UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO
FAKULTETA ZA MATEMATIKO IN FIZIKO

Tadej Janež

AKTIVNO UČENJE S PLANIRANJEM EKSPERIMENTOV

Diplomska naloga
na univerzitetnem študiju

Mentor: akad. prof. dr. Ivan Bratko

Ljubljana, 2008

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Namesto te strani **vstavite** original izdane teme diplomskega dela s podpisom mentorja in dekana ter žigom fakultete, ki ga diplomant dvigne v študentskem referatu, preden odda izdelek v vezavo!

Zahvala

Zahvaljujem se mentorju akad. prof. dr. Ivanu Bratku za nasvete in pomoč pri izdelavi diplomske naloge, izr. prof. dr. Blažu Zupanu za pomoč pri spoznavanju metod aktivnega učenja, Juretu Žabkarju za razlago robotske domene in Ashoku Mohanu za pomoč pri implementaciji robotske domene.

Moji družini, ki me je podpirala in prenašala.

Kazalo

Seznam uporabljenih kratic in simbolov

- AUC. Ploščina pod ROC krivuljo (area under curve).
- CA. Klasifikacijska točnost (classification accuracy).
- H. Shannonova entropija.
- ROC. Sprejemnikova operativna značilnost (receiver operating characteristic).
- XPERO. Ime raziskovalnega projekta 6. okvirnega programa Evropske komisije, katerega moto je: "Učenje z eksperimentiranjem".

Povzetek

Na področju strojnega učenja se čedalje bolj uveljavlja področje aktivnega učenja, še zlasti pri problemih, pri katerih imamo na voljo ogromno neoznačenih primerov, označevanje le-teh pa je drago ali časovno zamudno. Takrat lahko uporabimo metode aktivnega učenja, ki skušajo zgraditi čim boljši napovedni model na čim manj označenih primerih.

Novost predstavlja uporaba aktivnega učenja v časovno in prostorsko vezanih domenah. Primer take domene je samoučeči robot, ki izvaja poskuse za gradnjo svojega modela sveta. Pri tem mora upoštevati fizične omejitve glede izbiranja novih učnih primerov.

Po pregledu standardnih metod aktivnega učenja sledi njihovo ovrednotenje na različnih domenah. Za tem je predstavljeno kvalitativno modeliranje v robotski domeni, pri čemer se robot svojega modela uči z eksperimenti. Nato sledi opis adaptacije metod aktivnega učenja za planiranje poskusov. Njihov namen je, da robotu omogočijo planiranje akcij, ki ga bodo hitreje popeljale do boljšega modela sveta. Eksperimentalno ovrednotenje predlaganih metod je bilo narejeno v preprostem robotovem svetu, ki je vseboval le robota in žogo. Pokazalo se je, da planiranje eksperimentov lahko pomaga pri hitrejši gradnji boljšega robotovega modela sveta. Vendar se tudi standardna metoda aktivnega učenja, ki ne uporablja planiranja, zelo dobro odreže na izbrani testni množici primerov. Na koncu so podane možnosti za razvoj metod aktivnega učenja s planiranjem eksperimentov v prihodnje.

Ključne besede:

aktivno učenje, kvalitativno modeliranje, planiranje poskusov, robotsko učenje, kvalitativno učenje

Abstract

In machine learning, active learning is becoming increasingly more widely used, especially for types of problems, where we have an enormous amount of unlabeled examples and their labeling is either expensive or time consuming. In such cases, we can use active learning methods that try to build a good prediction model from as few labeled examples as possible.

A new contribution is the use of active learning in time-space bounded domains. An example of such domain is an autonomous learning robot that makes experiments to build its own model of the world. In order to do that, it has to consider the physical restrictions when choosing new learning examples.

After an overview of standard active learning methods comes their evaluation on various experimental domains. Then I explain qualitative modeling in a robotic domain, where robot learns its model by experimentation. After that comes a description of adaptation of active learning methods for planning of experiments. Their aim is to give robot an ability of planning its actions for quicker building of better models of the robotic world. Experimental evaluation of proposed methods was conducted in a simple robotic world, which contained only a robot and a ball. It turned out that planing of experiments can help with quicker building of better models of the robotic world. However, a standard active learning method that does not use planning, performed really well on selected set of testing examples. Finally, I describe the possibilities of future development of proposed methods.

