pod hodnotou 1 a postupně se zmenšuje.
  - Geometrický význam:
    - Vítězný neuron c posune svůj váhový vektor $\mathbf{w}_c$ o určitou vzdálenost směrem k aktuálnímu vstupu.

Kohonennovy samoorganizační mapy

- Organizační dynamika:
  - Podobná jednoduché samoorganizační síti.
  - Výstupní jednotky jsou navíc uspořádané do nějaké struktury, např. dvojrozměrná mřížka, jednorozměrná řada jednotek,...
  - Struktura určuje, které jednotky v síti navzájem sousedí.
  - Okolí neuronu c velikosti s je množina všech neuronů, jejichž vzdálenost od c je ≤ s

$$N_s(c) = \{j; d(j,c) \leq s\}$$

  - Měření vzdálenosti neuronů je závislé na topologické struktuře vstupních neuronů.

Kohonennovy samoorganizační mapy

- Aktivní dynamika:
  - Stejný způsob práce sítě, jako u předchádzejícího modelu.
  - Princip „vítěz bere vše“, jen jeden aktivní neuron.
  - Vstupy – reálná čísla, výstupy – 0,1.
  - Pokud dáme síti vstupní vektor, jednotky soutěží, kdo mu je nejblíže…tato jednotka má výstupní hodnotu rovnu 1.

Kohonennovy samoorganizační mapy

- Adaptivní dynamika:
  - Bere do úvahy uspořádání neuronů.
  - Upravují se váhy nejen vítězné jednotky, ale i jednotkám v okolí, tedy s vítězným neuronem se posouvají i jeho sousedi v síti.
  - Na začátku bývá okolí velké, na konci zahrnuje jen samotného vítěze.
  - Funkce, která pro neurony z okolí neuronu c dává hodnotu θ, pro ostatní 0.

$$h_c(j) = \begin{cases} \theta & j \in N_s(c) \\ 0 & \text{jinak.} \end{cases}$$

  - Adaptaci vah zapisujeme: $w_{ji}^{(t)} = w_{ji}^{(t-1)} + h_c(j)(x_i^{(t)} - w_{ji}^{(t-1)})$
  - Obecnější definování $h_c(j)$ pomocí Gaussovy funkce, aby přechod mezi nulovými a nenulovými hodnotami byl spojitý

$$h_c(j) = h_0 \exp\left(\frac{-d(j,c)^2}{\sigma^2}\right)$$

  - Parametr h_0 – maximální míra posunu.
  - V každém kroku je třeba projít a změnit všechny váhové vektory v síti.

LVQ (learning vector quantizations)

- Nyní se budeme zaobírat tím, jak se dá Kohonennova síť použít pro řešení problémů klasifikace dat do kategorií.
- 3 algoritmy učící vektorové kvantizace – slouží na doučení sítě.
- Uvažujme data $\{(\mathbf{x}^{(t)}, d^{(t)}); t=1,\ldots,k\}$, kde $\mathbf{x}^{(t)}$ je z R a d(t) je z $\{C_1,\ldots,C_q\}$, každý vstupní vektor $\mathbf{x}^{(t)}$ má přiřazenou jednu z konečného počtu kategorií C_k.
- Učení má 3 fáze:
  - Učení bez učitele, jako v předcházejícím případě.
  - Označení výstupních neuronů kategoriemi.
  - Doučení sítě jedním z algoritmů LVQ.

LVQ (learning vector quantizations)

- Postup učení:
  - Použijeme standardní učící algoritmus Kohonennovy sítě – rozmístíme neurony do vstupného prostoru – musí aproximovat hustotu pravděpodobnosti vzorů.
  - Využijeme výstupy $d^{(t)}$ z tréningové množiny – u každého tréningového vzoru zjistíme, který neuron je mu nejblíže, zapamatujeme si, do které kategorie patřil.
  - Po průchodu tréningovou množinou – každý výstupní neuron má tabulku četností jednotlivých kategorií – reprezentuje neuron.
  - Každému neuronu přiřadíme kategorii, kterou reprezentoval nejčastěji – označíme v_j.
  - Výsledek – rozdělení neuronů do skupin, které odpovídají jednotlivým kategoriím.
  - Použijeme jeden z třech algoritmů – pro doladění vah výstupních neuronů.

LVQ (learning vector quantizations)

- LVQ1:
  - Snaží se posílit správnou klasifikaci posunutím neuronu k danému vstupu, resp. napravit nesprávnou klasifikaci odsunutím neuronu od daného vstupu.
  - Posunutí se týká jen jednoho neuronu – ten, který „zvítězil“.
  - Posunutí se děje o malou část vzdálenosti neuronu od vstupního vzoru.

LVQ (learning vector quantizations)

- LVQ1 – přesnější algoritmus:
  - Předkládáme síti všechny tréningové vzory.
  - Každému vzoru určíme nejbližší neuron: $c = \arg \min_{l=,1\ldots,h} \{||\mathbf{x}^{(t)} - \mathbf{w}_l||\}$
  - Provedeme úpravy vah tohoto neuronu, přičemž ostatní neurony zůstávají beze změny: $$\mathbf{w}_c^{(t)} = \begin{cases} \mathbf{w}_c^{(t-1)} + \alpha(\mathbf{x}^{(t)} - \mathbf{w}_c^{(t-1)}) & d^{(t)} = v_c \\ \mathbf{w}_c^{(t-1)} - \alpha(\mathbf{x}^{(t)} - \mathbf{w}_c^{(t-1)}) & d^{(t)} \neq v_c \end{cases}$$
  - Parametr α by měl mít počáteční hodnotu 0,01 – 0,02 a během cca 1000tis. iterací by měl být roven nule.
  - Hranice vytvořená mezi třídami pomocí LVQ1 je aproximace Bayesovské rozhodovací hranice – určuje, do které třídy bod připadne podle jeho pozice vzhledem k místu, kde se střetávají distribuce vzorů daných dvou tříd.
  - LVQ1 posouvá vzory směrem od rozhodovací hranice, přičemž rozhodovací hranice se nachází uprostřed spojnice mezi dvěma neurony pocházejícími z různých tříd.

LVQ (learning vector quantizations)

- LVQ2:
  - Snaží se upravit předcházející algoritmus tak, aby posouval rozhodovací hranici směrem k Bayesovské hranici.
  - V jednom kroku posune vždy dva neurony.

LVQ (learning vector quantizations)

- LVQ2 - podmínky pro určení 2 neuronů:
  - Nechť máme vzor $\mathbf{x}^{(t)}$, uvažujeme případ, kdy pro 2 neurony $\mathbf{w}_i$, $\mathbf{w}_j$ nejblíže tomuto vzoru platí, že jeden klasifikujeme dobře a druhý špatně, přičemž nepřihlížíme k tomu, který je nejblíže.
  - Vzor $\mathbf{x}^{(t)}$ nesmí ležet příliš blízko ani jednoho neuronu, vzor se má nacházet v okně/okolí nadroviny v středu spojnice $\mathbf{w}_i$, $\mathbf{w}_j$, přesněji vzor padne do okna relativní šířky q, pokud

    platí $$\min\left\{\frac{d_i}{d_j}, \frac{d_j}{d_i}\right\} > s$$

    kde $$s = \frac{1-q}{1+q}, d_i = d(\mathbf{w}_i, \mathbf{x}^{(t)}), d_j = d(\mathbf{w}_j, \mathbf{x}^{(t)})$$

  - Hodnota q je mezi 0,1 a 0,3 – snaha o co nejužší okno (přesné umístění hranice) a dostatečnou šířku (zachycení statisticky významných dat).

LVQ (learning vector quantizations)

- LVQ2 - postup:
  - Předpokládejme např., že $\mathbf{x}^{(t)}$ a $\mathbf{w}_j$ patří do stejné kategorie … provedeme následující změny vah:

$$\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t-1)} - \alpha(\mathbf{x}^{(t)} - \mathbf{w}_i^{(t-1)})$$
$$\mathbf{w}_j^{(t)} = \mathbf{w}_j^{(t-1)} + \alpha(\mathbf{x}^{(t)} - \mathbf{w}_j^{(t-1)})$$

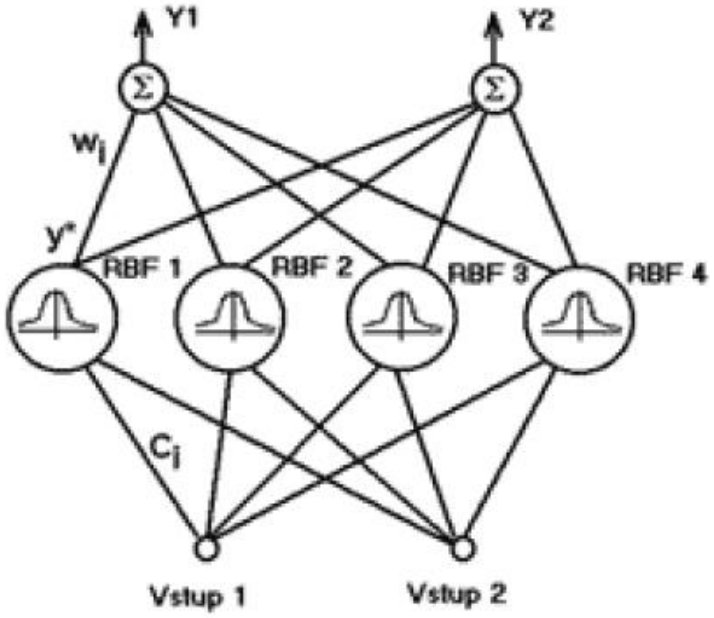
  - Algoritmus nejprve zlepšuje pozice rozhodovací hranice tím, že ji posune směrem k Bayesovské hranici, po jistém počtu kroků se však jednotky od této hranice začínají vzdalovat.
  - LVQ2 se osvědčil pro cca 10000 iterakcí.

LVQ (learning vector quantizations)

- LVQ3:
  - Doplněný o další pravidlo, kterým se zajistí, že správně klasifikující neurony se budou pohybovat směrem k předkládanému tréningovému vzoru.
  - Krok vypadá následovně: $$\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t-1)} - \alpha(\mathbf{x}^{(t)} - \mathbf{w}_i^{(t-1)})$$ $$\mathbf{w}_j^{(t)} = \mathbf{w}_j^{(t-1)} + \alpha(\mathbf{x}^{(t)} - \mathbf{w}_j^{(t-1)})$$

    kde i, j je pár výstupních neuronů, které jsou nejblíže k vzoru $\mathbf{x}^{(t)}$, $v_j=d^{(t)}$, $v_i \neq d^{(t)}$ a $\mathbf{x}^{(t)}$ patří do okna relativní šířky q.
  - Platí: $$\mathbf{w}_r^{(t)} = \mathbf{w}_r^{(t-1)} + \varepsilon\alpha(\mathbf{x}^{(t)} - \mathbf{w}_r^{(t-1)})$$ kde r=i, nebo r=j, a $v_i=v_j=d^{(t)}$.
  - Hodnota ε závisí na šířce okna, měla by být v rozmezí 0,1 – 0,5, je konstantní v čase.
  - Pro parametr α platí: $0< \alpha<1$.

3. RBF sítě, Modulární NN, Hammingova síť

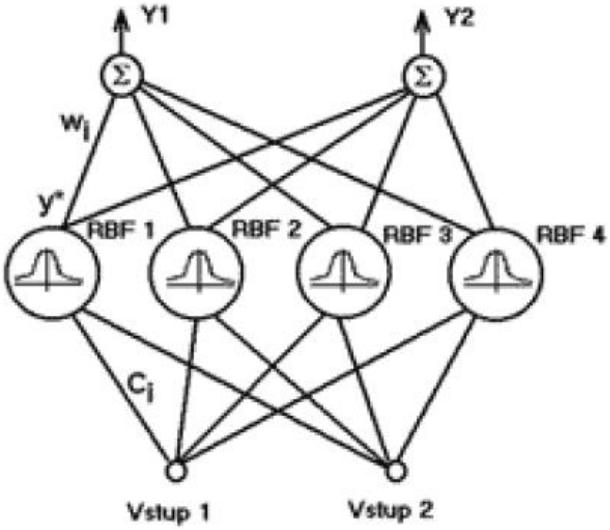


Neuronové sítě typu RBF

- Patří do kategorie sítí s lokálními neurony, což jsou modely *dopředných sítí* obsahující jednu skrytou vrstvu s lokálními jednotkami, které mají výstup lokalizován do okolí bodu určeného svými parametry
- **Radiální bazické funkce**: radiální funkci si lze představit jako funkci určenou středem, která pro argumenty se stejnou vzdáleností od středu dává stejné funkční hodnoty (v dvojrozměrném prostoru se jedná o kružnice)

Neuronové sítě typu RBF

- RBF síť má 3 vrstvy neuronů –vstupní, skrytou a výstupní.
- Vstupní vrstva neuronů má za úkol pouze zprostředkovávat přenos hodnot ze vstupů sítě do neuronů skryté vrstvy.
- Skrytá vrstva je tvořena RBF neurony, které realizují jednotlivé radiální funkce.
- Výstupní vrstvu tvoří perceptronovské neurony.



RBF jednotka (neuron)

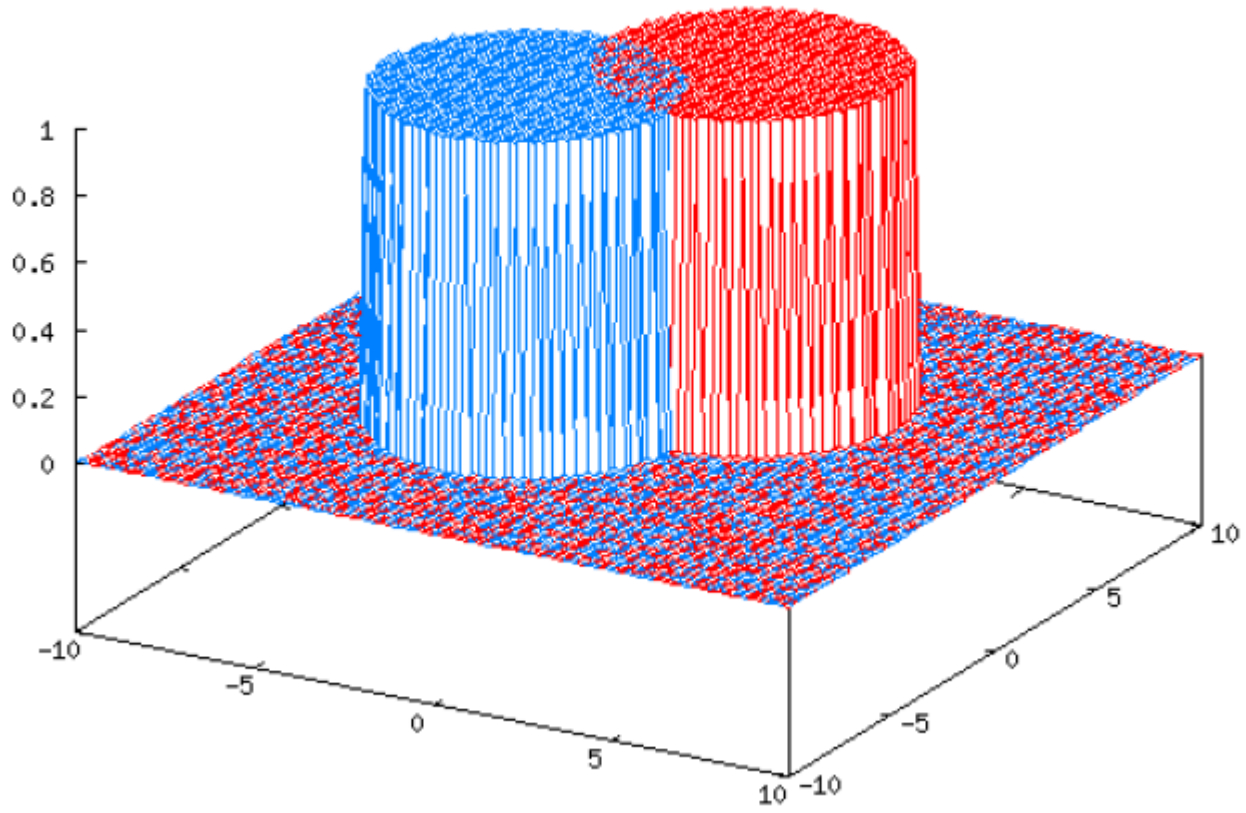
- Má *n* reálných vstupů $x=(x_1,....,x_n)$, z nichž má každý váhu (parametr) c_i
- Má jeden reálný výstup *y*, který odráží stav RBF neuronu, tzn. je-li neuron v klidu (y=0) nebo je-li aktivní (0<y<=1)
- Může mít další parametr *b*, tzv. šířku
- RBF jednotka si pamatuje souřadnice středu $c = (c_1,...,c_n)$ a dále jednu skalární veličinu (šířku *b*)
- Vektoru *c* se také říká prototyp

Funkce RBF jednotky

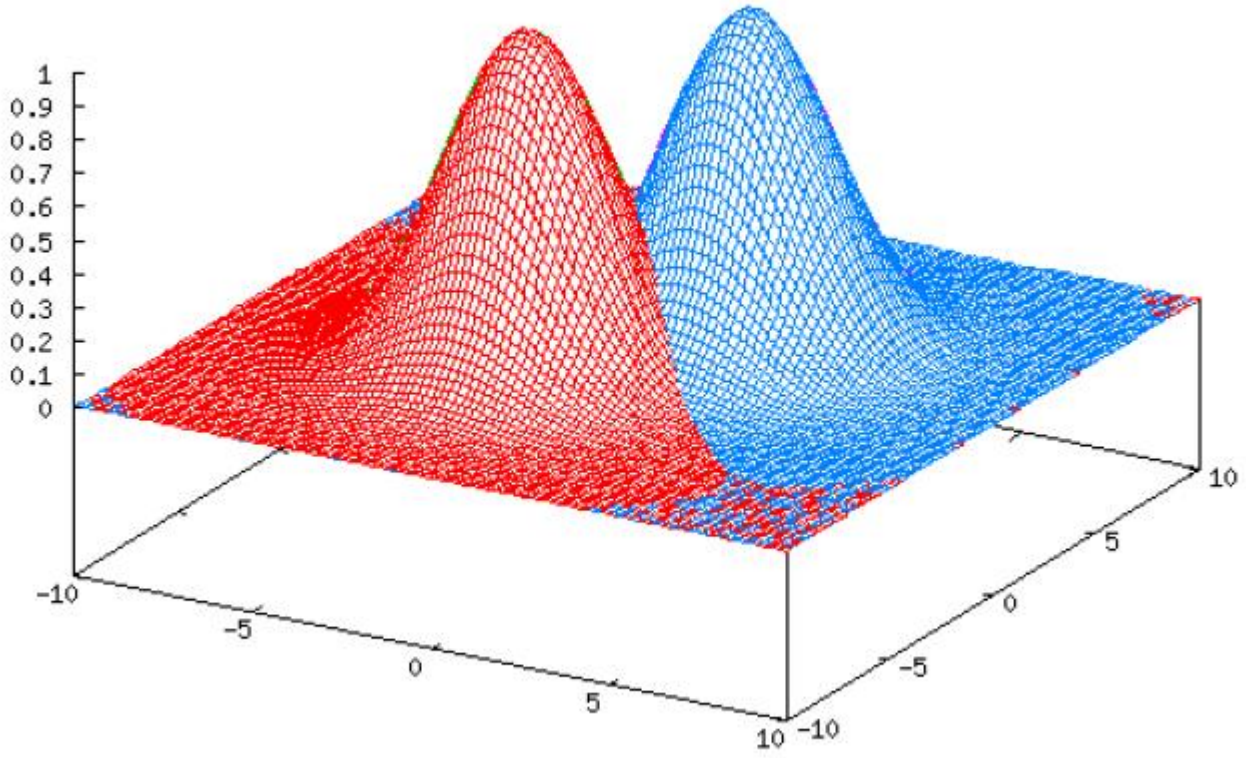
- RBF jednotka vypočte, jaká je (euklidovská) vzdálenost bodu *x* od středu *c* $\|X - C\| = \sqrt{\sum_{i=1}^{N} (X_i - C_{ji})^2}$
- Pokud je tato vzdálenost menší než šířka *b*, tak dojde k aktivaci neuronu a jeho výstupu
- Šířka *b* tedy určuje oblast kolem středu *c*, kde dává jednotka nenulový výstup, pokud do této oblasti spadne nějaká vstupní hodnota
- Průběh výstupní veličiny Y při aktivaci neuronu přitom může mít různý charakter podle pužité výstupní funkce

- Pro RBF neurony se používá euklidovská metrika na rozdíl od perceptronů, kde je metrikou skalární součin
- Jako výstupní funkce RBF neuronu se nejčastěj používá Gaussova funkce (výhodou Gaussovy funkce je, že výstup neuronu se pohybuje spojitě v rozmezí 0 až 1, čímž se vyjadřuje vzálenost vstupního vektoru *x* od středu *c*; jestliže tato vzdálenost je menší než šířka *b*, pak neuron je aktivován a jeho výstup je v rozmezí 0<Y<=1)
- Dále se používají i lineární výstupní funkce (výstupní hodnota je rovna vnitřnímu potenciálu) a diskrétní výstupní funkce

- **Diskrétní funkce**: výstup RBF jednotky nabývá hodnoty 0, pokud vzdálenost vstupního vektoru *x* od středu *c* je větší než šířka *b*, v opačném případě nabývá hodnoty 1
- V tomto případě zde není obsažena informace, jak je vektor *x* vzdálen od středu *c*
- Užití tohoto typu výstupní funkce je vhodné pro klasifikaci


1
0.8
0.6
0.4
0.2
0
-10
-5
0
5
10
-10
-5
0
5
10

- **Gaussova funkce**: pokud vstupní vektor *x* leží v dosahu šířky *b* od středu *c*, má výstup vzhled Gaussova rozdělení: $\varphi(z)=e^{-(z/\beta)^2}$, $\beta \geq 0$
- Pro x=c nabývá výstup hodnoty 1, pro jiné vzdálenosti hodnota výstupu klesá symetricky do všech stran, pro vzdálenost větší než šířka *b* nabývá hodnoty 0
- β určuje strmost Gaussovy funkce


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-10
-5
0
5
10
-10
-5
0
5
10

Přechodová funkce RBF jednotky

$$\xi = \frac{\|\mathbf{x} - \mathbf{c}\|}{b}, \quad y = \varphi(\xi)$$

- První výraz definuje vnitřní potenciál: vnitřní potenciál je vzdálenost vstupního vektoru **x** od středu **c** (příp. dělena šířkou *b*, která se také nazývá sféra vlivu neuronu).
(určuje, zda je vzdálenost vektorů *x* a *c* větší nebo menší než šířka *b*).

