**Univerzita Hradec Králové**

**Fakulta informatiky a managementu**

BAKALÁŘSKÁ PRÁCE

**2017 Adam Ouhrabka**

**Univerzita Hradec Králové**

**Fakulta informatiky a managementu**

**Katedra Informatiky**

Strojové učení v počítačové grafice

Autor: Adam, Ouhrabka

Aplikovaná informatika

Vedoucí práce: Ing. Bruno Ježek, PhD.

Hradec Králové XXX 2018

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne

# Anotace

# Annotation

# Úvod

1.       Provést literární rešerši a seznámit se s aktuálním poznáním v oblasti strojového učení.

2.       Vytvořit přehled postupů a metod používaných v počítačové grafice pro návrh modelu scény podporovaný metodami strojového učení.

3.       Prozkoumat vybrané dostupné technologie pro implementaci metod strojového učení.

4.       Zvolit vhodný příklad využití a navrhnout jeho softwarovou implementaci.

5.       Pro implementované řešení provést testování a zhodnotit dosažené výsledky.

**Strojové učení**

Z formálních definic strojového učení je vybráno tradiční znění Toma Mitchella, který definuje strojové učení násedujícím způsobem. *„Stroj (počítačový program) se učí plnit třídu zadání T ze zkušenost E a s úspěšností P v tom případě, když se úspěšnost plnění zadaných úkolů T zlepšuje s využitím znalostí nabytých ze* zkušenosti E.“ (Mitchell 1997a). K základním praktickým úlohám, na něž jsou algoritmy strojového učení aplikovány patří klasifikace, predikce, plánování, popřípadě obecná kontrola agenta konkrétního autonomního systému.

Následuje výčet hlavních argumentů pro upřednostňování algoritmů strojového učení, namísto implementace jiného systému. Mitchellem definovaná zkušenost *E* je v oblasti strojového učení reprezentována daty. Pro trénovací sadu je důležité jaké formy nabývá, v jakém pořadí vstupují jednotlivá data do algoritmu, a především „*…jak dobře reprezentuje distribuci příkladů, přes které musí být výsledný systém evaluován úspěšností P.“* (Mitchell 1997b) V takto definované trénovací sadě, lze pak za pomocí vhodných algoritmů strojového učení hledat obecné vztahy a korelace, které by tradiční analýze zůstaly skryté tj. jen těžko by se hledala přiměřená algoritmizovatelná logika. Vzhledem k záměru této práce lze dodat, že tyto vztahy mohou být později užity ke generování nových vzorků zachovávajících obecné charakteristiky a distribuce trénovací sady. Dalšími důvody pro preferenci ASU je dle Nilssona enormní velikost datové sady, která znemožňuje syntézu do tradičního programu. Dále je to dynamická proměnlivost dat, na kterou musí být systém schopný v ideálním případě reagovat. A konečně jsou některé úlohy definovatelné pouze pomocí předkládání četných příkladů. (Nilsson 1998).

## Rozlišení učících algoritmů s učitelem a bez učitele

Modely

## Rozlišení diskriminativních a generativních modelů

Kromě již uvedeného člelení na učení s učitelem a učení bez učitele, lze algoritmy rozdělit rovněž na diskriminativní a generativní. Na rozdíl mezi nimi lze dobře poukázat v aplikaci na klasifikaci, kde obvykle jde o rozdělení datové distribuce do několika oblastí. Zatímco u diskriminativních modelů jde o namapování vstupů na konkrétní výstup, kterým může být konkrétní třída či skalární hodnota, generativní přístup modeluje pravděpodobnostní relace mezi proměnnými daného modelu (Jebara 1996). Pro N proměnných lze v těchto modelech nalézt úplnou sdruženou pravděpodobnost ve formě p(x1,…..xn). Existuje-li vyjádření takové distribuce, lze na základě této formule odvozovat další hodnoty proměnných pomocí bayesovských pravidel. Tomuto procesu se říká inference. Ve vztahu ke klasifikaci je to pak především pravděpodobnost p(y|x), kde *y* je třída objektu vzhledem k rozdělení *x*. (Ng a Jordan nedatováno). Generativní modely existují většinou ve formě grafických modelů, mezi které patří primárně bayesovské sítě se směrovými relacemi mezi proměnnými a Markovovy modely, popřípadě Markovovy nahodilostní pole. (Jebara – google books). Na příkladu klasifikace jednociferných čísel demonstruje Revow zásadní praktický důsledek při použití generativních modelů, totiž ten, že najdeme-li generativní model pro daný systém, našli jsme i model, který je schopen generovat nové vzorky (Michael Revow et al. 1996).