- Druhý výraz definuje výstupní (aktivační) funkci: jejím argumentem je vnitřní potenciál a výsledkem výstupní hodnota.

- Každý spoj mezi i-tou vstupní jednotkou a j-tou jednotkou ve skryté vrstvě má váhu c_{ij}, kde j=1,...,h (i-tá souřadnice středu c_j u j-té RBF jednotky)

- Výstup j-té RBF jednotky je spojen s výstupní vrstvou pomocí synapse s vahou w_{js}.

- Výstupní jednotky počítají vážený součet svých vstupů.

- RBF síť provádí dvě transformace: první je nelineární transformace realizována RBF jednotkami, druhá je lineární transformace realizována výstupními neurony sítě a vede z prostoru skrytých jednotek do výstupního prostoru.

Výstupní vrstva sítě

$$y = \sum_{i=1}^{n} w_i y_i^*$$

- Obsahuje neurony perceptronového typu, které váženě sčítají příspěvky od dílčích RBF neuronů.

- Výsledky tohoto součtu jdou na výstupy *Y* sítě.

- Výstupní neuron si pamatuje váhy *w*, kterými násobí své vstupy (vstupy jsou výstupy RBF neuronů *y** propojených s výstupním neuronem).

- Neuronové sítě typu RBF jsou sítě s dopředným šířením signálu, tzn., že jde o přímou topologii sítě s minimálně jednou skrytou vrstvou, jako učící algoritmus využívá zpětné šíření chyby, kdy se po porovnání skutečného a očekávaného výstupu upravují nejdříve váhy v poslední vrstvě, potom v předposlední atd.

Učení neuronových sítí typu RBF

- Učení probíhá na základě trénovací množiny, kterou tvoří páry vektorů sestávající ze vstupů a požadovaných výstupů
- První fáze: učení bez učitele, určení pozice středů RBF jednotek, které jsou reprezentovány vahami mezi vstupní a skrytou vrstvou
- Druhá fáze: nastavení hodnot případných dalších parametrů RBF jednotek, pokud existují (šířky)
- Třetí fáze: učení s učitelem, určují se váhy výstupních neuronů

Využití RBF sítí

- Pro řešení aproximace funkce při znalosti jejího obecného průběhu a konkrétních naměřených hodnot (síť se nejprve naučí správný průběh funkce a po zadání vstupních vektorů s určitou chybou dává na výstupu aproximované funkční hodnoty)
- Pro klasifikaci (vstupní vektory tvoří shluky a určitá množina shluků tvoří jednu kategorii; síť se naučí jaké kategorie náleží jakým vstupním vektorům a poté je schopna třídit i vektory, které jí v učení nebyly předloženy)

Použití RBF sítě pro klasifikaci

- Ve fázi učení se síti předkládají na její vstupy vzory a na její výstupy diskrétní informace o kategorii, do které předložené vzory patří (vzory náležící do stejné kategorie jsou sdruženy do shluků)
- RBF síť si ve fázi učení nastaví středy jednotlivých RBF neuronů, šířky a pak i váhy výstupních neuronů

Použití RBF sítě pro aproximaci

- Ve fázi učení se předkládají síti uspořádané dvojice *argument - funkční hodnota*
- Je využito faktu, že dva argumenty ležící ve vstupním prostoru blízko sebe, budou mít i podobnou funkční hodnotu
- Ve fázi učení se nejprve RBF neurony naučí určovat shluky vzájemně blízkých vzorů a pak se výstupní neurony naučí přiřazovat těmto shlukům správnou funkční hodnotu

Modulární NN

- Modulární neuronové sítě jsou tvořeny různými modely neuronových sítí, které jsou sloučeny do jednoho systému
- Každá síť je uspořádána do modulu tak, aby mohla být volně kombinována s ostatními moduly v rámci tohoto systému
- V rámci modulárních NN dochází ke kombinaci technik a učení různých úkolů současně

- Každá neuronová síť v rámci svého modulu pracuje samostatně na určitém dílčím úkolu
- Výstupy jednotlivých modulů se skládají a vytváří výstup sítě jako celku
- Pokud je větší úkol rozčleněný lze dílčí úkoly řešit efektivněji než kdyby se úkol řešil jako celek
- Mezi jednotlivými moduly slouží tzv. zprostředkovatel, který přijímá jejich výstupy a jednotlivé moduly na sebe navzájem nepůsobí

Biologický základ

- Modulární neuronové sítě, jakožto kombinované struktury, mají také biologické pozadí:
  Přirozené nervové systémy se skládají z hierarchie sítí složených z prvků specializovaných na různé úkoly. Obecně platí, že kombinované sítě jsou silnější než ty nestrukturované.

Učení

- Každá neuronová síť může být přizpůsobena pro svůj úkol → Pro jednotlivé „podsítě“ může být při učení použit jedinečný tréningový algoritmus a tréningové údaje a učení lze provádět mnohem rychleji

- Velké neuronové sítě (ať už biologické nebo umělé) jsou velmi citlivé na chybu v jednom ze svých uzlů
- Při rozčlenění na dílčí úkoly je možné chyby snadněji rozpoznat a jejich vliv v ostatních „podsítích“ je odstraněn, protože jednotlivé sítě jsou navzájem nezávislé

Hammingova NN

- Hammingova síť je typ sítě s učitelem a binárním vstupem
- Je to tzv. třídič dle nejmenší chyby, tzn. Hammingova síť provádí klasifikaci binárních vektorů do tříd na základě chyby, která je definována pomocí Hammingovy vzdálenosti vstupního vektoru od třídy (třídy jsou definovány vzorovými vektory), vstupní vektor je přiřazen do třídy, od níž má nejmenší Hammingovu vzdálenost, tj. počet odlišných vstupů

Příklad

- Máme síť se čtyřmi třídami A, B, C, D a jejich vzorovými vektory:
- A (1 1 1 1 1 1)
- B (1 1 1 -1 -1 -1)
- C (-1 -1 -1 -1 -1 -1)
- D (-1 -1 -1 1 1 1)
- Chceme do některé ze tříd přiřadit vstupní vektor: (1 -1 1 -1 -1 1)

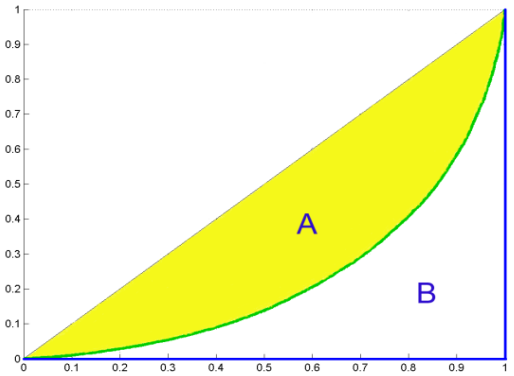
- *Řešení*: budeme zkoumat vzdálenost vstupního vektoru od vzorových vektorů
- Vzhledem k vektoru A má tři rozdílné prvky → jeho Hamminoga vzdálenost je 3 atd.
- Tedy: h(A)=3, h(B)=2, h(C)=3, h(D)=4, kde h je Hammingova vzdálenost
- Vstupní vektor je přiřazen k třídě B, protože má nejmenší Hammingovu vzdálenost od vzorového vektoru

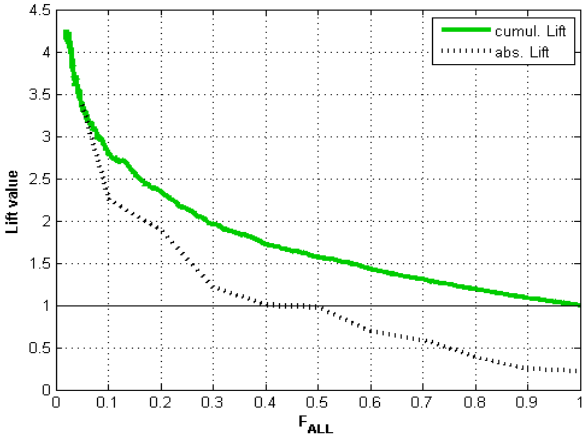
- Hammingova síť má tři vrstvy: vstupní vrstva s *N* elementy, kde *N* je počet bitů vstupního vektoru, vrstva kategorií s *M* elementy, kde *M* je počet kategorií, výstupní vrstva
- Každý prvek (neuron) ve vrstvě kategorií představuje jinou klasifikační třídu (kategorii), která je reprezentována vektorem zakódovaným do vah jeho vstupů

Učení

- Vstupní vzor (vektor) je načten a zpracován ve vstupní vrstvě
- Požadovaný výstup je vektor s jedním prvkem rovným 1 a s ostatními prvky rovnými 0 (resp. -1). Každému vstupnímu vektoru odpovídá jiný výstupní vektor (s jinou pozicí prvku rovného 1)
- Chybové pole u každého neuronu ve vrstvě kategorií je nastaveno na 0
- Požadovaný výstup je zpětným šířením načten do vrstvy kategorií tak, aby bylo chybové pole u všech jejích elementů rovno 0 až na jeden, který bude roven 1, tzn., že požadovaný výstup bude uložen v chybovém vektoru vrstvy kategorií

4. Vyhodnocení kvality prediktivního modelu





Úvod

- Je nemožné využívat predikční modely efektivně bez znalosti jejich kvality/diskriminační síly.

- Většinou je k dispozici celá řada modelů a je třeba vybrat jen jeden – ten nejlepší.

Měření kvality modelu

- Uvažujeme dva základní skupiny indexů kvality. První je založena na distribuční funkci. Mezi nejpoužívanější indexy patří

  - Kolmogorovova-Smirnovova statistika (KS)
  - Giniho index
  - C-statistika
  - Lift.

- Druhá skupina indexů je založena na pravděpodobnostní hustotě. Mezi nejznámější indexy patří

  - Střední diference (Mahalanobisova vzdálenost)
  - Informační statistika/hodnota (I_{Val}).

Indexy založené na distribuční funkci - KS

$$D_K = \begin{cases} 1, & klient \ \ je\ dobrý \\ 0, & jinak. \end{cases}$$

Počet dobrých klientů: n

Počet špatných klientů: m

Proporce dobrých/špatných klientů: $p_G = \frac{n}{n+m}$, $p_B = \frac{m}{n+m}$

- Empirické distribuční funkce:

$$F_{n.GOOD}(a) = \frac{1}{n}\sum_{i=1}^{n} I(s_i \le a \wedge D_K = 1)$$

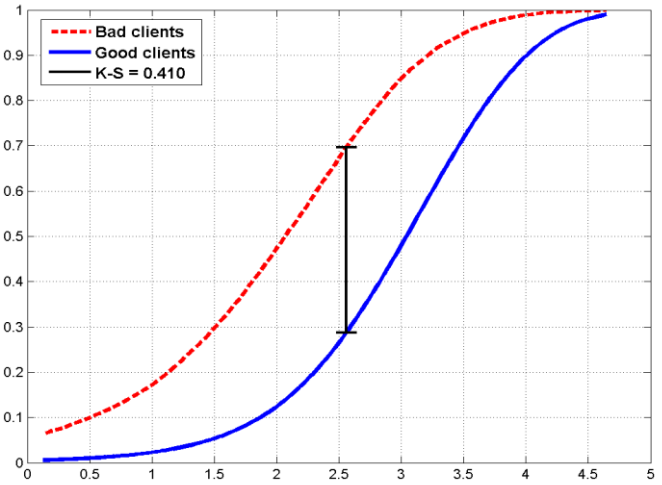
$$F_{m.BAD}(a) = \frac{1}{m}\sum_{i=1}^{m} I(s_i \le a \wedge D_K = 0)$$

$$F_{N.ALL}(a) = \frac{1}{N}\sum_{i=1}^{N} I(s_i \le a) \qquad a \in [L, H]$$

$$I(A) = \begin{cases} 1 & A \ \ platí \\ 0 & jinak \end{cases}$$

- Kolmogorovova-Smirnovova statistika (KS)

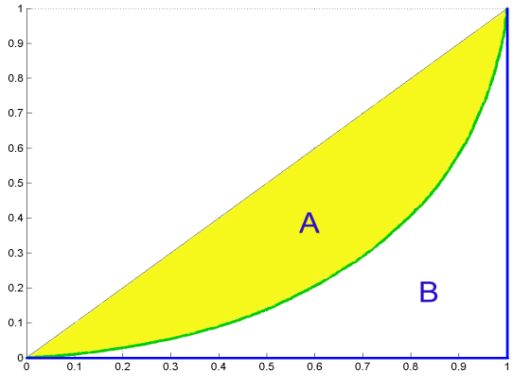
$$KS = \max_{a \in [L,H]} \left| F_{m,BAD}(a) - F_{n,GOOD}(a) \right|$$



Lorenzova křivka

- Lorenzova křivka (LC)

$$x = F_{m.BAD}(a)$$
$$y = F_{n.GOOD}(a),\ a \in [L, H].$$



- Giniho index

$$Gini = \frac{A}{A+B} = 2A$$

$$Gini = 1 - \sum_{k=2}^{n+m} \left(F_{m.BAD\,k} - F_{m.BAD\,k-1}\right) \cdot \left(F_{n.GOOD\,k} + F_{n.GOOD\,k-1}\right)$$

kde $F_{m.BAD\,k}$ ($F_{n.GOOD_k}$) je k-tá hodnota vektoru empirické distribuční funkce špatných (dobrých) klientů

Somersovo D, Kendalovo τ_α

- Giniho index je speciální případ Somersova *D* (Somers (1962)), které je pořadovou asociační mírou definovanou jako

$$D_{YX} = \frac{\tau_{XY}}{\tau_{XX}}$$

kde τ_{XY} je Kendallovo τ_a definované jako $\tau_{XY} = E[sign(X_1 - X_2)sign(Y_1 - Y_2)]$

kde (X_1, Y_1) (X_2, Y_2) jsou bivariantní, stochasticky nezávislé, náhodné vektory nad touž datovou populací, a $E[\cdot]$ značí střední hodnotu. V našem případě je *Y=1* jestliže je klient dobrý a *Y=0* jestliže je klient špatný. Proměnná *X* reprezentuje skóre.

Thomas (2009) uvádí, že Somersovo *D* hodnotící výkonnost daného credit scoringového modelu lze vypočítat pomocí

$$D_S = \frac{\sum_i g_i \sum_{j<i} b_j - \sum_i g_i \sum_{j>i} b_j}{n \cdot m}$$

kde g_i (b_j) je počet dobrých (špatných) klientů v i-tém intervalu skóre.

Somersovo D, Mann-Whitney U

- Dále platí, že D_S může být vyjádřeno pomocí Mann-Whitneyho U-statistiky.
  - Seřaď datový vzorek ve vzestupném pořadí podle skóre a sečti pořadí dobrých klientů ve vzniklé posloupnosti. Označme tento součet jako R_G. Potom

$$U = R_G - \frac{1}{2} n(n+1)$$

$$D_S = 2\frac{U}{n \cdot m} - 1$$

Konkordantní, diskordantní páry

- Konkordantní pár (X_1,Y_1), (X_2,Y_2):

$$\operatorname{sgn}(X_2 - X_1) = \operatorname{sgn}(Y_2 - Y_1)$$

- Diskordantní pár:

$$\operatorname{sgn}(X_2 - X_1) = -\operatorname{sgn}(Y_2 - Y_1)$$

- V našem případě X představuje skóre a Y ukazatel dobrého klienta (D_K). Protože dobrý klient má hodnotu Y=1 a špatný Y=0, je zřejmé, že u konkordantního páru má dobrý klient vyšší hodnotu skóre než klient špatný.

Somersovo D, Goodman-Kruskal gamma

- Uvažujme tedy dva náhodně vybrané klienty, přičemž jeden je dobrý (Y_1=1) a druhý špatný (Y_2=0), skóre prvního označme s_1, druhého s_2. Pak
  - Konkordantní pár (Concordant): $s_1>s_2$
  - Diskordantní pár (Discordant): $s_1<s_2$
  - Vázaný pár (Tied): $s_1=s_2$

- Somersovo D:

$$D_S = \frac{\#Concordant - \#Discodrant}{\#Concordant + \#Discodrant + \#Tied}$$

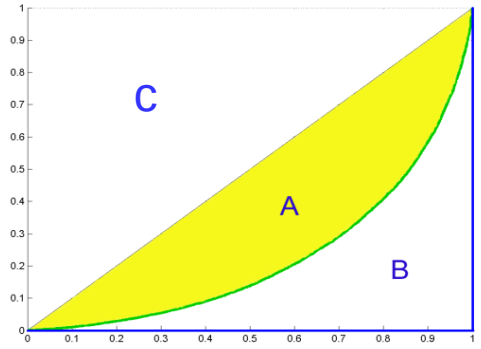
- Goodmanovo-Kruskalovo Gamma:

$$\gamma = \frac{\#Concordant - \#Discodrant}{\#Concordant + \#Discodrant}$$

C-statistika

- C-statistika:

$$c-stat = A + C = \frac{1 + Gini}{2}$$



Tato statistika je rovna pravděpodobnosti, že náhodně vybraný dobrý klient má vyšší skóre než náhodně vybraný špatný klient, tj.

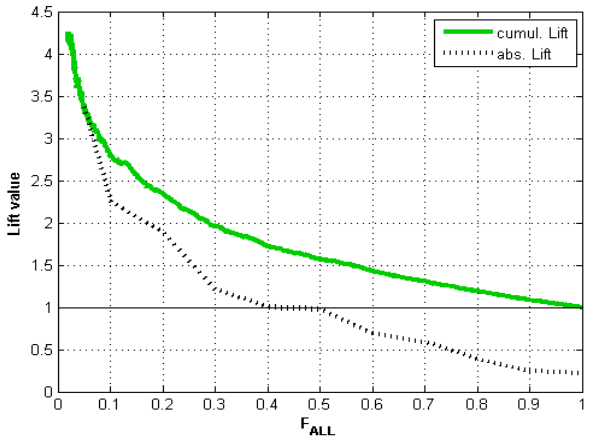
$$c-stat = P\left(s_1 \geq s_2 \;\middle|\; D_{K_1} = 1 \wedge D_{K_2} = 0\right)$$

Lift

- Další možnou mírou kvality scoringového modelu je Lift, který říká kolikrát je daný model, při dané úrovni zamítání, lepší než náhodný model. Přesněji řečeno jde o poměr proporce špatných klientů se skóre menším nebo rovno dané hodnotě skóre a, $a \in [L, H]$, ku proporci špatných klientů v celé populaci. Formálně jej lze zapsat takto:

$$Lift(a) = \frac{CumBadRate(a)}{BadRate} = \frac{\frac{\sum_{i=1}^{n+m} I(s_i \le a \wedge Y = 0)}{\sum_{i=1}^{n+m} I(s_i \le a)}}{\frac{\sum_{i=1}^{n+m} I(Y = 0)}{\sum_{i=1}^{n+m} I(Y = 0 \vee Y = 1)}} = \frac{\frac{\sum_{i=1}^{n+m} I(s_i \le a \wedge Y = 0)}{\sum_{i=1}^{n+m} I(s_i \le a)}}{\frac{n}{N}}$$

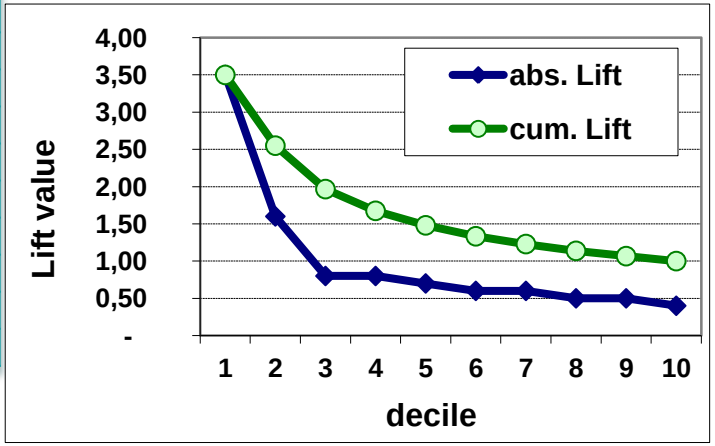
$$absLift(a) = \frac{BadRate(a)}{BadRate}$$



Lift

- Usually it is computed using table with numbers of all and bad clients in some score bands (deciles).

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	35	35.0%	3.50	35	35.0%	3.50
2	100	16	16.0%	1.60	51	25.5%	2.55
3	100	8	8.0%	0.80	59	19.7%	1.97
4	100	8	8.0%	0.80	67	16.8%	1.68
5	100	7	7.0%	0.70	74	14.8%	1.48
6	100	6	6.0%	0.60	80	13.3%	1.33
7	100	6	6.0%	0.60	86	12.3%	1.23
8	100	5	5.0%	0.50	91	11.4%	1.14
9	100	5	5.0%	0.50	96	10.7%	1.07
10	100	4	4.0%	0.40	100	10.0%	1.00
All	1000	100	10.0%				

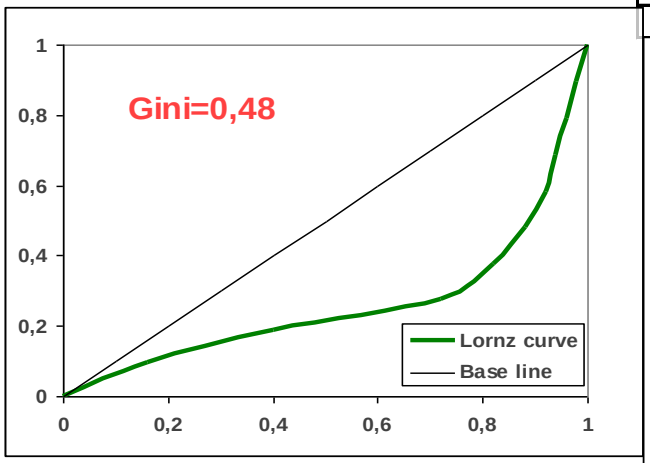


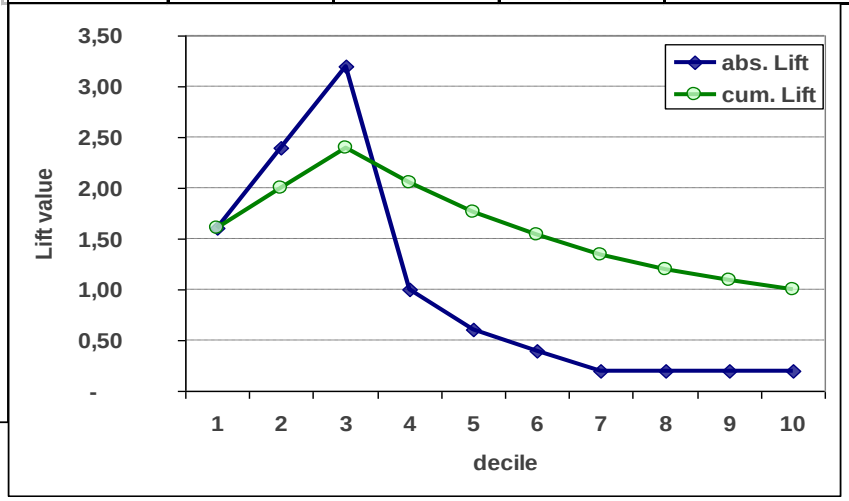
- It takes positive values. Cumulative form ends in value 1.
- Upper limit of Lift depends on p_B .

Lift

- Pokud bad rate není monotonní:
  - LC vypadá OK
  - Gini se mírně sníží
  - Lift ovšem vypadá podivně

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	8	8,0%	1,60	8	8,0%	1,60
2	100	12	12,0%	2,40	20	10,0%	2,00
3	100	16	16,0%	3,20	36	12,0%	2,40
4	100	5	5,0%	1,00	41	10,3%	2,05
5	100	3	3,0%	0,60	44	8,8%	1,76
6	100	2	2,0%	0,40	46	7,7%	1,53
7	100	1	1,0%	0,20	47	6,7%	1,34
8	100	1	1,0%	0,20	48	6,0%	1,20
9	100	1	1,0%	0,20	49	5,4%	1,09
10	100	1	1,0%	0,20	50	5,0%	1,00
All	1000	50	5,0%				



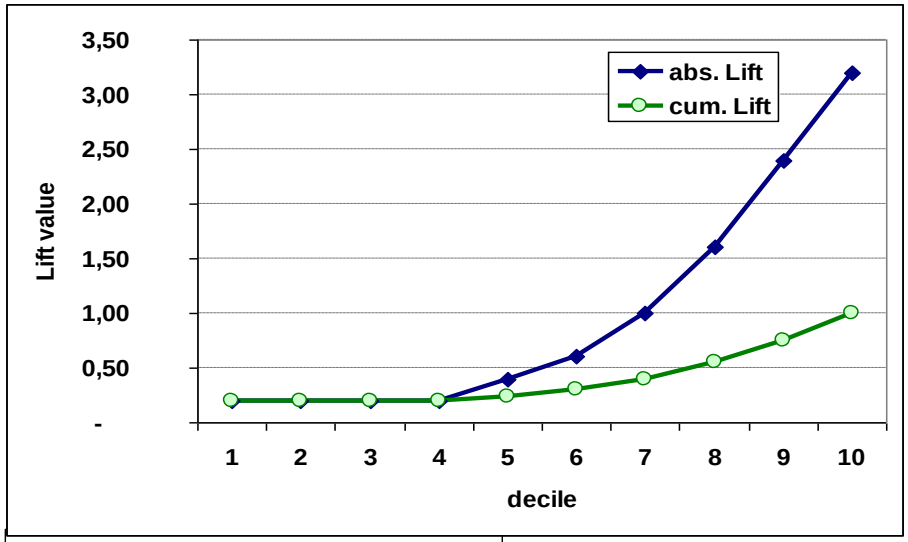


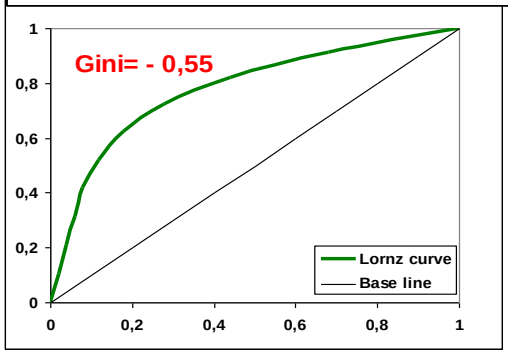
Lift

- Pokud má skóre zcela opačný smysl, obdržíme „opačné“ obrázky.

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	16	16,0%	3,20	16	16,0%	3,20
2	100	12	12,0%	2,40	28	14,0%	2,80
3	100	8	8,0%	1,60	36	12,0%	2,40
4	100	5	5,0%	1,00	41	10,3%	2,05
5	100	3	3,0%	0,60	44	8,8%	1,76
6	100	2	2,0%	0,40	46	7,7%	1,53
7	100	1	1,0%	0,20	47	6,7%	1,34
8	100	1	1,0%	0,20	48	6,0%	1,20
9	100	1	1,0%	0,20	49	5,4%	1,09
10	100	1	1,0%	0,20	50	5,0%	1,00
All	1000	50	5,0%				

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	1	1,0%	0,20	1	1,0%	0,20
2	100	1	1,0%	0,20	2	1,0%	0,20
3	100	1	1,0%	0,20	3	1,0%	0,20
4	100	1	1,0%	0,20	4	1,0%	0,20
5	100	2	2,0%	0,40	6	1,2%	0,24
6	100	3	3,0%	0,60	9	1,5%	0,30
7	100	5	5,0%	1,00	14	2,0%	0,40
8	100	8	8,0%	1,60	22	2,8%	0,55
9	100	12	12,0%	2,40	34	3,8%	0,76
10	100	16	16,0%	3,20	50	5,0%	1,00
All	1000	50	5,0%				



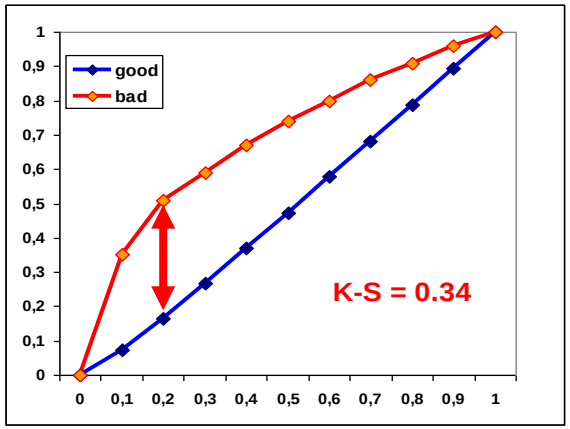


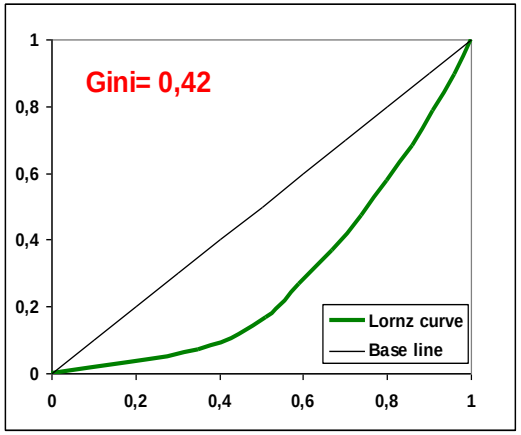
Lift vs. Gini a KS

- **Je evidentní, že pouze Gini nestačí!!!**

SC 1:

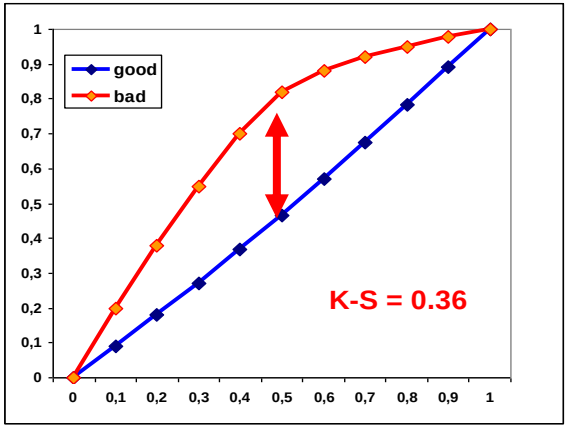
decile	# cleints	# bad clients	Bad rate
1	100	35	35,0%
2	100	16	16,0%
3	100	8	8,0%
4	100	8	8,0%
5	100	7	7,0%
6	100	6	6,0%
7	100	6	6,0%
8	100	5	5,0%
9	100	5	5,0%
10	100	4	4,0%
All	1000	100	10,0%

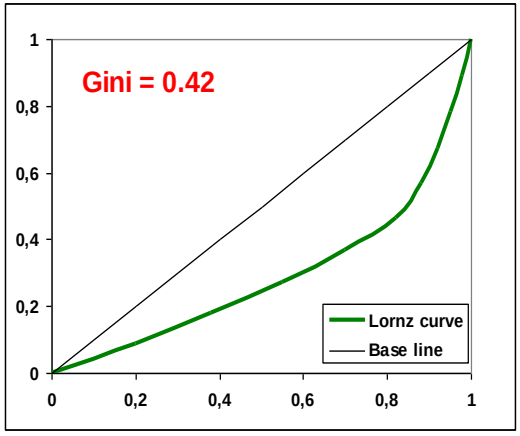




SC 2:

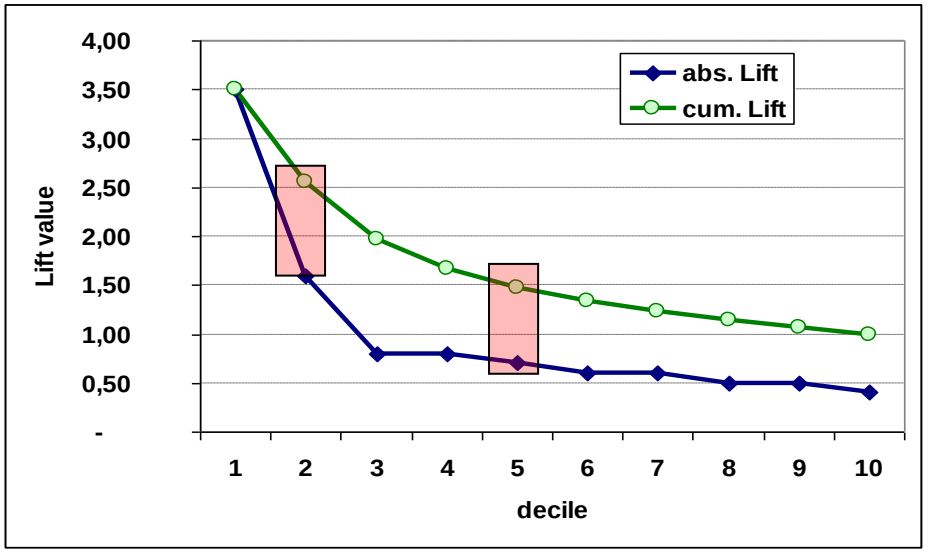
decile	# cleints	# bad clients	Bad rate
1	100	20	20,0%
2	100	18	18,0%
3	100	17	17,0%
4	100	15	15,0%
5	100	12	12,0%
6	100	6	6,0%
7	100	4	4,0%
8	100	3	3,0%
9	100	3	3,0%
10	100	2	2,0%
All	1000	100	10,0%



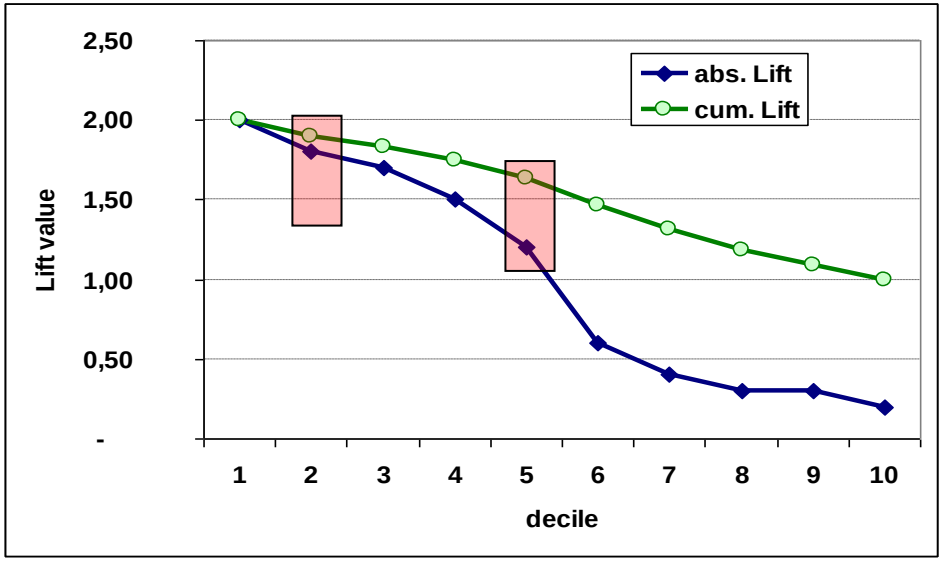


Lift vs. Gini a KS

- **SC 1:**



- **SC 2:**



$Lift_{20\%} = 2.55 \quad > \quad Lift_{20\%} = 1.90$

$Lift_{50\%} = 1.48 \quad < \quad Lift_{50\%} = 1.64$

SC 2 je lepší, pokud je předpokládaná míra zamítaní (reject rate) přibližně 50%.
SC 1 je významně lepší, pokud je předpokládaný reject rate přibližně 20%.

Lift, QLift

- Lift can be expressed and computed by formula:

$$Lift(a) = \frac{F_{m.BAD}(a)}{F_{N.ALL}(a)}, \quad a \in [L, H]$$

- In practice, Lift is computed corresponding to 10%, 20%, . . . , 100% of clients with the worst score. Hence we define:

$$QLift(q) = \frac{F_{m.BAD}(F^{-1}_{N.ALL}(q))}{F_{N.ALL}(F^{-1}_{N.ALL}(q))} = \frac{1}{q} F_{m.BAD}(F^{-1}_{N.ALL}(q)), \quad q \in (0,1]$$

$$F^{-1}_{N.ALL}(q) = \min\{a \in [L, H], F_{N.ALL}(a) \geq q\}$$

- Typical value of q is 0.1. Then we have

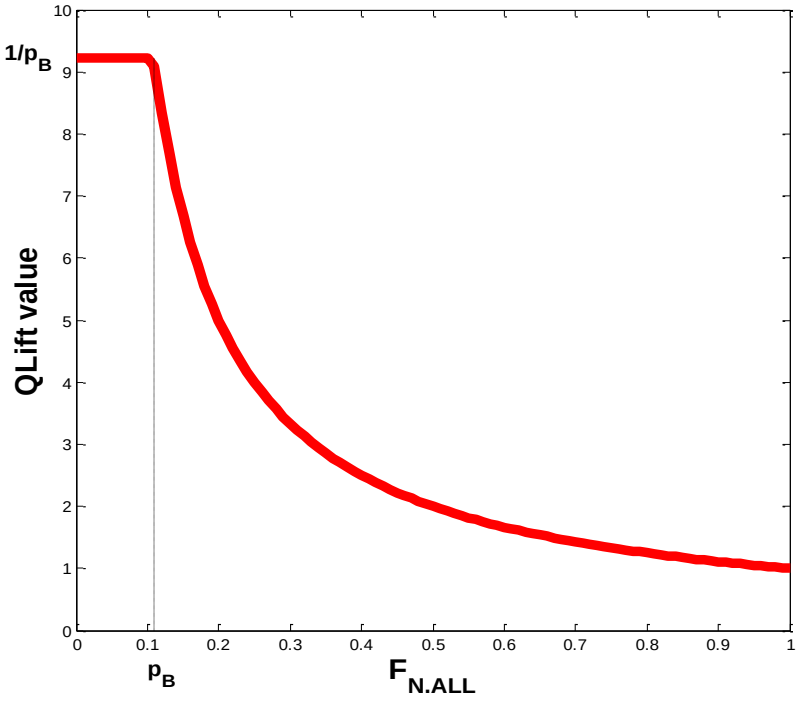
$$QLift_{10\%} = QLift(0.1) = 10 \cdot F_{m.BAD}(F^{-1}_{N.ALL}(0.1))$$

Lift and QLift for ideal model

- It is natural to ask how look Lift and QLift in case of ideal model. Hence we derived following formulas.

- Lift for ideal model:

$$Lift_{ideal}(a) = \begin{cases} \frac{1}{p_B}, & a \leq c \\ \frac{1}{F_{N.ALL}(a)}, & a > c \end{cases}$$



- QLift for ideal model:

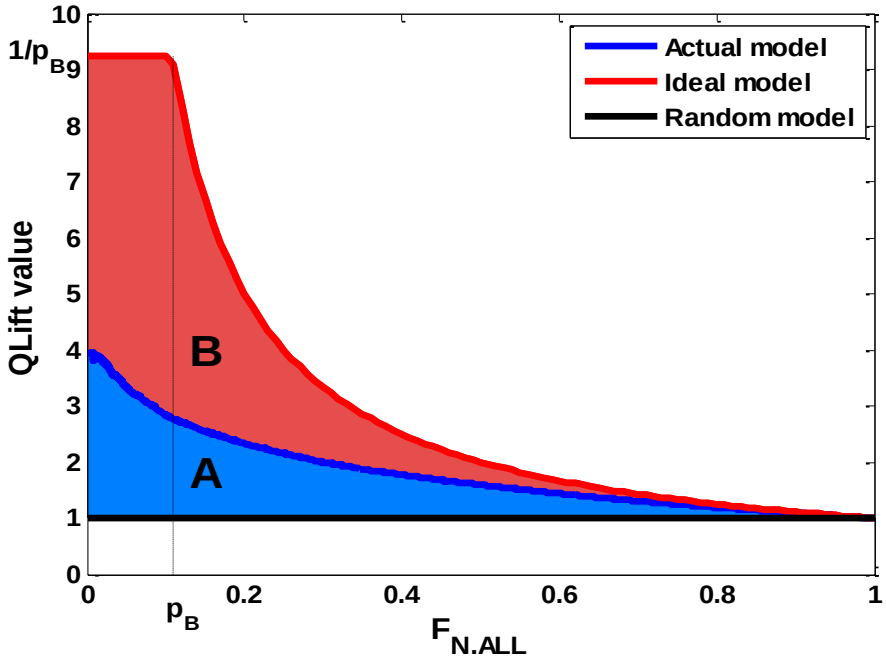
$$QLift_{ideal}(q) = \begin{cases} \frac{1}{p_B}, & q \in (0, p_B] \\ \frac{1}{q}, & q \in (p_B, 1] \end{cases}$$

We can see that the upper limit of Lift and QLift is equal to $1/p_B$.

Lift Ratio (LR)

- Once we know form of QLift for ideal model, we can define Lift Ratio as analogy to Gini index.

$$LR = \frac{A}{A+B} = \frac{\int_0^1 QLift(q)\,dq - 1}{\int_0^1 QLift_{ideal}(q)\,dq - 1}$$

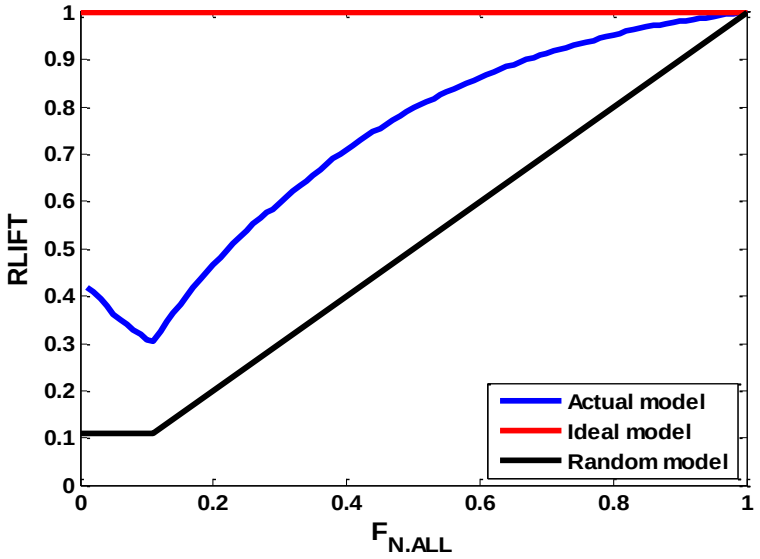


- It is obvious that it is global measure of model's quality and that it takes values from 0 to 1. Value 0 corresponds to random model, value 1 match to ideal model. Meaning of this index is quite simple. The higher, the better. Important feature is that Lift Ratio allows us to fairly compare two models developed on different data samples, which is not possible with Lift.

Rlift, IRL

- Since Lift Ratio compares areas under Lift function for actual and ideal models, next concept is focused on comparison of Lift functions themselves. We define Relative Lift function by

$$RLift(q) = \frac{QLift(q)}{QLift_{ideal}(q)}, \quad q \in (0, 1]$$



- In connection to RLift we define Integrated Relative Lift (IRL):

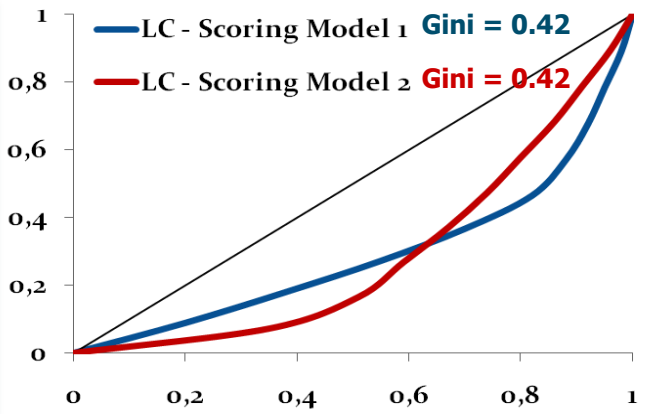
$$IRL = \int_0^1 RLift(q)\, dq$$

- It takes values from $0.5 + \frac{p_B^2}{2}$, for random model, to 1, for ideal model. Following simulation study shows interesting connection to c-statistics.