Diskriminativní modely se v jádru svého fungovaní snaží aproximovat ideální hranici, která žádoucím způsobem odděluje vícedimenzionální datovou distribuci. Mezi diskriminativní algoritmy patří typicky neuronové sítě, support vector machines, lineární regresní algoritmy a další (Jebara 1996). Primární užití diskriminativních algoritmů lze vidět v klasifikaci popřípadě regresi.

**Metody strojového učení**

V následujících kapitolách budou postupně představeny konkrétní metody strojového učení. Vzhledem k praktické části práce je důraz kladen na představení generativních popřípadě pravděpodobnostních grafických modelů a těch, které se dle Summervilla mohou uplatnit v rámci PCGML (Summerville et al. 2017).

### Pravděpodobnostní grafické modely

### Markovovy řetězce

### Neuronové sítě

Neuronové sítě jsou jednou z často používaných metod strojového učení. Je to aparát, který má charakter funkce a lze jej popsat jako *masivně paralelní procesor*. Mnohačetným předkládáním *m* vzorů do neuronové sítě je možné adaptovat funkční parametry jednotlivých neuronů [] tak, že vznikne obecně použitý model pro klasifikaci či predikci dat na základě nového vektoru vstupních hodnot. Tento proces adapatace je chápán jako učení neuronové sítě. Implementace tohoto učení stojí na minimalizaci chybové funkce [], která agreguje rozdíl výstupních výsledných hodnot vyprodukovaných neuronovou sítí s výslednými hodnotami pro konkrétní vzor [] (Vladimír Olej a Hájek Petr 2010).

*Rovnice 1 chybová funkce množiny předkládaných vzorů (neuronová síť s dvěmi a více výstupními neurony)*

Přestože existují různé varianty a topologie neuronových sítí, základní principy korelují (Tučková 2003).

Stavebním elementem neuronových sítí je neuron, jež si lze představit jako funkci na jejímž vstupu je vektor hodnot, které vyšly z předcházející vrstvy neuronové sítě (NS) a na jejím výstupu skalární hodnota, jež figuruje jako vstup pro další vrstvy popřípadě jako výstup neuronové sítě. Vstupní vektor je nejprve agregován do skalární hodnoty, která je následně vstupem do aktivační funkce. Ta převede vstupní hodnotu do výstupní hodnoty celého neuronu. Parametr přidává prahovou hodnotu neuronu (bias), která přispívá k aktivizaci neuronu.

*Rovnice 2 agregace vstupnich hodnot neuronu*

Konkrétních aktivačních funkcí existuje celá řada a mají zásadní charakter na fungování NS. Aktivační funkce mohou mít za důsledek lineární i nelineární charakter klasifikačních oblastí vyprodukovaných neuronovou sítí, přičemž však vzhledem ke složitosti uchovávaných znalostí, užívají se především tzv. nelineární aktivační funkce jako je například sigmoidální funkce (Vladimír Olej a Hájek Petr 2010).

*Rovnice 3 výstupní hodnota neuronu při užití funkce sigmoid*

### Učení neuronových sítí

K adaptaci standardních modelů hlubokého učení s učitelem tj. primárně umělých neuronových sítí a jejich variant se používá algoritmus zpětného šíření chyby (Backpropagation algorithm). Ten spočívá v iterační minimalizaci chybové funkce viz. *Rovnice 1,* za pomocí úpravy vah napojených na jednotlivé neurony ve vrstvách sítě. Výsledným vztahem pro korekci všech váh v NS je rovnice, kde je keocifient posunu v rámci gradientu, je směr gradientu a je původní hodota příslušné váhy.

*Rovnice 4 korekce vah v BPA*

Ve chvíli, kdy je v rámci jedné iterace průchodu dat neuronovou sítí spočítána hodnota chybové funkce výstupní vrstvy, je nutné zpětně pro všechny neurony skrytých vrstev vypočítat nakolik váhy, které z nich vedou přispívají ke konkrétní hodnotě celkové chyby *E* výstupní vrstvy.