Příklad

- We consider two scoring models with score distribution given in the table below.
- We consider standard meaning of scores, i.e. higher score band means better clients (the highest probability of default have clients with the lowest scores, i.e. clients in score band 1).
- Gini indexes are equal for both models.
- From the Lorenz curves is evident, that the first model is stronger for higher score bands and the second one is better for lower score bands.
- The same we can read from values of QLift.

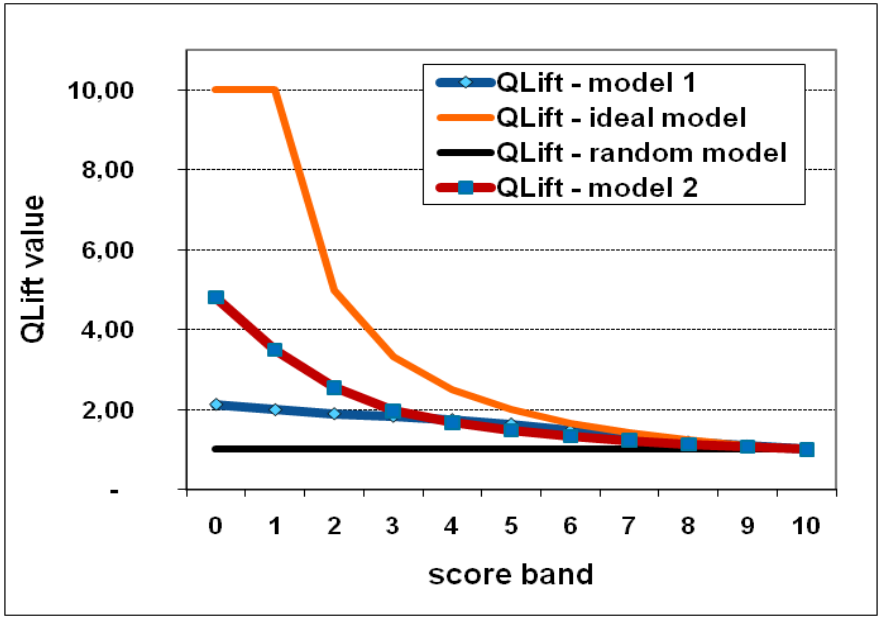
			Scoring Model 1				Scoring Model 2			
score band	# clients	q	# bad clients	# cumul. bad clients	# cumul. bad rate	QLift	# bad clients	# cumul. bad clients	# cumul. bad rate	QLift
1	100	0.1	20	20	20.0%	**2.00**	35	35	35.0%	**3.50**
2	100	0.2	18	38	19.0%	**1.90**	16	51	25.5%	**2.55**
3	100	0.3	17	55	18.3%	**1.83**	8	59	19.7%	**1.97**
4	100	0.4	15	70	17.5%	**1.75**	8	67	16.8%	**1.68**
5	100	0.5	12	82	16.4%	**1.64**	7	74	14.8%	**1.48**
6	100	0.6	6	88	14.7%	**1.47**	6	80	13.3%	**1.33**
7	100	0.7	4	92	13.1%	**1.31**	6	86	12.3%	**1.23**
8	100	0.8	3	95	11.9%	**1.19**	5	91	11.4%	**1.14**
9	100	0.9	3	98	10.9%	**1.09**	5	96	10.7%	**1.07**
10	100	1.0	2	100	10.0%	**1.00**	4	100	10.0%	**1.00**
All	1000		100				100			

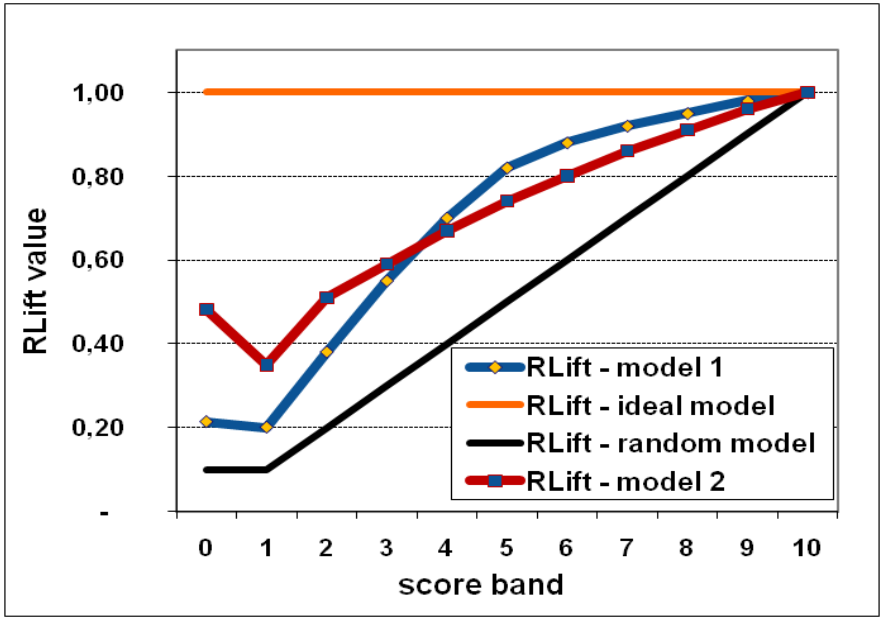


Příklad

- Since Qlift is not defined for *q=0*, we extrapolated the value by

$$QLift(0) = 3 \cdot QLift(0.1) - 3 \cdot QLift(0.2) + QLift(0.3)$$



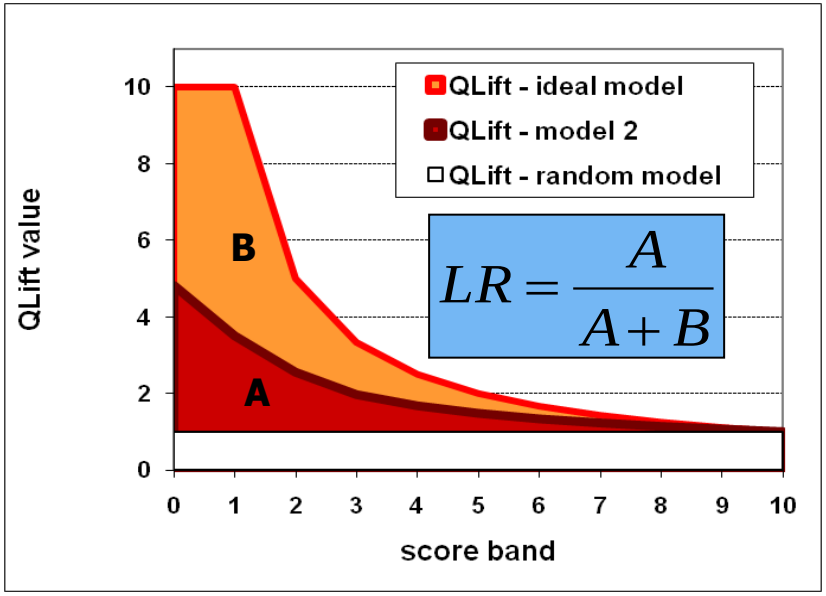


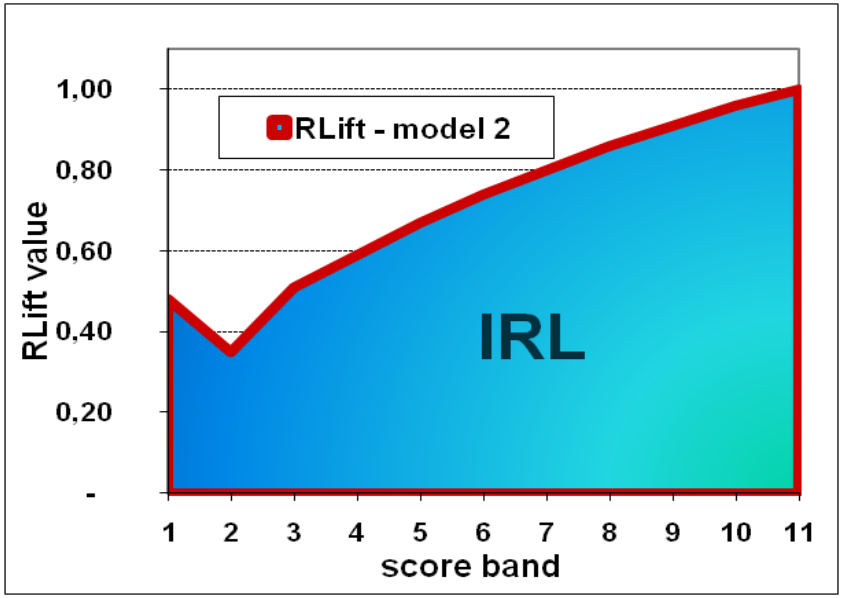
According to both Qlift and Rlift curves we can state that:

- If expected reject rate is up to 40%, then model 2 is better.
- If expected reject rate is more than 40%, then model 1 is better.

Příklad

- Now, we consider indexes LR and IRL:





	scoring model 1	scoring model 2
GINI	**0.420**	**0.420**
QLift(0.1)	**2.000**	**3.500**
LR	**0.242**	**0.372**
IRL	**0.699**	**0.713**

Using LR and IRL we can state that model 2 is better than model 1 although their Gini coefficients are equal.

Střední diference

- Střední diference (Mahalanobis distance):

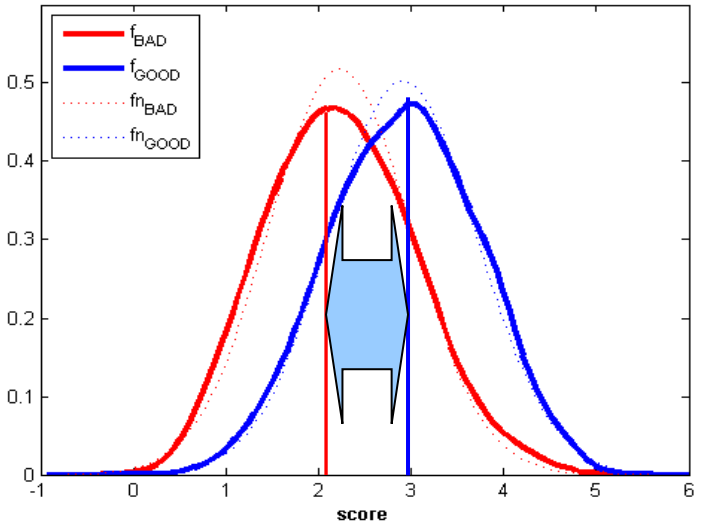
$$D = \frac{M_g - M_b}{S}$$

kde S je společná směrodatná odchylka:

$$S = \left(\frac{nS_g^{\ 2} + mS_b^{\ 2}}{n+m} \right)^{\frac{1}{2}}$$

M_g , M_b jsou střední hodnoty dobrých (špatných) klientů

S_g , S_b jsou příslušné směrodatné odchylky.



Normálně rozložené skóre

- Předpokládejme, že skóre dobrých a špatných klientů je normálně rozloženo, tj. jejich pravděpodobnostní hustoty mají tvar

$$f_{GOOD}(x) = \frac{1}{\sigma_g \sqrt{2\pi}} e^{-\frac{(x-\mu_g)^2}{2\sigma_g^2}} \qquad f_{BAD}(x) = \frac{1}{\sigma_b \sqrt{2\pi}} e^{-\frac{(x-\mu_b)^2}{2\sigma_b^2}}$$

- Odhady parametrů μ_b, μ_b, σ_g a σ_b :

  M_g , M_b jsou aritmetické průměty skóre dobrých (špatných) klientů

  S_g , S_b jsou směrodatné odchylky skóre dobrých (špatných) klientů

- Společná směrodatná odchylka:

$$S = \left(\frac{nS_g^{\,2} + mS_b^{\,2}}{n+m} \right)^{\frac{1}{2}}$$

- Odhady střední hodnoty a směrodatné odchylky skóre všech klientů μ_{ALL} , σ_{ALL}:

$$M = M_{ALL} = \frac{nM_g + mM_b}{n+m} \qquad S_{ALL} = \left(\frac{nS_g^{\,2} + mS_b^{\,2} + n(M_g - M)^2 + m(M_b - M)^2}{(n+m)} \right)^{\frac{1}{2}}$$

Normálně rozložené skóre

- Předpokládejme, že směrodatné odchylky obou skóre jsou rovny hodnotě σ, pak:

$$D = \frac{\mu_g - \mu_b}{\sigma}$$

$$D = \frac{M_g - M_b}{S}$$

$$KS = \Phi\left(\frac{D}{2}\right) - \Phi\left(\frac{-D}{2}\right) = 2 \cdot \Phi\left(\frac{D}{2}\right) - 1$$

$$Gini = 2 \cdot \Phi\left(\frac{D}{\sqrt{2}}\right) - 1$$

$$Lift_q = \frac{1}{q}\Phi\left(\frac{\sigma_{ALL}}{\sigma} \cdot \Phi^{-1}(q) + p_G \cdot D\right)$$

$$Lift_q = \frac{1}{q}\Phi\left(\frac{S_{ALL}}{S}\Phi^{-1}(q) + p_G \cdot D\right)$$

Kde $\Phi(\cdot)$ je distribuční funkce standardizovaného normálního rozložení, $\Phi_{\mu,\sigma^2}(\cdot)$ je distribuční funkce s parametry μ , σ^2 a $\Phi^{-1}(\cdot)$ je standardizovaná kvantilová funkce.

Normálně rozložené skóre

- Obecně, tj. bez předpokladu rovnosti směrodatných odchylek skóre:

$$D^* = \frac{\mu_g - \mu_b}{\sqrt{\sigma_g^2 + \sigma_b^2}}$$

$$D^* = \frac{M_g - M_b}{\sqrt{S_g^2 + S_b^2}}$$

$$KS = \Phi\left(\frac{a}{b}\sigma_b \cdot D^* - \frac{1}{b}\sigma_g\sqrt{a^2 D^{*2} + 2b \cdot c}\right) - \Phi\left(\frac{a}{b}\sigma_g \cdot D^* - \frac{1}{b}\sigma_b\sqrt{a^2 D^{*2} + 2b \cdot c}\right)$$

kde $a = \sqrt{\sigma_b^2 + \sigma_g^2}, \quad b = \sigma_b^2 - \sigma_g^2, \quad c = \ln\left(\frac{\sigma_g}{\sigma_b}\right)$

$$KS = \Phi\left(\frac{\sqrt{S_b^2 + S_g^2}}{S_b^2 - S_g^2} S_b \cdot D^* - \frac{1}{S_b^2 - S_g^2} S_g \sqrt{\left(S_b^2 + S_g^2\right) D^{*2} + 2 \cdot \left(S_b^2 - S_g^2\right) \ln\left(\frac{S_g}{S_b}\right)}\right)$$

$$-\Phi\left(\frac{\sqrt{S_b^2 + S_g^2}}{S_b^2 - S_g^2} S_g \cdot D^* - \frac{1}{S_b^2 - S_g^2} S_b \sqrt{\left(S_b^2 + S_g^2\right) D^{*2} + 2 \cdot \left(S_b^2 - S_g^2\right) \ln\left(\frac{S_g}{S_b}\right)}\right)$$

Normálně rozložené skóre

- Obecně, tj. bez předpokladu rovnosti směrodatných odchylek skóre:

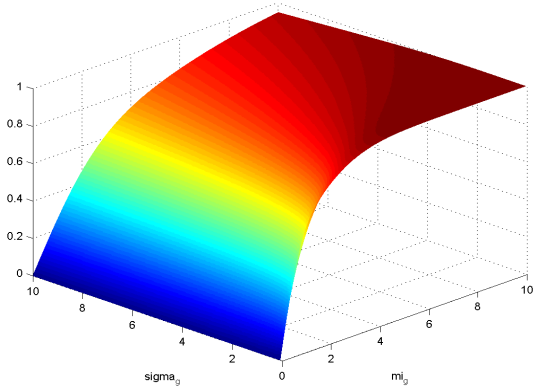
$$Gini = 2 \cdot \Phi\left(D^*\right) - 1$$

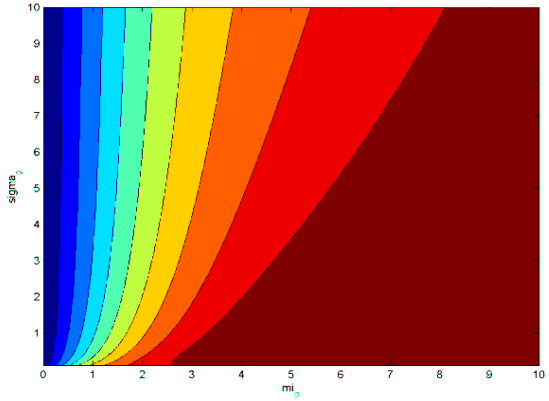
$$Lift_q = \frac{1}{q} \Phi_{\mu_b, \sigma_b^2}\left(\mu_{ALL} + \sigma_{ALL} \cdot \Phi^{-1}(q)\right) = \frac{1}{q} \Phi\left(\frac{\sigma_{ALL} \cdot \Phi^{-1}(q) + \mu_{ALL} - \mu_b}{\sigma_b}\right)$$

$$Lift_q = \frac{1}{q} \Phi\left(\frac{S_{ALL} \cdot \Phi^{-1}(q) + M - M_b}{S_b}\right)$$

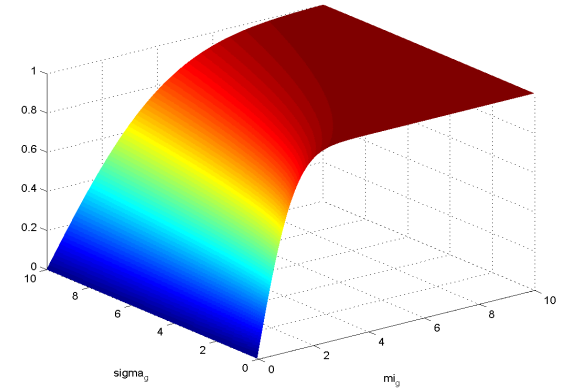
Normálně rozložené skóre

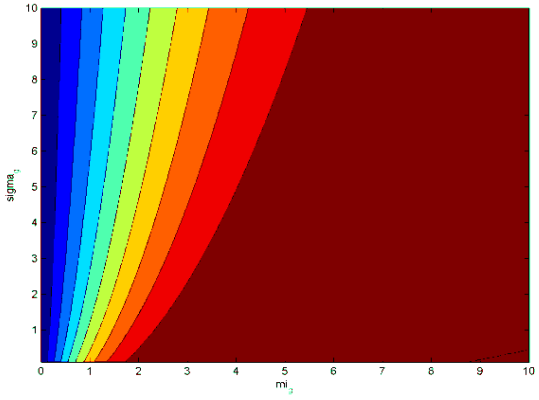
- **KS:** $\mu_b = 0, \sigma_b^2 = 1$





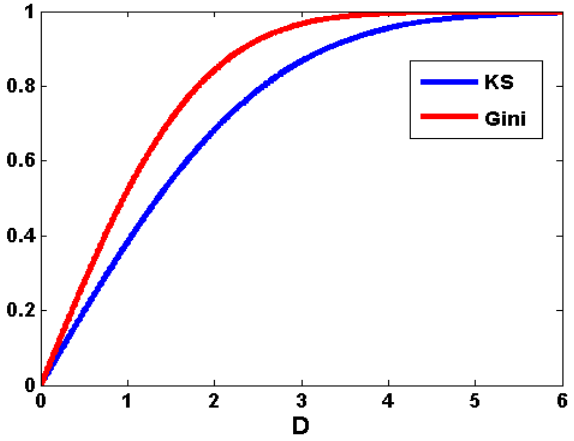
- **Gini** $\mu_b = 0, \sigma_b^2 = 1$





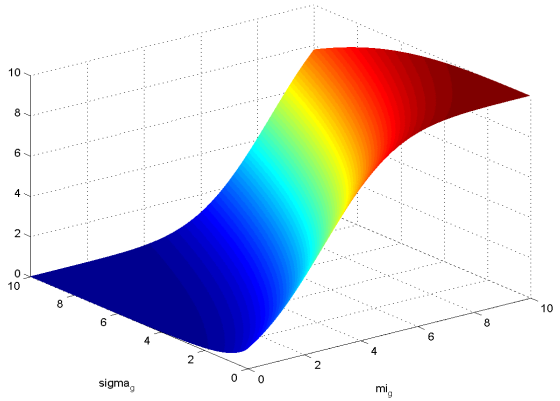
- KS i Gini reagují velmi silně na změnu μ_g, ale zůstávají téměř beze změny ve směru σ_g^2.

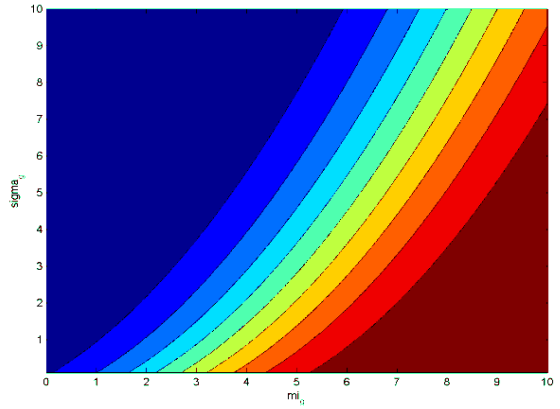
- **Gini > KS**



Normálně rozložené skóre

- **Lift$_{10\%}$:** $\mu_b = 0$, $\sigma_b^2 = 1$





- V případě indexu Lift$_{10\%}$ je evidentní silná závislost na μ_g a významně vyšší závislost na σ_g^2 než v případě KS a Gini.

ROC (Receiver operating characteristic)

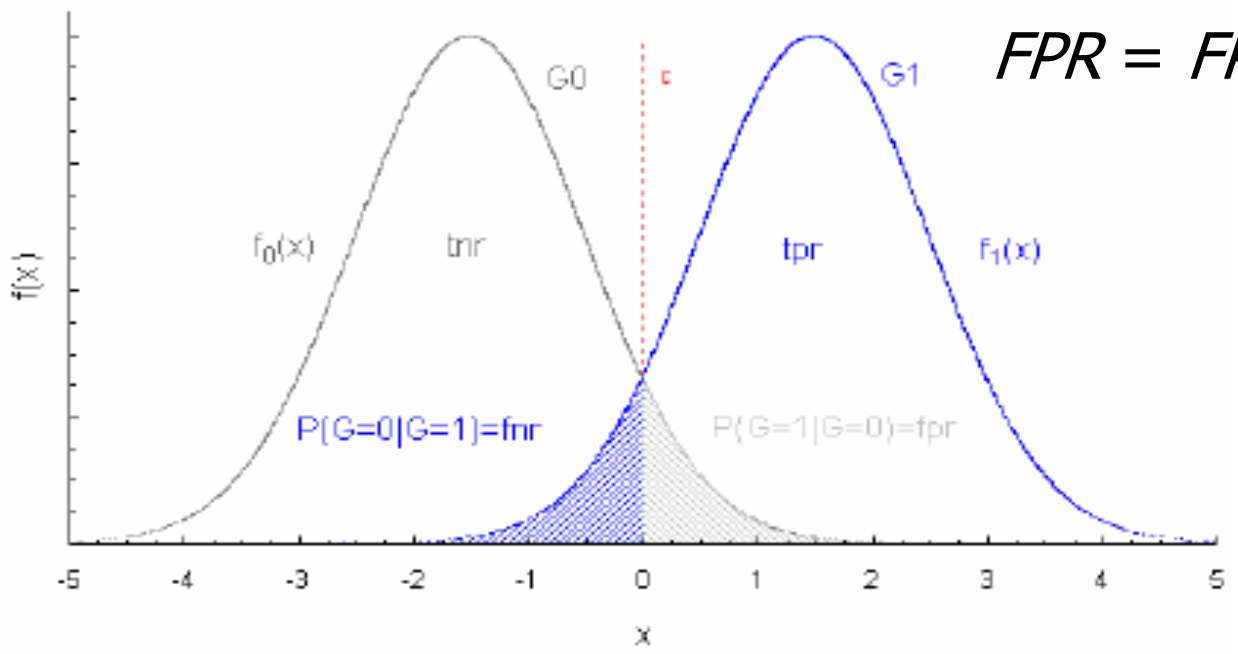
- *TN* (*true negative*) - počet správně klasifikovaných negativních případů
- *TP* (*true positive*) – počet správně klasifikovaných positivních případů
- *FP (false positive)* – počet nesprávně klasifikovaných negativních případů
- *FN (false negative)* – počet nesprávně klasifikovaných positivních případů

	Predikce		
Skuteč.	**G0**	**G1**	**Celkem**
G0	*TN*	*FP*	*N*
G1	*FN*	*TP*	*P*
Celkem	*PNeg*	*PPos*	*n*

ROC –TPR, FPR

$$TPR = TP / P = TP / (TP + FN)$$

$$FPR = FP / N = FP / (FP + TN)$$



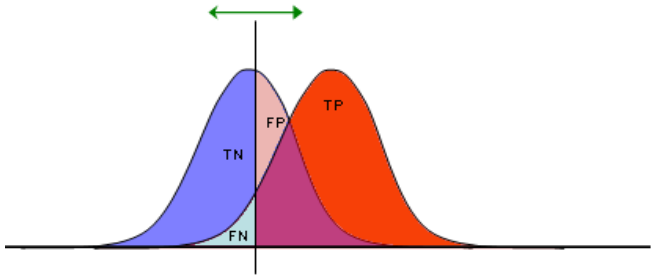
$$tpr(c) = P(X > c \mid G_1) = 1 - F_1(c)$$

$$tnr(c) = P(X < c \mid G_0) = F_0(c)$$

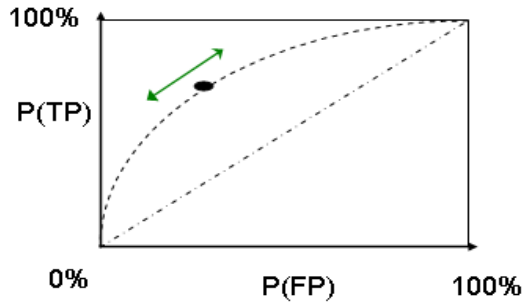
$$fpr(c) = P(X > c \mid G_0) = 1 - F_0(c)$$

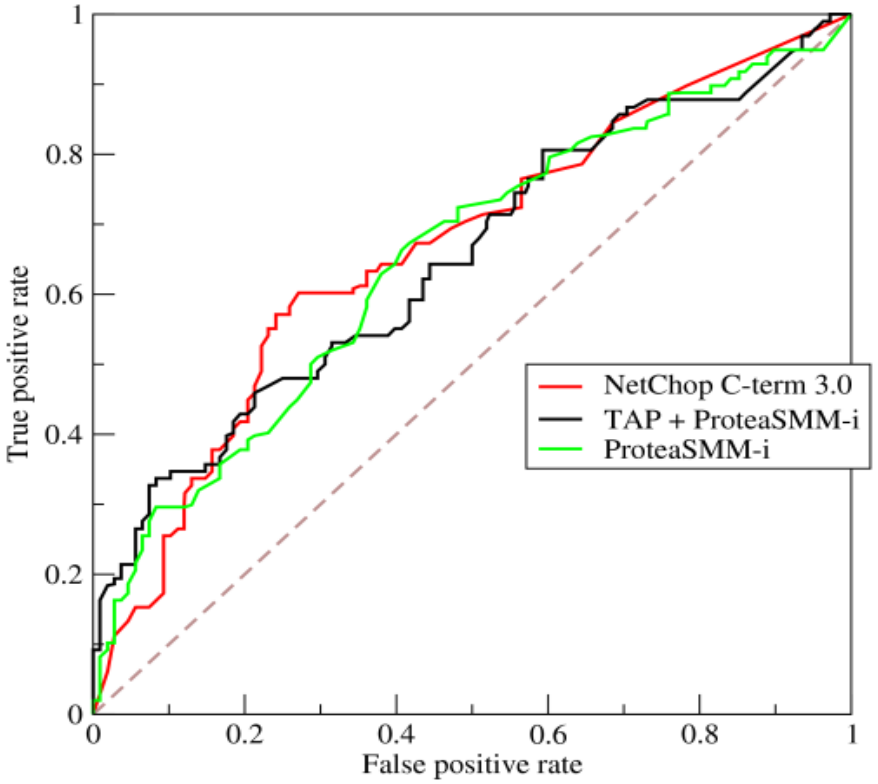
$$fnr(c) = P(X < c \mid G_1) = F_1(c)$$

ROC



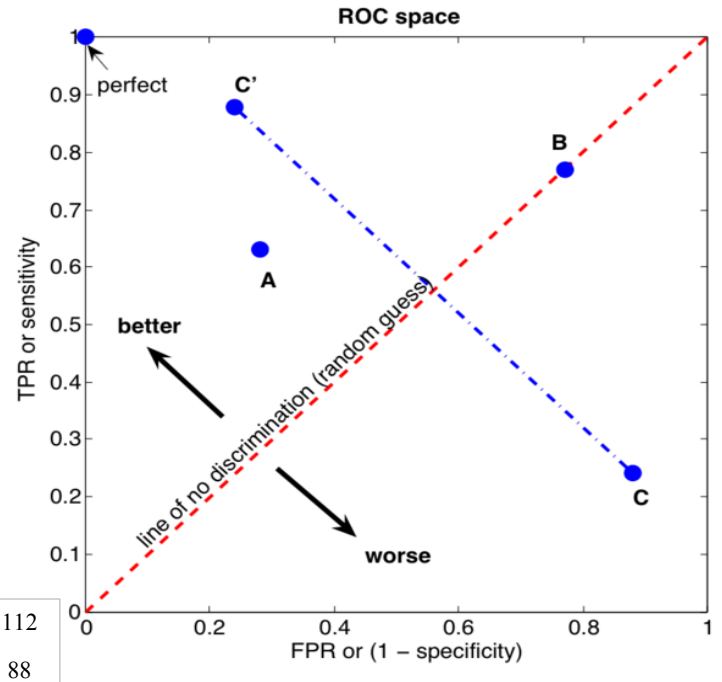
TP	FP
FN	TN
1	1





ROC - ACC

Accuracy:
ACC = (*TP* + *TN*) / (*P* + *N*)



A			B			C			C'		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=88	FP=24	112
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=12	TN=76	88
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.88		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.24		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

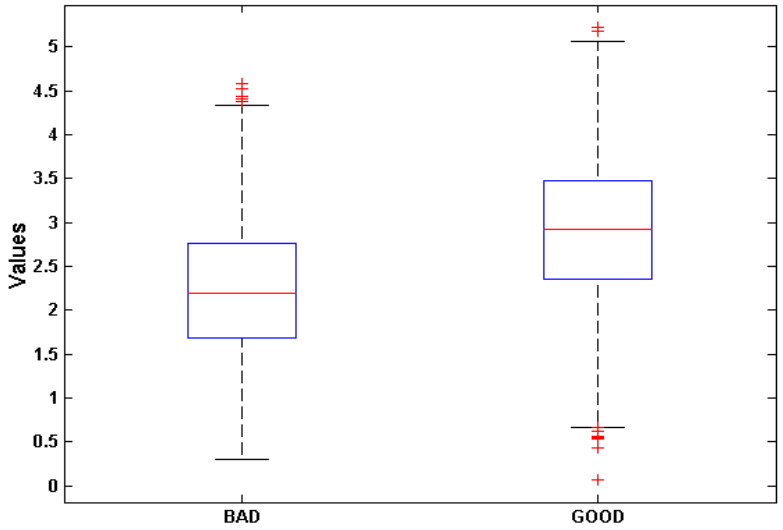
ROC – AUC, Gini

AUC (area under curve, neboli plocha pod ROC křivkou) je rovna pravděpodobnosti, že daný model ohodnotí náhodně vybraného dobrého klienta vyšším skóre než náhodně vybraného špatného klienta. Dá se ukázat, že plocha pod ROC křivkou se dá vyjádřit pomocí Mann-Whitneymu U, které testuje rozdíl mediánů mezi dvěma skupinami spojitých skóre. AUC se dá vyjádřit i pomocí Giniho koeficientu pomocí vzorce

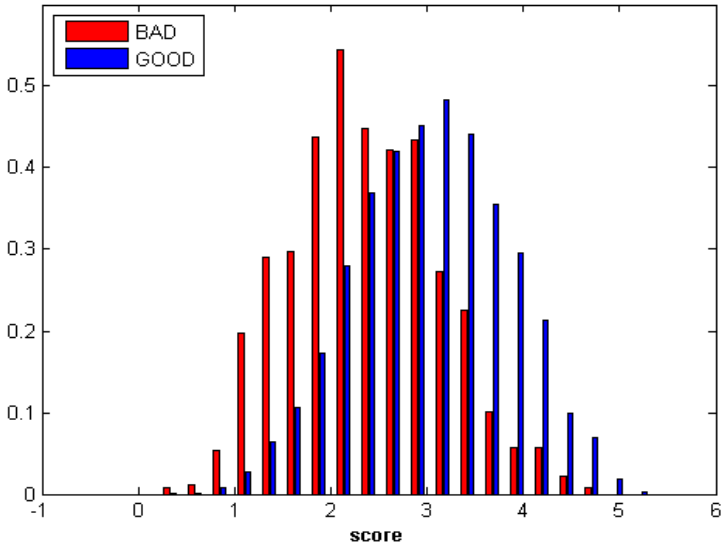
$Gini + 1 = 2xAUC$

Další evaluační grafy

Boxplot

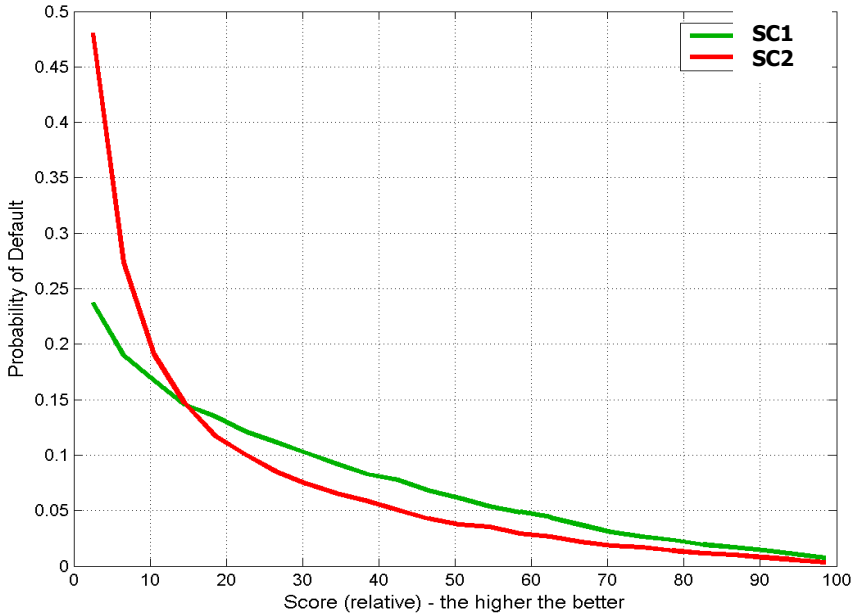


Histogram

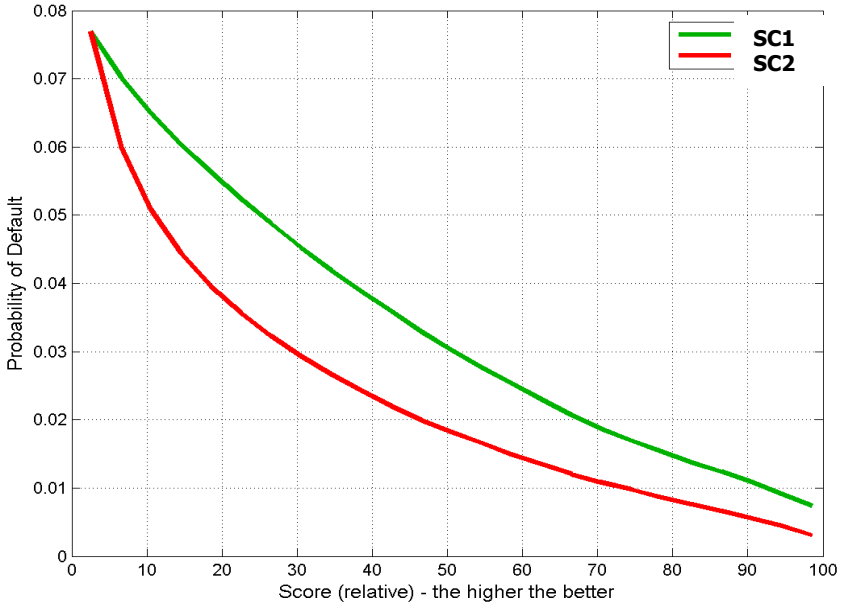


Další evaluační grafy

PD - absolutně

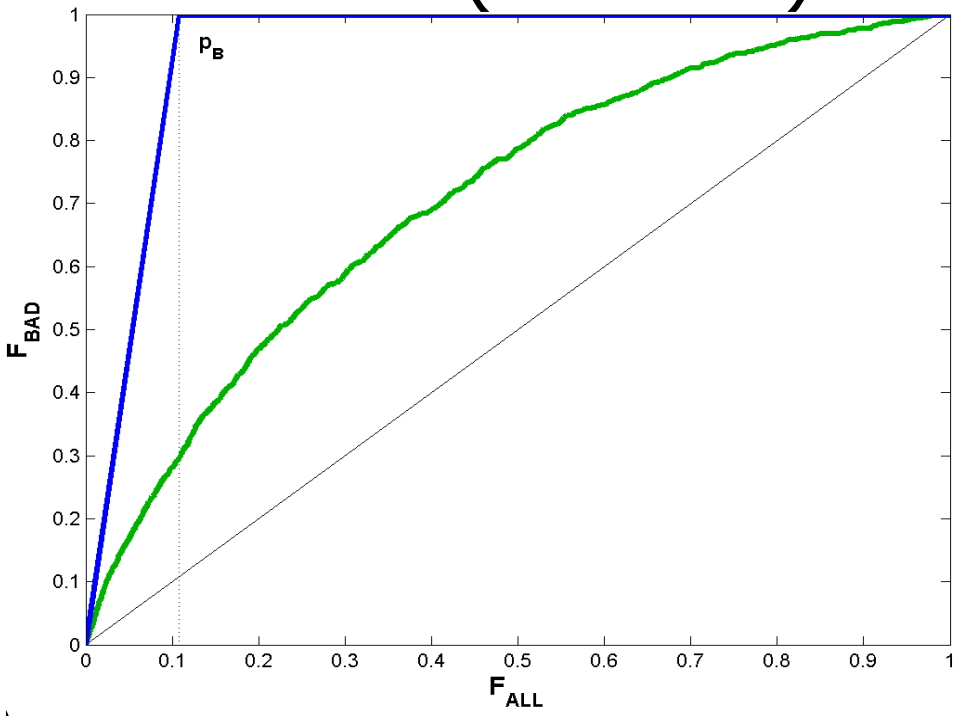


PD - kumulativně



Další evaluační grafy

CAP (Lift chart):



AR (Accuracy Ratio)

$$\mathrm{AR} = \frac{\text{Plocha mezi CAP a diagonálou}}{\text{Plocha mezi CAP ideálního modelu a diagonálou}}$$

$$= \frac{\text{Plocha mezi CAP a diagonálou}}{0.5(1-p_B)} = Gini$$

V tomto případě máme na x-ové ose proporci všech klientů (F_{ALL}) a na y-vé ose proporci špatných klientů (F_{BAD}). Ideální model je tentokrát reprezentován lomenou čarou z bodu [0, 0] přes [pB, 1] do bodu [1, 1]. Výhoda tohoto obrázku je ta, že je možné odečíst proporci zamítnutých špatných klientů vs. celková proporce zamítnutých klientů. Např. vidíme, že pokud chceme zamítnout 70% špatných klientů, musíme zamítat přibližně 40% všech žadatelů.

Postupy evaluace

- **evaluace na učících datech**

Evaluace na učících datech použitých k učícímu procesu není ke zjištění kvality modelu vhodná a má nízkou vypovídací schopnost, protože často může dojít k přeučení modelu. Odhad predikční kvality modelu na učících datech se nazývá resubstituční nebo interní odhad. Odhady ukazatelů kvality modelů provedených na učících datech jsou nadhodnocené, proto se místo nich používají testovací data, která se v rámci přípravy dat pro tyto účely vyčlení.

Postupy evaluace

- **evaluace na testovacích datech**

Evaluace na testovacích datech již má patřičnou vypovídací schopnost, jelikož tato data nebyla použita k sestavení modelu. Na testovací data jsou kladeny určité požadavky. Soubor testovacích dat by měl obsahovat dostatečné množství dat a měl by reprezentovat či vystihovat charakteristiky učících dat. Empiricky doporučený poměr učících a testovacích dat je 75%, resp. 25% případů. Zajištění patřičné reprezentativnosti je realizováno pomocí náhodného stratifikovaného výběru.

Postupy evaluace

➢ **křížové ověřování** *(cross-validation)*

V případě nedostatečného počtu pozorování, kdy rozdělení datového souboru na učící a testovací data za účelem vyhodnocení modelu není možné, je vhodné použít metodu křížového ověřování. Výhodou této metody na rozdíl od dělení datového souboru je, že každý případ z dat je použit k sestavení modelu a každý případ je alespoň jednou použit k testování. Postup je následující:

- Soubor dat je náhodně rozdělen do *n* disjunktních podmnožin tak, že každá podmnožina obsahuje přibližně stejný počet záznamů. Výběry jsou stratifikovány podle tříd (příslušnosti k určité třídě), aby bylo zajištěno, že podíly jednotlivých tříd podmnožin jsou zhruba stejné jako v celém souboru.
- Z těchto *n* disjunktních podmnožin se vyčlení *n-1* podmnožin pro sestavení modelu (konstrukční podmnožina) a zbývající podmnožina (validační podmnožina) je použita k jeho vyhodnocení. Model je tedy evaluován na podmnožině dat, ze kterých nebyl sestaven a na této množině dat je odhadována jeho predikční kvalita.
- Celý postup se zopakuje *n*-krát a dílčí odhady ukazatelů kvality se zprůměrňují. Velikost validační podmnožiny lze přibližně stanovit jako poměr počtu případů ku počtu validačních podmnožin.