*Rovnice 5 výpočet chyby neuronu*

### Generativně kontradiktorní sítě

Jsou specifické architektury neuronových sítí, které patří do rodiny generativních modelů a byly vynalezeny Ianem Goodfellowem v roce 2014 (Goodfellow et al. 2014). Sám Goodfellow obecně definuje generativní modely jako *„...modely, které se na základě tréninkové distribuce dat tvořených množinou pdata naučí odhad takové distribuce reprezentovat“*. V konkrétních aplikacích mohou takové modely buď plnit pouze onu reprezentativní funkci a nebo mohou na základě vnitřní struktury modelu generovat nové originální vzorky.

# Strojové učení v počítačové grafice

## Stručné vymezení počítačové grafiky

Terminologicky je počítačová grafika obor, který se zabývá…

## Metody a výsledky strojového učení v počítačové grafice

Metody strojového učení byly v posledních dekádách uplatňovány ve většině oblastech počítačové grafiky zahrnujících vizualizaci, klasifikaci, segmentaci, komprimaci obrazu, a další podoblasti tohoto oboru. Cílem této kapitoly je vytvoření stručného přehledu požívaných metod zejména v podoblasti dvourozměrné grafiky.

Rastrová tj. maticová reprezentace grafiky vstupuje do algoritmů v podobě normalizovaného vektoru popřípadě tensoru.

Za vrchol a jeden z posledních úspěchů v oblasti zpracování obrazu jsou

# Procedurální generování

V rámci počítačové grafiky existují v podstatě dva způsoby pro definici grafického obsahu modelované scény. Ať už uvažujeme dvojrozměrné, popřípadě trojrozměrné zobrazení scény, tak vlastnosti zobrazovaných objektů této scény musí být vždy předem zaneseny v příslušných datových strukturách, které pak figurují v rámci vizualizačního řetězce. Zachycení geometrie a umístění objektu v souřadnicích scény může být definováno buď staticky tj. „ručním“ modelováním objektů anebo algoritmicky, kdy se vzhled a transformace objektů odvíjejí od sekvence příkazů v rámci daného algoritmu. Podobné rozdělení se týká například i textur a dalších elementů, které scénu dotvářejí. Druhý ze zmíněných přístupu lze označit termínem procedurální generování obsahu (PCG).

Zkratka PCG bývá skloňována především v oblasti počítačových her, kde najdou tyto algoritmy široké uplatnění. Tyto algoritmy jsou však použitelné i v rámci budování parametrizovatelných virtuálních prostředí, simulací, vizualizací, popřípadě v oblasti designu. Nutno zmínit, že PCG se ve své definici neomezuje jen na generování grafického obsahu jako takového, nýbrž se tento pojem používá i pro algoritmické generování jakéhokoliv obsahu jako je například obsah textový, animovaný, popřípadě hudební. Yannakis zahrnuje do oblasti PCG i generování personalizovaného obsahu, které se uplatňuje například v rozhraních pro e-commerce a uživatelských rozhraních obecně (Yannakakis a Togelius 2011).

Univerzální vlastností většiny algoritmů pro PCG je, že ve svém těle pracují s faktorem nahodilosti, který je nezbytný pro zajištění diverzity obsahu. Základní metodou pro naplnění tohoto kritéria je například implementace pseudonáhodných čísel v podobě šumu (HENDRIKX a MEIJER nedatováno). Mezi další podstatné vlastnosti PCG algoritmů patří rychlost, spolehlivost, kontrolovatelnost a uvěřitelnost (Julian et al. 2016). Smith ve svém článků užívá v kontextu PCG výstižný pojem „řízená nahodilost“ (Smith nedatováno). Nahodilost má v PCG efekt spíše na dílčí částí negenerovaného obsahu, ale nikoliv už na celek, který musí být uvěřitelný, popřípadě hratelný. Demonstrováno na příkladu: distribuce jednotlivých listů může být náhodná do té míry, do kdy se objekt jeví stále jako objekt dané třídy, v našem případě jako strom. Termínem „řízená“ je vystihnuto kritérium pro smysluplnost, ze kterého vyplývá další podstatná vlastnost PCG algoritmů, kterou je přítomnost určitých omezení (pravidel, constraints), které v jádře algoritmu zajišťují, že obsah bude koherentní.