Postupy evaluace

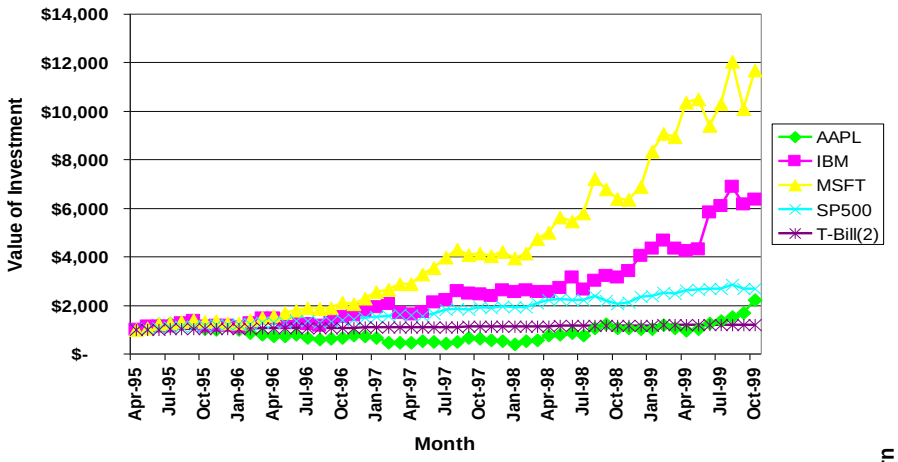
- ***bootstrap* metoda**

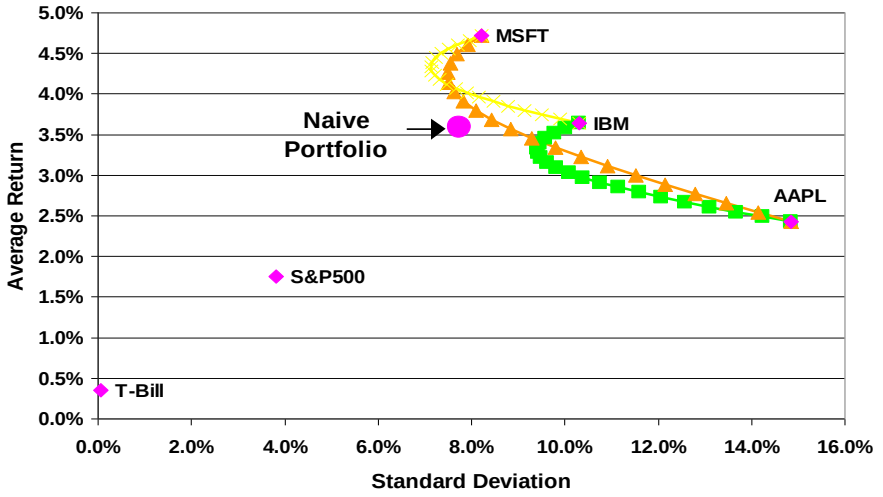
Metoda *bootstrap* zkoumá charakteristiky jednotlivých resamplovaných vzorků, které byly pořízeny z empirického výběru. Pokud původní výběr osahuje *m* prvků, tak každý má naději objevit se v resamplovaném výběru. Při úplném resamplování o velikosti vzorku *n* jsou uvažovány všechny možné výběry a existuje tedy m^n možných výběrů. Úplné resamplování je teoreticky proveditelné, ale vyžádalo by si mnoho času. Alternativou je simulace *Monte Carlo*, pomocí níž se aproximuje úplné resamplování tak, že se provede *B* náhodných výběrů (obvykle se volí 500 – 10000 výběrů) s tím, že každý prvek je vždy nahrazen (vrácen zpět do osudí). Jsou-li dána data $X=\{X1, \ldots, Xn\}$ a je-li požadován odhad parametru θ, provede se z původních dat *B* výběrů a pro každý výběr je spočítán odhad parametru θ . *Bootstrap* odhad parametru je určen jako průměr dílčích odhadů. V případě evaluace modelů bude parametrem θ zvolený ukazatel predikční kvality.

- ***jackknife***

Tato metoda je založena na sekvenční strategii odebírání a vracení prvků do výběru o velikosti *n*. Pro datový soubor, který obsahuje *n* prvků, procedura generuje *n* vzorků s počtem prvků *n-1*. Pro každý zmenšený výběr o velikosti *n*-1 je odhadnuta hodnota parametru. Dílčí odhady se následně zprůměrují podobně jako u metody *bootstrap*.

5. Úvod do teorie portfolia





Stručná historie teorie portfolia

- J. Hickse: Application of Mathematical Methods to the Theory of Risk (1934) – investoři si všímají statistického rozdělení pravděpodobnosti dosažení výnosu
- Harry Markowitz: Portfolio Selection, Journal of Finance, březen 1952 – je považován za zakladatele moderní teorie portfolia

Harry Markowitz

- jako první se zabývá vztahem mezi výnosností a rizikem
- konstruuje efektivní hranici portfolií, která znázorňuje body s maximálním výnosem pro danou úroveň rizika
- tím pokládá základy pro teorii portfolia

Odkaz na přednášku k příležitosti udělení Nobelovy ceny:
http://nobelprize.org/economics/laureates/1990/markowitz-lecture.pdf

Harry Markowitz

- Markowitz předpokládá, že investor má na počátku období k dispozici *určité množství kapitálu*, který bude investovat na *předem určené časové období*, na jehož konci pak investor nakoupené a držené cenné papíry prodá a zisk buď použije pro vlastní potřebu nebo jej opět reinvestuje
- na *investování* se Markowitz dívá jako na *periodickou aktivitu*, při které si investor vybírá mezi investicemi s různými očekávanými výnosy a s různou mírou jistoty, že očekávaného výnosu bude dosaženo
- podle Markowitze sleduje investor dva protichůdné cíle a to *maximalizaci výnosu* na jedné straně a *minimalizaci rizika* na straně druhé

Další vývoj (1)

- model CAPM (model oceňování kapitálových aktiv) – základy položeny článkem W. F. Sharpe: Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk (1964) – dochází k rozšíření portfolia rizikových aktiv o bezrizikovou investici
- v návaznosti na možnost bezrizikového investování byla vytvořena přímka CML
- objevuje se také přímka SML

Další vývoj (2)

- důležitou etapou vývoje teorie portfolia je APT (arbitrážní teorie oceňování)
- není založena na myšlence, že všichni investoři pohlížejí na portfolio ve smyslu očekávaného výnosu a rizika dosažení tohoto výnosu
- je postaven na myšlence, že investoři dávají přednost vyšší úrovni bohatství před nižší

Základní pojmy

- portfolio – soubor různých investic (peněžní hotovost, cenné papíry včetně derivátů, nemovitosti atd.), které investor vytváří se záměrem minimalizovat riziko spojené s investováním a současně maximalizovat výnos z těchto investic
- teorie portfolia – jedná se o mikro-ekonomickou disciplínu, která zkoumá, jaké kombinace aktiv je vhodné držet, aby takto vytvořené portfolio mělo předem určené vlastnosti.

Aktiva v teorii portfolia

- portfolio je obvykle definováno jako skupina aktiv
- hmotná, nehmotná a finanční – dále budeme uvažovat pouze aktiva finanční, a to cenné papíry
- výnos(nost), riziko a likvidita – magický trojúhelník investování

Finanční aktiva

- finanční aktiva dělíme na
  - hotovost a depozita
  - cenné papíry – majetkové, dluhové, nárokové
- existují i jiné pohledy na členění aktiv
- dále nás budou zajímat především akcie

Výnosnost aktiv

- jedním z hlavních ukazatelů

$$r = \frac{P_t - P_{t-k} + D}{P_{t-k}}$$

- kde P_{t-k} je cena akcie v čase *t-k* (počátek sledovaného období), je cena P_t akcie v čase *t* (konec období).
- *D* jsou inkasované dividendy.
- pro *k* = 1 se jedná o jednodenní výnosnost

Očekávaný výnos a riziko

- Výnos akcie je náhodná veličina
- Očekávaný výnos portfolia:

$$r(\mathbf{x}) = \sum_{i=1}^{I} x_i r_i = \mathbf{r}'\mathbf{x}$$

- Riziko portfolia:

$$\sigma^2(\mathbf{x}) = \sum_{i,j=1}^{I} x_i x_j V_{ij} = \mathbf{x}'\mathbf{V}\mathbf{x}.$$

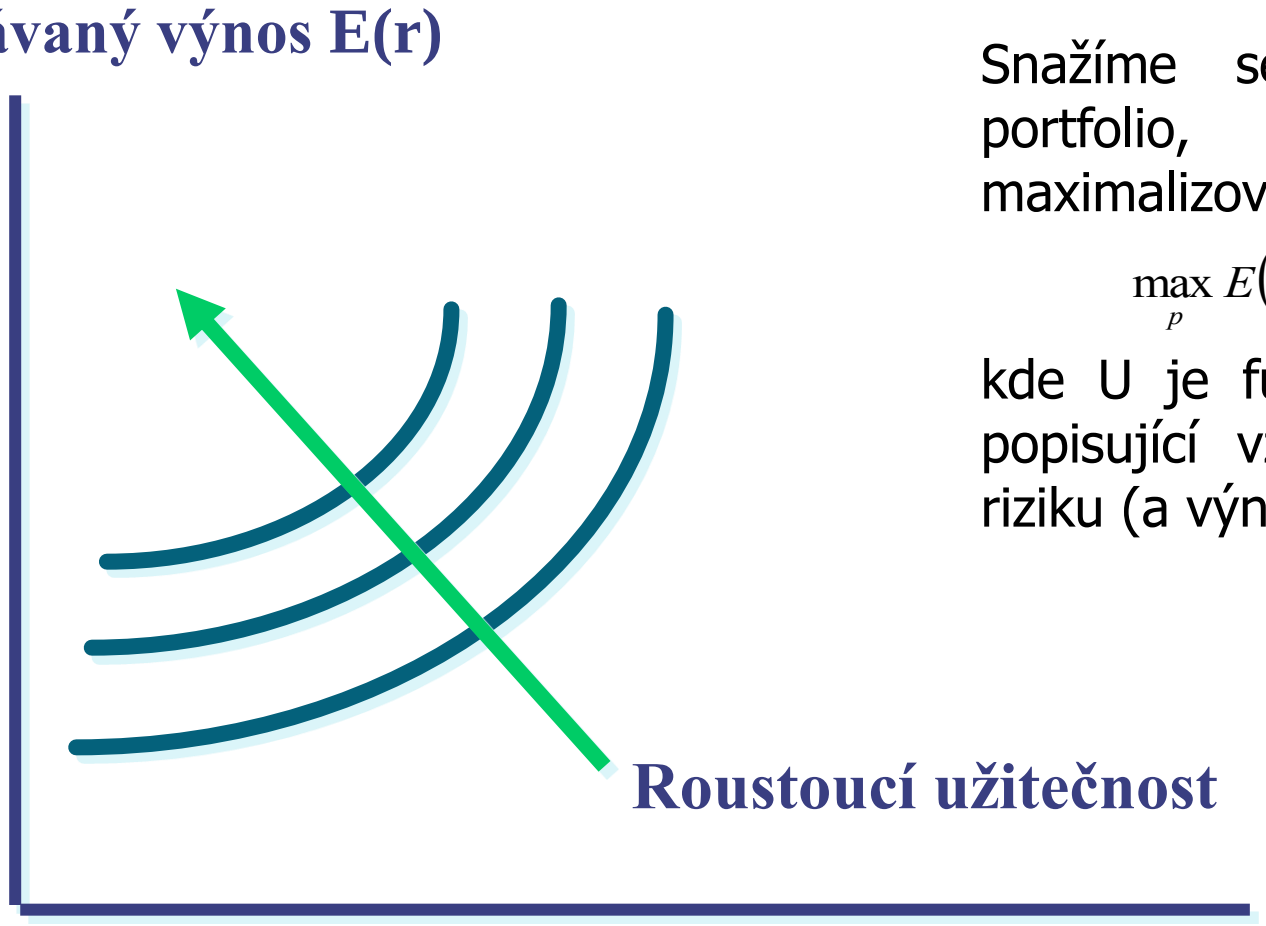
r_i...očekávaný výnos i-té akcie (střec

x_i...váhy investic do akcií v rámci portfolia

I...počet akcií v portfoliu

V_{ij}...varianční matice výnosů akcií

Indiferenční křivky, funkce užitečnosti

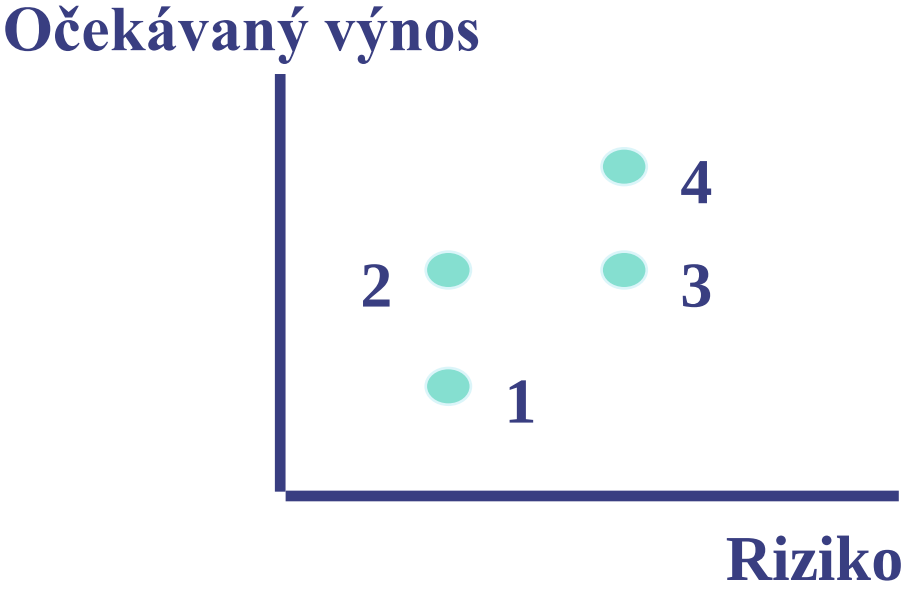


Snažíme se najít takové portfolio, aby byla maximalizována hodnota

$$\max_{p} E\left(U\left(r_p, \sigma_p\right)\right)$$

kde U je funkce užitečnosti popisující vztah investora k riziku (a výnosu).

Dominance



- 2 dominuje 1; má vyšší výnos
- 2 dominuje 3; má nižší riziko
- 4 dominates 3; má vyšší výnos

Příklad: 2 riziková aktiva

- Předpokládejme, že máme k dispozici 2 riziková akativa (x & y), obě normálně rozdělená.

$$r_x \sim N(E(r_x), \sigma^2_x) \;\&\; r_y \sim N(E(r_y), \sigma^2_y)$$

- Investujeme objem a do x, b do y.
- $a + b = 1$.
- Očekávaný výnos portfolia:

$$E(r_p) = E[ar_x + br_y] = aE(r_x) + bE(r_y)$$

Příklad: 2 riziková aktiva

- $r_x \sim N(E(r_x), \sigma^2_x)$ & $r_y \sim N(E(r_y), \sigma^2_y)$
- Riziko portfolia:

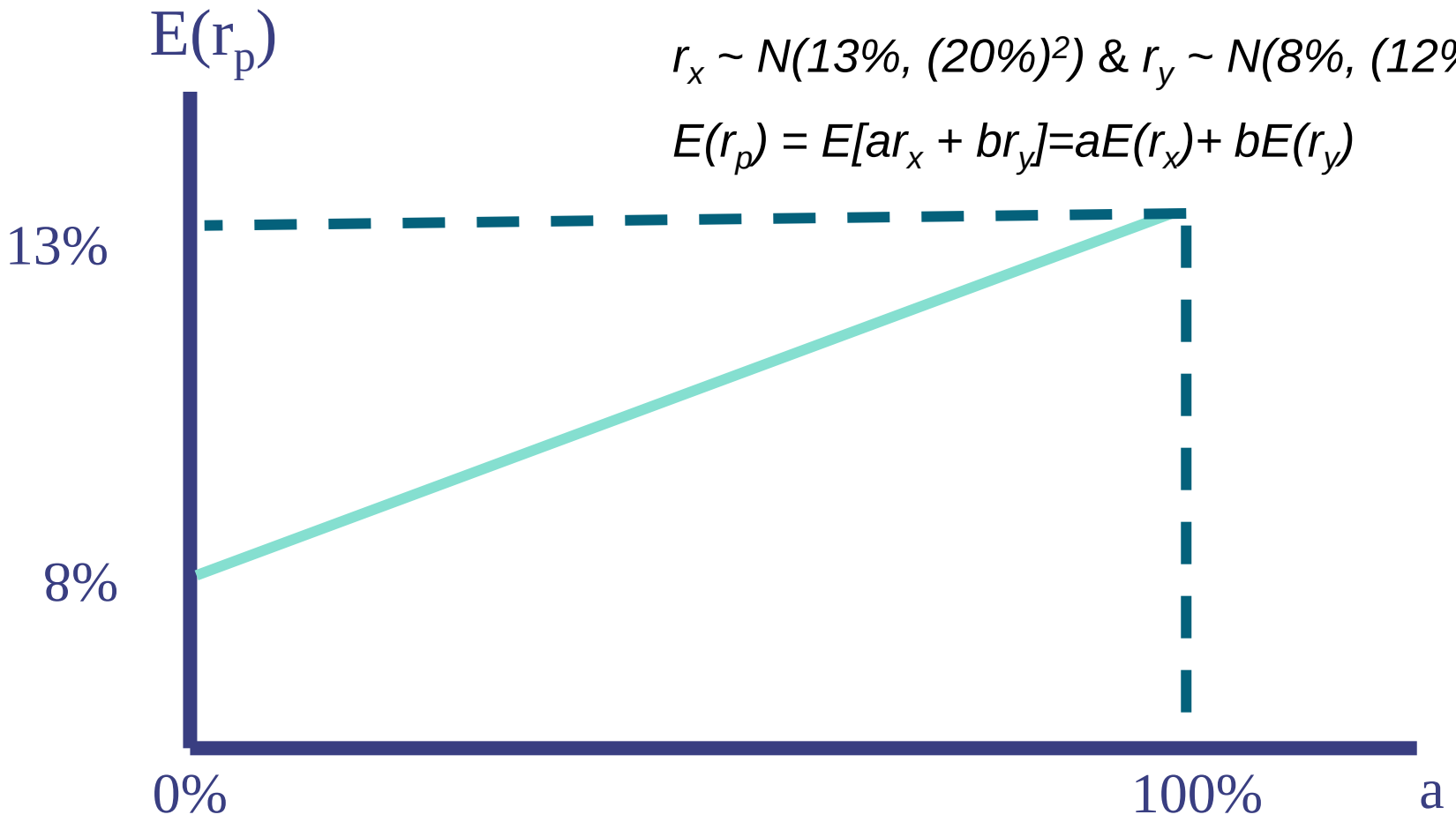
$$\begin{aligned}
\sigma^2_p &= E[r_p - E(r_p)]^2 \\
&= E[(ar_x + br_y) - E[ar_x + br_y]]^2 \\
&= E[(ar_x - aE[r_x]) + (br_y - bE[br_y])]^2 \\
&= E[a^2(r_x - E[r_x])^2 + b^2(r_y - E[r_y])^2 + 2ab(r_x - E[r_x])(r_y - E[r_y])] \\
&= a^2\sigma^2_x + b^2\sigma^2_y + 2abCov(r_x, r_y) \\
&= a^2\sigma^2_x + b^2\sigma^2_y + 2abCov(r_x, r_y) \\
\sigma^2_p &= a^2\sigma^2_x + b^2\sigma^2_y + 2ab\sigma_x\sigma_y\rho_{xy} \\
\sigma_p &= \sqrt{(a^2\sigma^2_x + b^2\sigma^2_y + 2ab\sigma_x\sigma_y\rho_{xy})}
\end{aligned}$$

Závislost výnosu na proporci investice do X a Y

Předpokládejme, že:

$r_x \sim N(13\%, (20\%)^2)$ & $r_y \sim N(8\%, (12\%)^2)$

$E(r_p) = E[ar_x + br_y] = aE(r_x) + bE(r_y)$

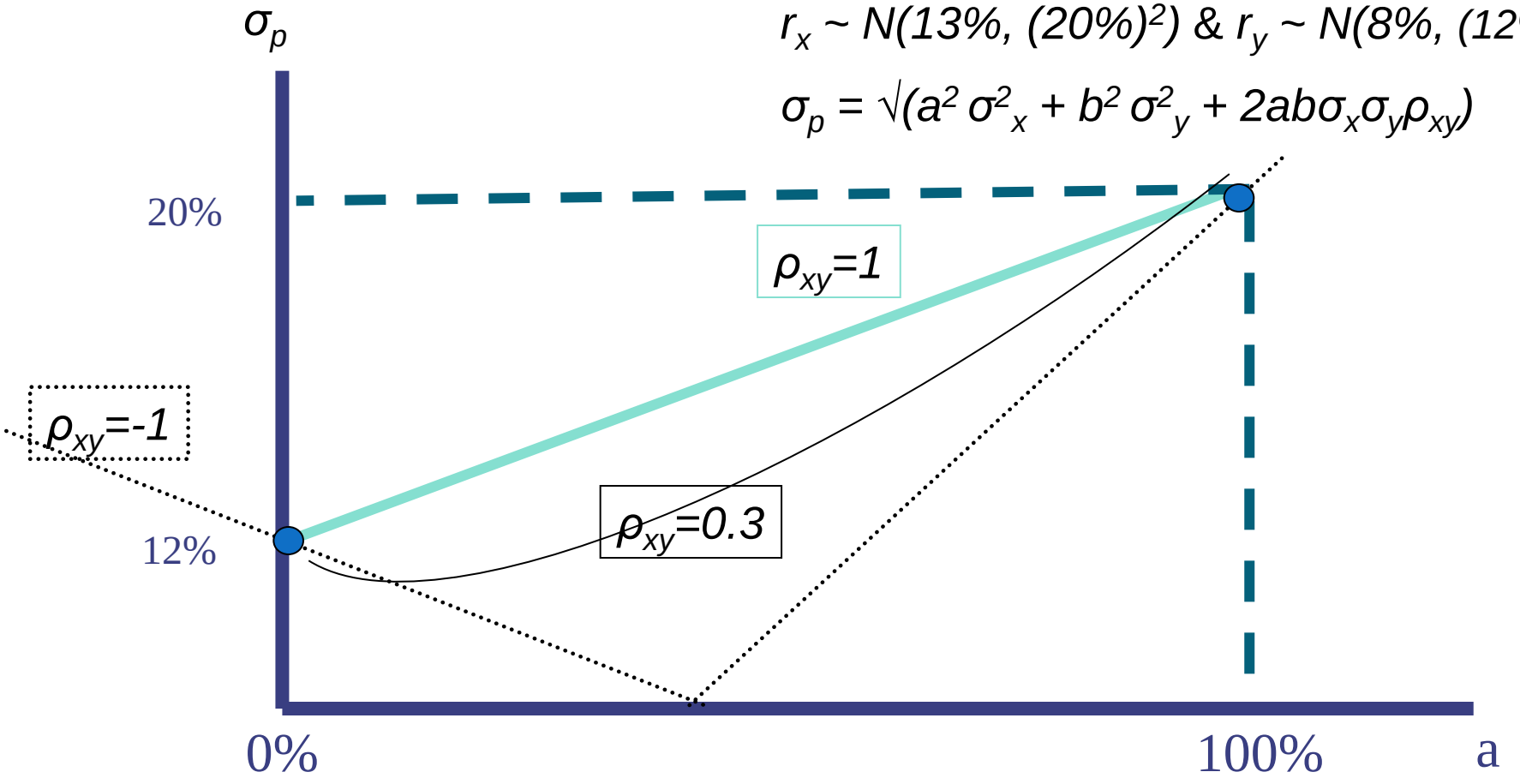


Závislost výnosu na proporci investice do X a Y

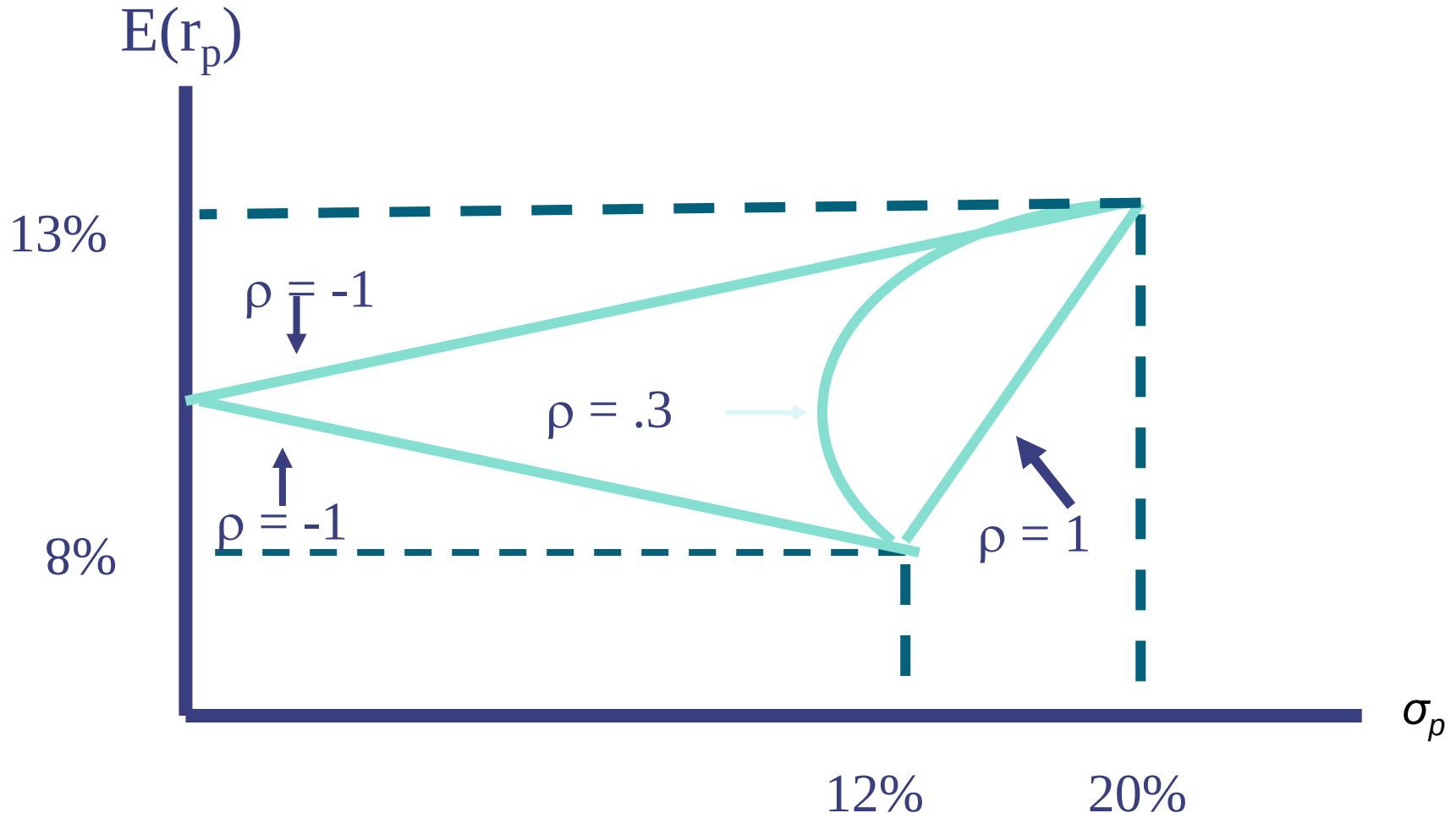
Předpokládejme, že:

$r_x \sim N(13\%, (20\%)^2)$ & $r_y \sim N(8\%, (12\%)^2)$

$\sigma_p = \sqrt{(a^2 \sigma^2_x + b^2 \sigma^2_y + 2ab\sigma_x\sigma_y\rho_{xy})}$



Portfolio dvou rizikových aktiv s minimálním rizikem



Portfolio dvou rizikových aktiv s minimálním rizikem -detail

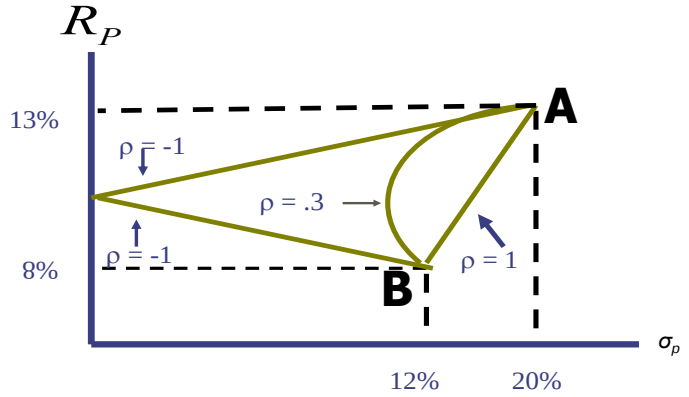
X_A ...váha investice do akcie A
X_B ...váha investice do akcie B
R_P...očekávaný výnos portfolia
R_A...očekávaný výnos investice do akcie A
R_B...očekávaný výnos investice do akcie B
σ_P...rozptyl výnosu portfolia
σ_A...rozptyl výnosu investice do akcie A
σ_B...rozptyl výnosu investice do akcie B
ρ_{AB}...korelační koeficient výnosů investic do akcií A, B

$$R_P = X_A R_A + (1 - X_A) R_B$$

$$\sigma_P = \left[X_A^2 \sigma_A^2 + (1 - X_A)^2 \sigma_B^2 + 2 X_A (1 - X_A) \rho_{AB} \sigma_A \sigma_B \right]^{1/2}$$

Portfolio dvou rizikových aktiv s minimálním rizikem -detail

- perfektní pozitivní korelace



$$\rho_{AB} = 1$$

$$\sigma_P = X_A\sigma_A + (1 - X_A)\sigma_B$$

$$X_A = \frac{\sigma_P - \sigma_B}{\sigma_A - \sigma_B}$$

$$R_P = \left(\frac{R_A - R_B}{\sigma_A - \sigma_B}\right)\sigma_P + \left(R_B - \frac{R_A - R_B}{\sigma_A - \sigma_B}\sigma_B\right)$$

Tedy vychází rovnice přímky.

Portfolio dvou rizikových aktiv s minimálním rizikem -detail

- perfektní negativní korelace

$$\rho_{AB} = -1$$

$$\sigma_P = X_A\sigma_A - (1 - X_A)\sigma_B$$

nebo

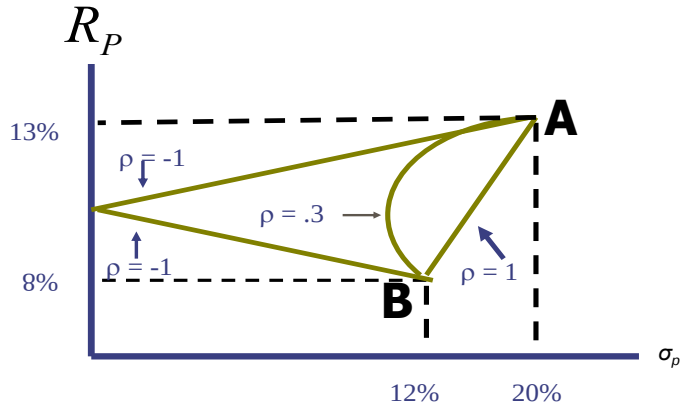
$$\sigma_P = -X_A\sigma_A + (1 - X_A)\sigma_B$$

$$X_A = \frac{\sigma_P + \sigma_B}{\sigma_A + \sigma_B}$$

$$X_A = \frac{\sigma_B - \sigma_P}{\sigma_A + \sigma_B}$$

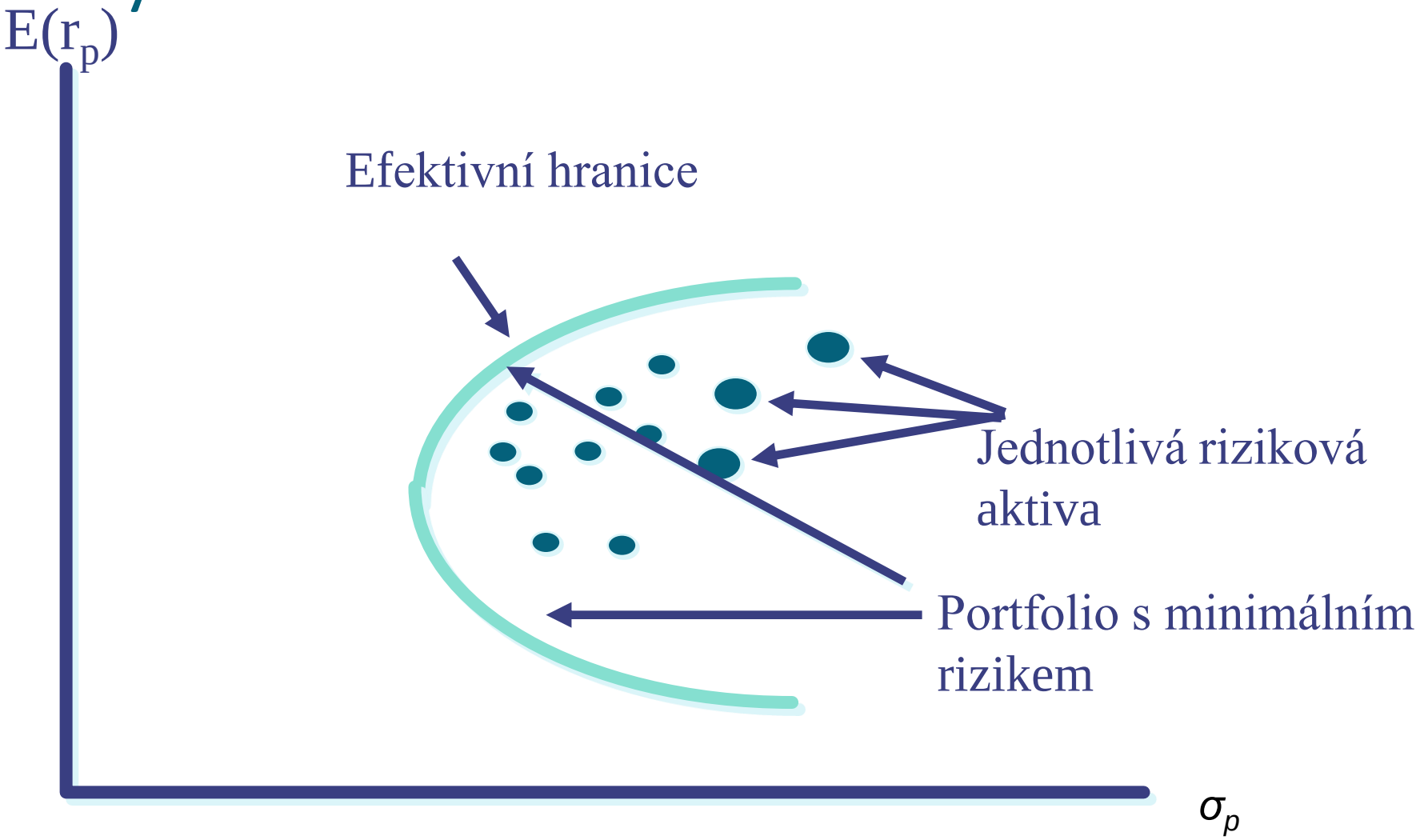
$$R_P = \left(\frac{R_A - R_B}{\sigma_A + \sigma_B}\right)\sigma_P + \left(R_B + \frac{R_A - R_B}{\sigma_A + \sigma_B}\sigma_B\right)$$

$$R_P = -\left(\frac{R_A - R_B}{\sigma_A + \sigma_B}\right)\sigma_P + \left(R_B + \frac{R_A - R_B}{\sigma_A + \sigma_B}\sigma_B\right)$$



Tedy tentokrát vychází rovnice přímek.

Portfolio s minimálním rizikem pro několik rizikových aktiv

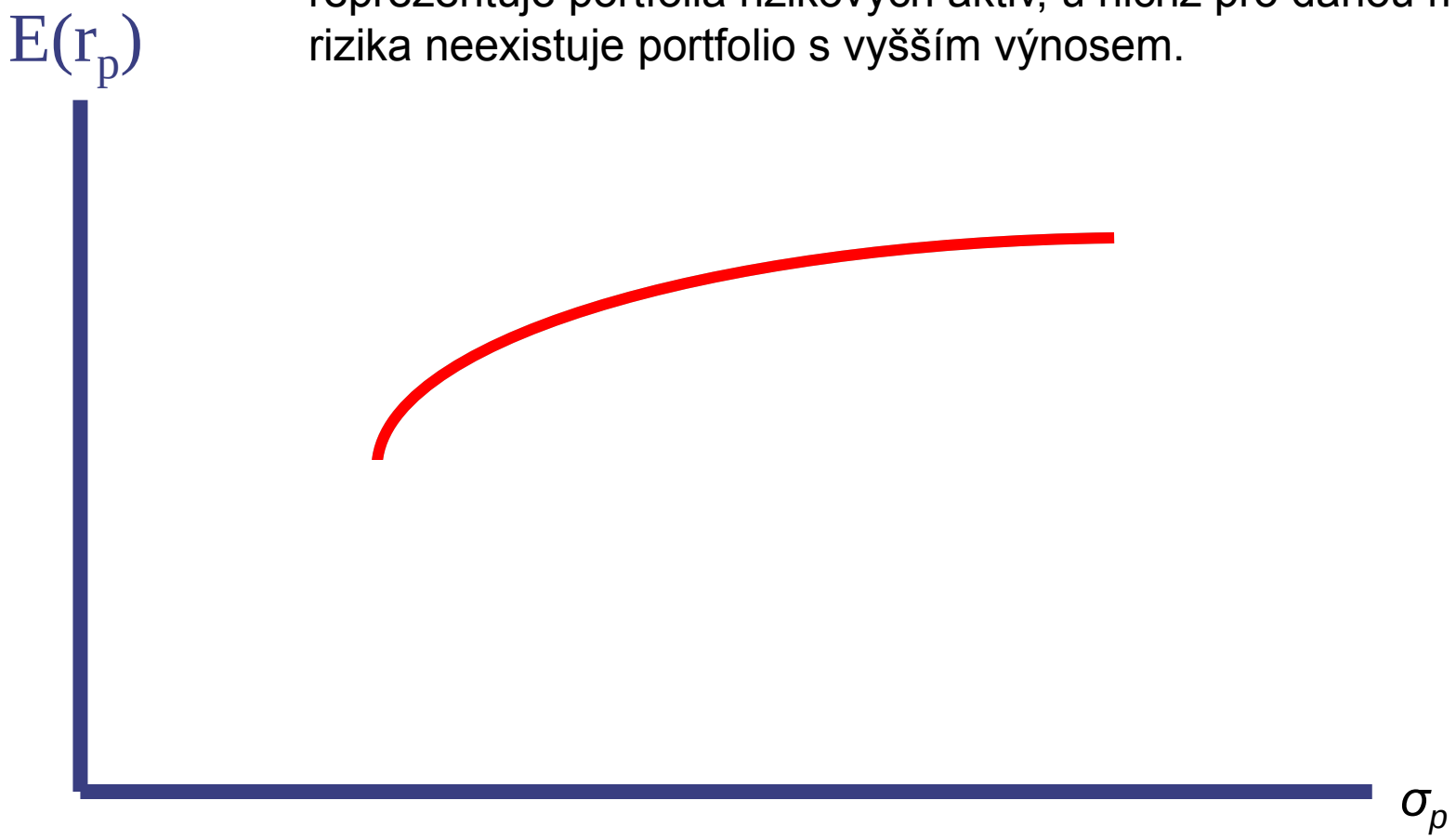


Portfolio s minimálním rizikem

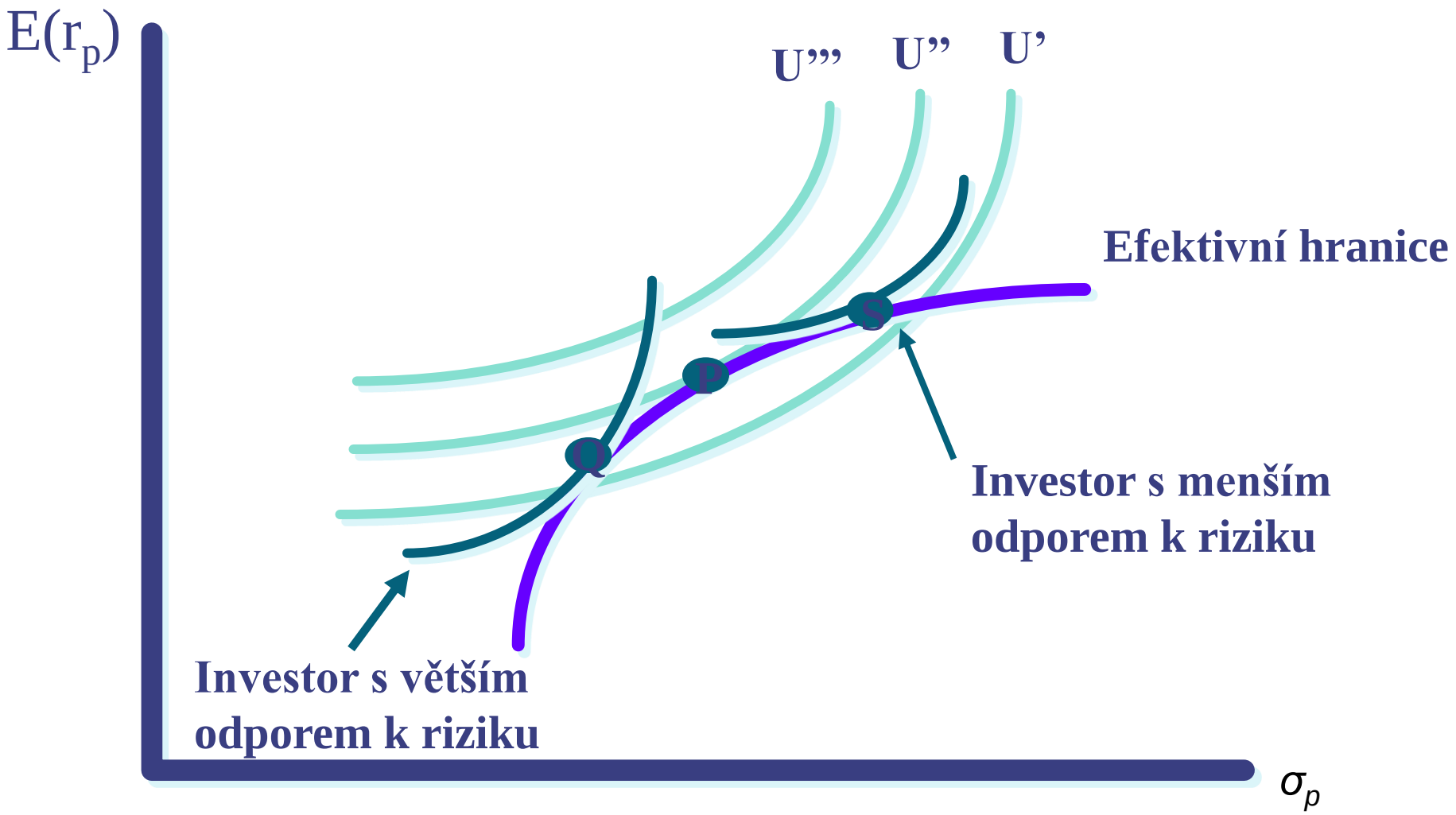
(Min-Variance Opportunity set) – množina bodů v riziko-výnosové rovině, která reprezentuje portfolia rizikových aktiv, u nichž je dosaženo minimálního rizika při dané míře výnosu.

Efektivní hranice (množina)

(Efficient set) – množina bodů v riziko-výnosové rovině, která reprezentuje portfolia rizikových aktiv, u nichž pro danou míru rizika neexistuje portfolio s vyšším výnosem.