## Klasifikace PCG algoritmů

Algoritmy PCG se dělí na parametrizovatelné (assisted) a na ty které ke svému fungování žádnou parametrizaci nepotřebují (non-assisted). I minimální parametrizace jako například definice počtu iterací spadá do skupiny parametrizovatelných PCG algoritmů, které nabízejí designérovy kontrolu nad finální podobou generovaného obsahu. (Carli et al. 2011).

Další dělení definující PCG algoritmy se týká toho, zdali má generování zpětnou vazbu na uživatelské interakce nebo takovou vazbu nemá. Popřípadě zdali se obsah generuje při běhu aplikace (online) nebo slouží pouze jako podpůrný nástroj designu při vývoji (offline) (Togelius et al. 2010).

Jedná-li se o algoritmus, který přímo reaguje na interakci s uživatelem a v rámci běhu aplikace dotváří „personalizovaný obsah“, označuje ho Yannakakis zkratkou EDPCG (Experience-drive procedural content generation).

## Tradiční metody procedurálního generování

Tradiční PCG algoritmy jsou založeny na poznatcích z mnoha oblastí informatiky. Pro generování grafického obsahu se užívají evoluční algoritmy, Lindenmayerovi systémy, fraktály, a další specifickým způsobem aplikované postupy, které pro každý svůj běh negenerují jiný, přesto smysluplný a monotematický obsah. K tomu lze podotknout, že „Všechny tyto metody mají společné to, že tyto algoritmy, parametry, omezení a cíle, které se podílí na vytváření obsahu jsou v podstatě ručně definované svými tvůrci“ (Summerville et al. 2017)

Stěžejní proměnou PCG algoritmů je podoba reprezentace obecných genotypů, na základě kterých vzniká vždy originální entita (fenotyp). Tato reprezentace nabývá mnoha podob. Takové genotypy mohou reprezentovat jak obecné geometrie objektů, tak například obecné vztahy mezi nimi. Tato biologická paralela vychází z často aplikované metody PCG, kterou jsou genetické algoritmy.

## Potenciál, účel a vize PCG

Klíčovou součástí moderních vizualizací, popřípadě počítačových her jsou namodelované objekty plnící buď kosmetickou či interaktivní funkcí. Nevyužije-li se v rámci produkce PCG metod je všechen obsah modelován ručně. Taková praxe výrazně zvyšuje cenu produktu a prodlužuje dobu jeho vývoje. Na druhé straně je možné potenciál PCG použít pouze tam, kde prvek prvek náhody dotváří funkci herního mechanismu nebo vytváří diverzitu, která nenarušuje účel obsahu (nebrání například v postupu). Generovaný obsah tedy může být na základě tohoto kritéria rozdělen na funkční (necessary) a kosmetický (optional content)(Togelius et al. 2010).

Vize a cíle současného bádaní PCG v grafické oblasti definuje ve svém článku Julian Togelius. Jako první vize je nadneseně představena univerzální metoda na vytvoření komplexních virtuálních světů určitého žánru. I v případě, že by se jednalo pouze o statický grafický obsah takového světa dá se společně s Togeliusem konstatovat, že podobný algoritmus je nerealizovatelný i v rámci vzdálené budoucnosti. Komplexita takového úkolu je enormní. Problémů, které stojí v cestě takovému zadání je hned několik a tvoří primární úkoly v rámci výzkumu PCG.

Generovaný obsah se potýká s originalitou, nedostatečnou diverzitou a smysluplností. To vše jsou znaky kreativity v rámci ručního modelování. Klíčová otázka zní, zdali se v případě PCG dá kreativita algoritmizovat a v případě generování prostřednictvím strojového učení stochasticky modelovat i v rámci komplexních celků. Kritéria výpočetní kreativity, která významě korelují s požadavky na PCG definoval ve své práci Pereira. Jako hlavní znaky vidí: uchování modelu znalostí o dané třídě, schopnost redistribuce znalosti do nových spojení, schopnoost kriticky zhodnotit produkt (validace), uvážení kontextu do kterého je tvorba zasazena a užití konvergentních i divergentních postupů při kreaci (Pereira 2008). Všechny tyto podmínky, které Pereira postuluje pro svůj model (Creative general problem solver) jsou realizovatelné a byly realizovány prostřednictvím generativních modelů strojového učení například v rámci Generative adversarial neural networks.

Togelius ve svém článku nadále vybízí k syntéze metod, které byly vynalezeny v rámci generování jednotlivých objektů jako například terénu, budov či interiéru, do komplexního parametrizovatelného generátoru. Interakci mezi jednotlivými generátory podtrhuje ve svém článku i Hendrikx (HENDRIKX a MEIJER nedatováno). Výstupy jednotlivých generátorů by tedy měly být vzájemně kompatibilní a ve vztahu, který dodává generovanému celku přidanou hodnotu.

Tradiční metody PCG jsou často aplikovány ad hoc ke konkrétnímu použití. Znovupoužitelnost je realizovatelná pouze principiálně a jen stěží se dají stejné algoritmy použít napříč různými aplikacemi. I to je nedostatek na který Togelius poukazuje (Togelius et al. 2013).

Uvážíme-li Togeliusův nárok na komplexní procedurální generátor, je nutné do něj zahrnout i dynamickou stránku scénu tj. animační. Procedurální animování je další oblast, v rámci které probíhá aktivní výzkum (Togelius et al. 2013)

Jako obecný trend v rámci PCG lze tedy primárně identifikovat redukci genericity generovaného obsahu a směřování k univerzálním navzájem propojeným generátorům obsahů komplexních. K naplnění těchto cílů se výzkumníci v oblasti PCG často uchylují k metodám strojového učení, jež má ve v návaznosti na současné cíle PCG velký potenciál, neboť umožňuje modelování složitých funkcí a datových distribucí tj. uchování znalosti o modelu a jeho redistribuci do nových celků.

## Nové metody procedurálního generování

Pro uvedení terminologie užívané v následujících kapitolách je nutné zmínit, že současnými cestami pro tvorbu algoritmicky generovaného obsahu je generování za pomocí metod strojového učení. Tradiční PCG přistupuje k tvorbě obsahu s konkrétní referencí vzhledem k cíli. Tato reference však není součástí samotného algoritmu, ale figuruje pouze jako inspirace toho, kdo algoritmus navrhuje. Znamená to, že se v rámci algoritmu hledají pravidla a omezení, která pomohou aproximovat žádoucí obsah. Na rozdíl od tohoto přístupu procedurální generování prostřednictvím strojového učení (PCGML) vezme existující strukturu a na základě zpracování a „pochopení“ daného obsahu vytvoří model, pomocí kterého nageneruje další diverzifikovaný obsah. Tak poprvé definuje PCGML Summerville (Summerville et al. 2017).

Širší pojednání o těchto metodách je předmětem kapitoly Procedurální generování prostřednictvím strojového učení.

# Procedurální generování prostřednictvím strojového učení

Definice strojového učení od Toma Mitchella uvedená v úvodní kapitole ke strojovému učení je aplikovatelná rovněž pro PCGML, kdy zadáním *T* je generování konkrétní třídy obsahu na základě zkušenosti *E*, která je tvořena množinou ručně modelovaných obsahů (v případě učení s učitelem). Měření úspěšnost *P* může být v této konkrétní aplikaci strojového učení značně komplikované a může v konečné fázi spočívat v subjektivním posouzení omezeného množství nagenerovaného obsahu. Příkladem implementace takové metriky jsou dotazníky vyplněné testery, na jejichž základě byly evaluovány hratelnosti procedurálně nagenerovaných levelů a tedy celého algoritmu. (Roberts a Chen 2015). Pro posouzení většího vzorku dat je třeba implementace algoritmů, jejichž funkcí je evaluace generovaného obsahu na základě příslušných kritérií.

Ne všechny metody strojového učení, které se jinak výborně hodí pro klasifikaci či predikci, jsou vhodné pro generování nového obsahu. Odpovídá tomu i rozdělení metod strojového učení na diskriminativní a generativní modely. Z generativních technik strojového učení vyzdvihuje Summerville jako vhodné zejména generativně kontradiktorní sítě (Generative adversarial networks) dále n-gramy, markovy modely a specifické architektury rekurentních neuronových sítí (Summerville et al. 2017).