Individuální rozhodování pro dvě riziková aktiva, bez bezrizikového aktiva

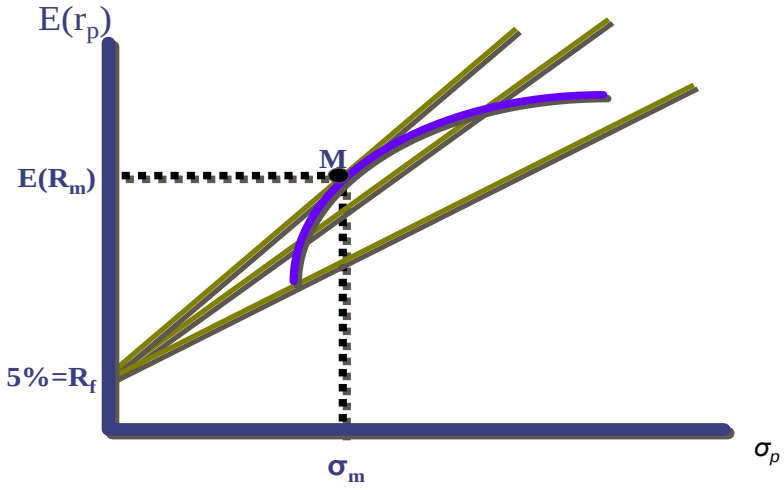


Bezriziková investice (aktiva)

- Předpokládejme, že úroková sazba = zápůjční sazba.
- To znamená, že množina přípustných portfolií může obsahovat libovolnou přímku vycházející z bodu bezrizikového aktiva směřující do libovolného bodu rizikového portfolia s minimálním rizikem.
- Ovšem právě jedna z těchto přímek dominuje všem ostatním.
- Tato dominující přímka prochází bodem bezrizikového aktiva a je tečnou množiny portfolií s minimálním rizikem.
- Tečný bod = portfolio M (the market)

Efektivní hranice portfolia s bezrizikovou investicí

R_A...očekávaný výnos portfolia A
R_F...očekávaný výnos bezrizikové investice B
σ_A ...rozptyl výnosu portfolia A
$\sigma_F = 0$...rozptyl výnosu investice B



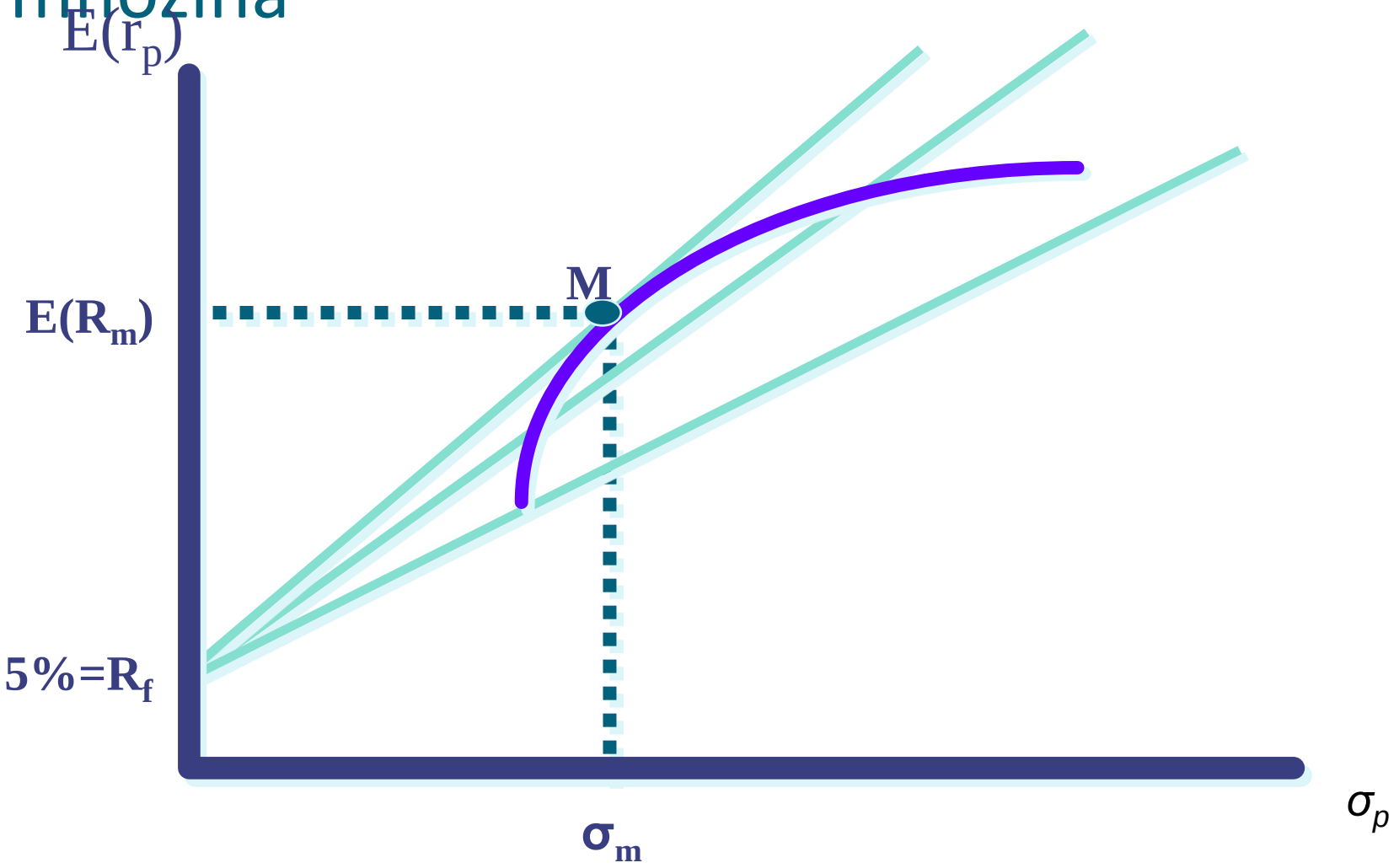
$$R_P = (1-X)R_F + XR_A$$

$$\sigma_P = \left[(1-X)^2\sigma_F^2 + X^2\sigma_A^2 + 2X(1-X)\sigma_A\sigma_F\rho_{FA}\right]^{1/2}$$

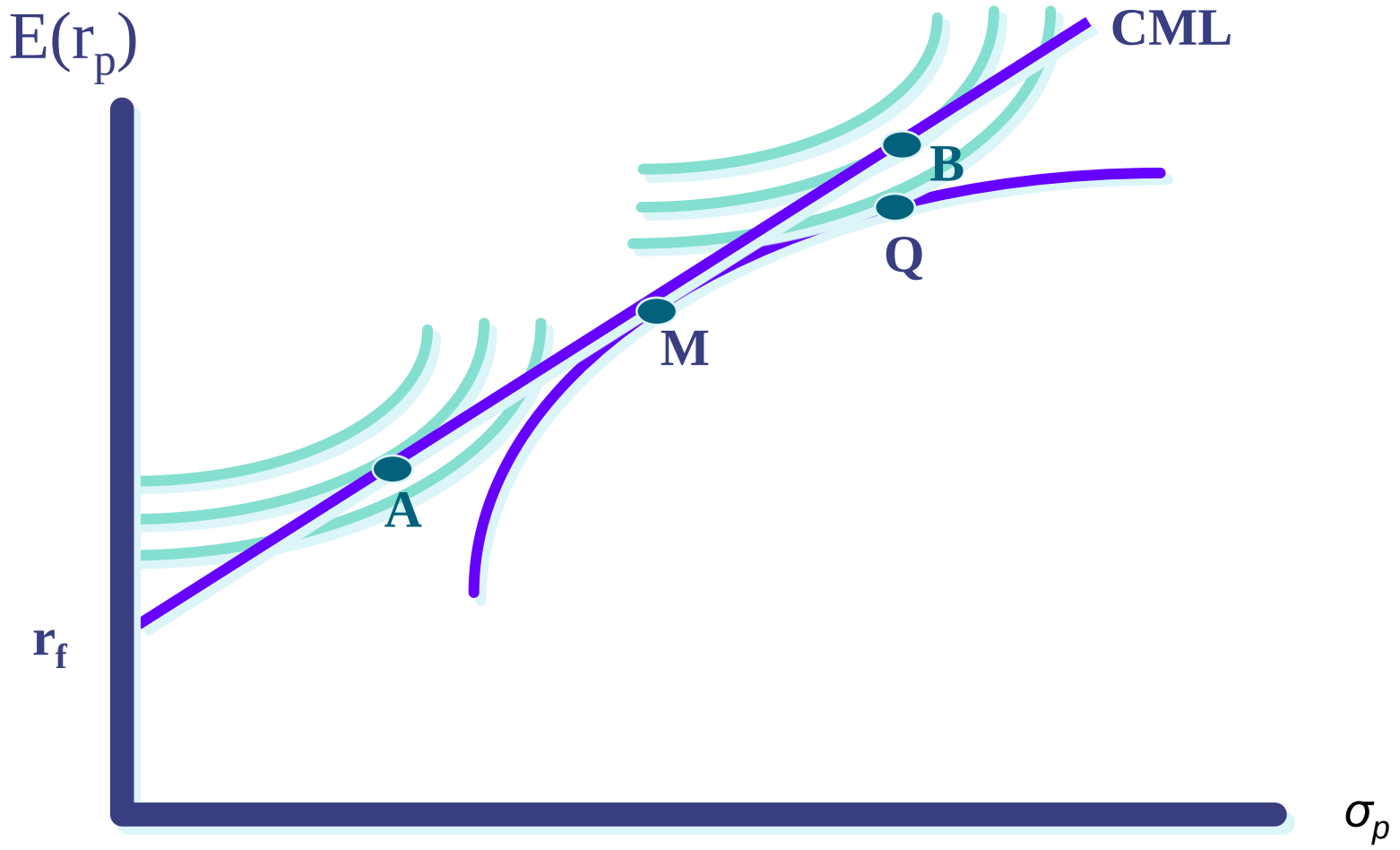
$$\sigma_P = X\sigma_A \qquad X = \frac{\sigma_P}{\sigma_A}$$

$$R_P = R_F + \left(\frac{R_A - R_F}{\sigma_A}\right)\sigma_P$$

CML (Capital market line) = lineární efektivní množina



Individuální rozhodování pro dvě riziková aktiva, s bezrizikovým aktivem



Stanovení optimálního portfolia- Markowitzův model

Markowitzův model je jedním z přístupů, jak hledat optimalní portfolio. Tento model předpokladá, že je investor racionalní, tedy jeho cílem je maximalizovat zisk a minimalizovat riziko. Ziskem se v Markowitzově modelu rozumí střední hodnota náhodného výnosu a rizikem pak jeho směrodatná odchylka. Tento model má řadu zjednodušujících předpokladů:

- předpokládá ideální trh bez transakčních nákladů a bez arbitráže, neomezenou možnost investování a půjčovaní, neomezenou dělitelnost aktiv, předpokládá, že investoři preferují vyšší výnosy a nižší riziko a využívají k tomu shodné informace - hodnoty očekávaných výnosností akcií a rozptylů a kovariancí těchto výnosností.

Označení

I	počet akcií, z nichž skládáme portfolio,
x_i	váha i-té akcie v portfoliu, $i = 1, \ldots, I$
x_0	váha bezrizikového aktiva v portfoliu,
ρ_i	náhodný výnos i-té akcie ve zvoleném období, $i = 1, \ldots, I$
r_i	očekávaný výnos i-té akcie ve zvoleném období, $i = 1, \ldots, I$
r_p	minimální požadovaný výnos portfolia ve zvoleném období,
r_0	výnos bezrizikového aktiva ve zvoleném období.

Vektor vah označíme $\mathrm{x} = (x_1, \ldots, x_I)'$, vektor náhodných výnosů $\rho = (\rho_1, \ldots, \rho_I)'$. Rozdělení náhodného vektoru ρ je charakterizováno známým vektorem středních hodnot $E(\rho) = \mathrm{r} = (r_1, \ldots, r_I)'$ a varianční maticí $\mathrm{var}(\rho) = \mathbf{V} = [\mathrm{cov}(\rho_i, \rho_j)]_{i,j=1}^{I}$.

Výnos portfolia s vahami x budeme chápat jako střední hodnotu celkové výnosnosti

$$r(\mathrm{x}) = \sum_{i=1}^{I} x_i r_i = \mathrm{r}'\mathrm{x}$$

a **riziko** tohoto portfolia chápeme ve smyslu Markowitzova modelu jakožto směrodatnou odchylku celkové výnosnosti - odmocninu z rozptylu

$$\sigma^2(\mathrm{x}) = \sum_{i,j=1}^{I} x_i x_j V_{ij} = \mathrm{x}'\mathbf{V}\mathrm{x}.$$

Optimalizační úloha

Při hledání optimálního portfolia ve smyslu Markowitzova modelu pak musíme řešit optimalizační úlohu vícekriteriálního programování

$$\begin{aligned} &\max\ \mathbf{r}'\mathbf{x} \\ &\min\ \mathbf{x}'\mathbf{V}\mathbf{x} \\ &\text{za podmínek} \\ &\quad \mathbf{x} \in \chi, \end{aligned} \tag{1}$$

kde množina χ je určena požadavkem $\mathbf{1}'\mathbf{x} = 1$ a případně dalšími podmínkami na složení portfolia.

Jinou možností je řešit zjednodušenou úlohu nelineárního programování ve tvaru

$$\max_{\mathbf{x}\in\chi}\ \lambda\mathbf{r}'\mathbf{x} - \frac{1}{2}\mathbf{x}'\mathbf{V}\mathbf{x}, \tag{2}$$

kde $\lambda \geq 0$ je parametr modelující investorův vztah k riziku.

Úloha, kterou použijeme při hledání optimálního portfolia my, je následujícího tvaru

$$\begin{aligned} &\min\ \mathbf{x}'\mathbf{V}\mathbf{x} \\ &\text{za podmínek} \\ &\quad \mathbf{x} \in \chi, \\ &\quad \mathbf{r}'\mathbf{x} \geq r_p, \end{aligned} \tag{3}$$

kde r_p je zvolená hodnota minimálního požadovaného výnosu.

Markowitzův model -varianty

a) **Na trhu nejsou povoleny krátké prodeje. Sestavte efektivní hranici portfolií. Vyberte některá portfolia na efektivní hranici a uveďte jejich složení (váhy) a očekávané výnosnosti titulů zastoupených v portfoliu.**

b) **Jak se změní efektivní hranice, pokud budete mít možnost investovat do bezrizikového aktiva (např. depozita v bance).**

c) **Jak se změní efektivní hranice, pokud budete mít možnost výpůjček od správce portfolia až do 30 % hodnoty portfolia.**

d) **Co když budete mít povoleny krátké prodeje až do 30 % počátečního vkladu? Nakreslete efektivní hranici v tomto případě.**

e) **V souladu s vnitřní politikou investiční společnosti, kterou zastupujete, nesmí žádný z titulů portfolia přesáhnout 15% váhu v celkovém portfoliu. Nakreslete efektivní hranici při tomto omezení.**

Stanovení efektivní hranice

Naším úkolem je pro jednotlivé úlohy najít efektivní hranice, tj. množinu portfolií, které jsou **eficientní**. Řekneme, že portfolio je eficientní vzhledem ke střední hodnotě a rozptylu, jestliže neexistuje jiné portfolio, jehož výnos by byl větší nebo roven výnosu uvažovaného portfolia a jehož riziko by bylo menší nebo rovno riziku uvažovaného portfolia, s alespoň jednou nerovností ostrou. Efektivní hranice odpovídá optimálním řešením úlohy(3) pro různé nastavené hodnoty $r_p \geq r_{min}$, kde r_{min} je výnos portfolia $\mathbf{x}_G$, které je optimálním řešením úlohy (4) bez podmínek na očekávanou výnosnost

$$\begin{aligned} &\min\ \mathbf{x}'\mathbf{V}\mathbf{x} \\ &\text{za podmínek} \\ &\quad \mathbf{x} \in \chi. \end{aligned} \tag{4}$$

Formulace úlohy a)

Nejsou povoleny krátké prodeje, to znamená, že váhy jednotlivých akcií v portfoliu musí být nezáporné.

$$\begin{aligned} &\min \sum_{i=1}^{I} \sum_{j=1}^{I} x_i x_j V_{ij} \\ &\text{za podmínek} \\ &\quad \sum_{i=1}^{I} x_i = 1, \\ &\quad x_i \geq 0, \quad i = 1, \ldots, I, \\ &\quad \sum_{i=1}^{I} r_i x_i \geq r_p. \end{aligned} \tag{5}$$

Formulace úlohy b)

Nejsou povoleny krátké prodeje a máme možnost investovat do bezrizikového aktiva. Zavedeme novou proměnnou x_0, která bude vyjadřovat, jakou část investujeme do bezrizikového aktiva. Bezrizikový výnos značíme r_0.

$$\begin{aligned} &\min \textstyle\sum_{i=1}^{I} \sum_{j=1}^{I} x_i x_j V_{ij} \\ &\text{za podmínek} \\ &\quad x_0 + \textstyle\sum_{i=1}^{I} x_i = 1, \\ &\quad x_i \geq 0, \quad i = 0, \ldots, I, \\ &\quad x_0 r_0 + \textstyle\sum_{i=1}^{I} r_i x_i \geq r_p. \end{aligned} \tag{6}$$

Formulace úlohy c)

Nejsou povoleny krátké prodeje, máme možnost investovat do bezrizikového aktiva a máme možnost výpůjček od správce portfolia až do 30 % hodnoty portfolia. Zavedeme novou proměnnou x_v, která bude vyjadřovat velikost půjčky. Proměnná x_v může nabývat hodnot v intervalu $[0, 0.3]$, přičemž $x_v = 0$ pokud si nic nepůjčujeme a $x_v = 0.3$, pokud možnosti půjčky využijeme naplno a půjčíme si celých 30 % hodnoty portfolia. Výpůjční sazbu značíme r_v. Výnos portfolia je pak roven $x_0 r_0 + \sum x_i r_i - x_v r_v$.

$$
\begin{aligned}
&\min \textstyle\sum_{i=1}^{I} \sum_{j=1}^{I} x_i x_j V_{ij} \\
&\text{za podmínek} \\
&\quad x_0 + \textstyle\sum_{i=1}^{I} x_i = 1 + x_v, \\
&\quad x_i \geq 0, \quad i = 0, \ldots, I, \\
&\quad x_v \geq 0, \\
&\quad x_v \leq 0.3, \\
&\quad x_0 r_0 + \textstyle\sum_{i=1}^{I} r_i x_i - x_v r_v \geq r_p.
\end{aligned}
\tag{7}
$$

Formulace úlohy d)

Máme povoleny krátké prodeje, a to až do 30 % počátečního vkladu. To znamená, že váhy x_i mohou být i záporné, ale součet záporných částí vah $\sum_{i=1}^{I} x_i^-$ $(x^- = -\min(0, x))$ nesmí přesáhnout hodnotu 0.3.

$$
\begin{aligned}
&\min \sum_{i=1}^{I} \sum_{j=1}^{I} x_i x_j V_{ij} \\
&\text{za podmínek} \\
&\quad \sum_{i=1}^{I} x_i = 1, \\
&\quad \sum_{i=1}^{I} x_i^- \leq 0.3, \\
&\quad \sum_{i=1}^{I} r_i x_i \geq r_p.
\end{aligned}
\tag{8}
$$

Formulace úlohy e)

Žádný z titulů nesmí přesáhnout 15% váhu v portfoliu. Toto omezení se jednoduše vyjádří tak, že $x_i \leq 0.15$ pro každé i.

$$
\begin{aligned}
&\min \textstyle\sum_{i=1}^{I} \sum_{j=1}^{I} x_i x_j V_{ij} \\
&\text{za podmínek} \\
&\quad \textstyle\sum_{i=1}^{I} x_i = 1, \\
&\quad x_i \geq 0, \quad i = 1, \ldots, I, \\
&\quad x_i \leq 0.15, \quad i = 1, \ldots, I, \\
&\quad \textstyle\sum_{i=1}^{I} r_i x_i \geq r_p.
\end{aligned} \tag{9}
$$

Řešení zmíněných modelů

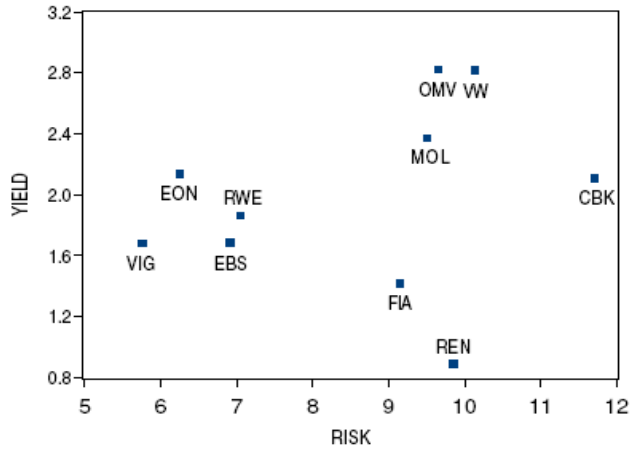
Jde o modely kvadratického programování, které lze řešit např. v programech R, GAMS,…:

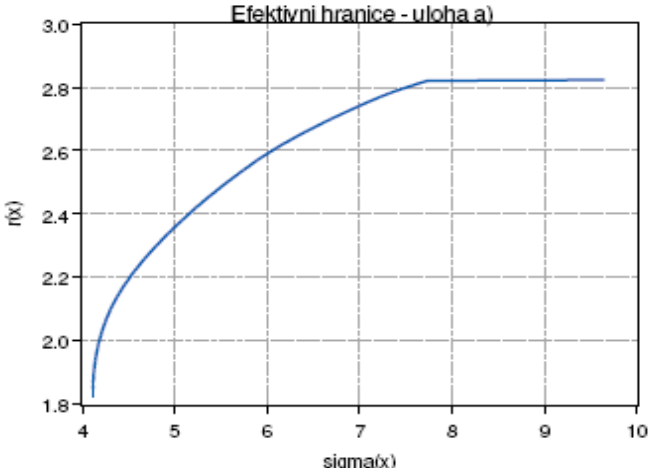
http://www.r-project.org/

- Knihovna fPortfolio:
  http://cran.r-project.org/web/packages/fPortfolio/index.html
- Knihovna Quadprog:
  http://cran.r-project.org/web/packages/quadprog/index.html

http://www.gams.com/

Možné výsledky





parametry	Portfolio
$r\,(\mathbf{x})$	1.9
$\sigma\,(\mathbf{x})$	4.1264

	CBK	EBS	VIG	EON	RWE
Váhy	0.00	0.20	0.35	0.22	0.04
Investované částky	0	5 000 000	8 750 000	5 500 000	1 000 000
	VW	FIA	REN	OMV	MOL
Váhy	0.04	0.05	0.00	0.03	0.07
Investované částky	1 000 000	1 250 000	0	750 000	1 750 000