## Metody strojového učení v aplikaci na procedurální generování

# Technologie pro implementaci algoritmů strojového učení

# Nápady

Srovnat algoritmy tradičního PCG s PCGML

Vymyslet exemplární příklad a vyřešit ho třemi různými metodami strojového učení. Srovnat porovnat => grafíky.

# Demonstrace metod strojového učení na příkladech generování

Konkrétně specifikovaný cíl této kapitoly a bakalářské práce vůbec je demonstrovat a evaluovat navrženou metodu pro generování scény v diskrétním dvojrozměrném prostoru.

## Užité technologie

Veškeré implementace, které jsou zde demonstrovány byly implementovány v jazyce Python ve verzi 3.6. Argumentem pro užití této technologie je především existence použitých knihoven pro rychlou práci s neuronovými sítěmi (tensorflow a na něj navázaný keras). Exemplární příklady [x] a nagenerované výstupy [y] byly pro názornost modelovány a uspořádány v herním enginu Unity, který pro své skripty užívá C#. Vzhledem k přímé nekompatibilitě programovacích jazyků byla všechna data, která bylo třeba zpracovat či exportovat formátována v csv souborech.

## Definice cíle a východisek

Tato kapitola představuje předpoklady, východiska, a užité abstrakce, které byly v rámci prezentované metody aplikovány. Nutno už úvodem podotknout, že navržený algoritmus slouží především jako demonstrace použitelnosti či nepoužitelnosti navrženého principu. Objektivně jsou v závěru práce zkonstatovány jeho ... i nedostatky. Reálná aplikace algoritmu je podmíněna jeho dalším rozpracováním.

Na nejvyšší úrovni jsou vstupní data pro algoritmus definována jako scény *S* = {s1, s2, s3...sn}. Scéna je pro nás diskrétním prostorem n2 možných pozic [x0...xn, z0...zn] uspořádaných v matici M. Na těchto pozicích se mohou vyskytovat objekty. V případě, že je pozice prázdná je tato skutečnost reprezentována nulou.

Rovnice Příklad diskrétního prostoru 32, kde xij > 0 představuje jednu konkrétní třídu objektů

Dohromady vytvářejí tyto objekty určitou množinu *O*. Tato množina vytváří smysluplnost tj. scénický význam *SV* ve dvou aspektech. Za prvé je pro *SV* podstatná samotná přítomnost nebo nepřítomnost objektů ve scéně. Například vyskytuje-li se ve scéně 5 postelí je její význam jiný než vyskytuje-li se v ní 5 krabic s dynamitem. Za druhé je význam scény v menší míře  spoluvytvářen i vzájemnými pozičními (a jinými) relacemi v diskrétním prostoru scény. Ukázáno na příkladu: Nachází-li se ve scéně 5 židlí a 5 stolů, je význam scény jiný, když se všech pět židlí nachází u stolu, než když tomu tak není. Oba tyto aspekty jsou v algoritmu uvažovány a jsou více rozvedeny v následujících kapitolách.

Každý objekt přítomný ve scéně sebou nese vektor své pozice, rotace a velikosti. První abstrakcí prezentovaného modelu je, že z těchto tří vlastností objektu je pro zjednodušení úlohy uvažován pouze poziční dvoudimenzionální vektor P = [x, z]. Vektory rotace a velikosti uvažovány nejsou. Kromě geometrických vlastností objektu, je každý objekt kategorizován danou třídou kterou představuje. To umožňuje udržet identitu kategorie přítomných objektů napříč jednotlivými scénami.

Z uvedného již lze názorně definovat nejprve obecný cíl a poté jednotlivé kroky, které povedou k jeho naplnění.

Obecný cíl je definován následovně: Za předpokladu výchozí scény *s0*, která je definována množinou objektů *O*, doplnit tuto množinu objekty, které svou přítomností a pozicí podpoří scénický význam *s0*. Tím vzniká nová scéna *s*. Tento proces bude iterativní, to znamená, že v jednom kroku generovacího algoritmu bude scéna doplněna právě o jeden objekt. Načež se tato nová scéna stane východiskem pro další krok generování.

Rovnice vyjadřuje proces generování

Rovnice sedm naznačuje proces generování výsledné scény *s*, kde *s0* odpovídá výchozí scéně, *n* vyjadřuje počet iterací algoritmu a *ox*představuje predikovaný objekt třídy *x*.

## Slepé uličky?