Key words:

active learning, qualitative modeling, planning of experiments, robotic learning, qualitative learning

Poglavje 1

Uvod

Aktivno učenje je področje strojnega učenja, ki se ukvarja s problemi, pri katerih imamo na voljo ogromno neoznačenih primerov, njihovo označevanje pa je bodisi drago bodisi traja veliko časa (ali pa celo oboje). Zato želimo uporabiti "aktivno" metodo, ki na vsakem koraku na podlagi trenutno označenih primerov in klasifikatorja, naučenega iz njih, čim bolje izbira nove primere za označevanje. Kriterij, po katerem ocenjujemo metode aktivnega učenja, zato ni končna klasifikacijska točnost ali AUC, temveč krivulja naraščanja le-te. Čim manj iteracij metoda potrebuje za dosego določene klasifikacijske točnosti ali AUC-ja, tem boljša je.

Ideja o avtonomnih robotih, ki so se zmožni učiti sami, brez človeškega posredovanja, je eden temeljnih ciljev umetne inteligence. Ena od mnogih paradigem učenja je tudi učenje s poskusi, pri kateri učenec ne potrebuje učitelja, ker se uči avtonomno, v interakciji z resničnim svetom. V našem primeru bi radi zgradili robota, ki se bo sam naučil kvalitativnega modela svojega okolja z izvajanjem akcij in merjenjem njihovih učinkov. Robot ne bo imel nobenega predznanja o svojem okolju. Naša naloga je izdelati čim boljšo strategijo izvajanja akcij in planiranja poskusov, ki bo robotu omogočila čim hitreje zgraditi čim boljši model svojega okolja.

Za strategijo izvajanja akcij bi radi uporabili metode aktivnega učenja. Vendar v osnovni verziji aktivnega učenja ni nobenih omejitev glede izbire novih primerov. Naša naloga je torej izdelati posplošeno metodo za aktivno učenje, ki upošteva realne fizične omejitve glede izbiranja novih učnih primerov. Takšne omejive pa moramo upoštevati, ko robot dejansko izvaja poskuse za pridobivanje novih primerov.

V naslednjem poglavju sledi opis standardnih metod aktivnega učenja in njihova primerjava na različnih naborih podatkov. Nato je podano kvalita-

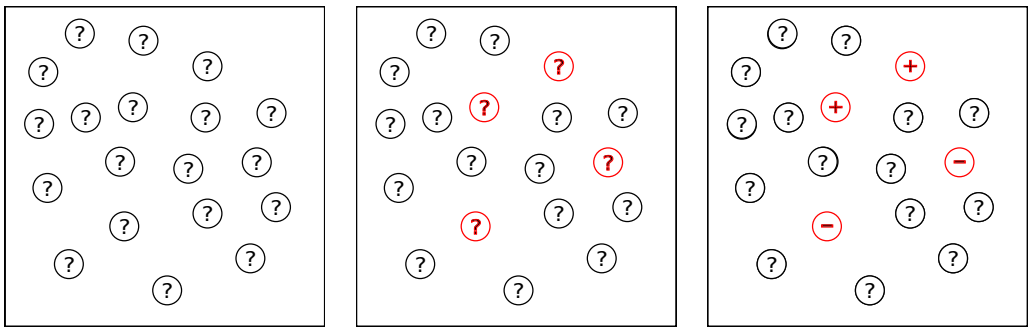
tivno modeliranje v izbrani robotski domeni XPERO in opis adaptacije metod aktivnega učenja za planiranje eksperimentov. Sledijo eksperimentalni rezultati testiranja metod aktivnega učenja v robotski domeni in njihova analiza, na koncu pa še sklepne ugotovitve o prednostih in slabostih predlaganih metod ter možnosti za njihov razvoj v prihodnje.

Poglavje 2

Metode aktivnega učenja

Osnovni princip delovanja aktivnega učenja je prikazan na sliki 2.1.

Metode aktivnega učenja za atributne probleme učenja imajo skupno ogrodje, ki je s psevdokodo opisano na sliki 2.2. Primer X, predstavljen z atributi $x_1, \ldots, x_n$, pripada razredu Y. M je klasifikator, dobljen s tradicionalno metodo strojnega učenja (npr. klasifikacijsko drevo, naivni Bayesov klasifikator, itd.). Na začetku so vsi primeri neoznačeni in pripadajo množici U. Prva while zanka napolni učno množico z *init_size* naključno izbranimi primeri in služi za inicializacijo algoritma. Druga while zanka pa z dano metodo *chooser* $(T - init\_size)$-krat pametno izbira nove primere, za katere od učitelja dobi oznako in jih doda v učno množico. Na vsakem koraku se tudi zgradi klasifi-



(a) Na začeku so vsi primeri neoznačeni.

(b) Metoda na vsakem koraku izbere nove primere, ki jih bo dodala v učno množico.

(c) Za oznake izbranih primerov na vsakem koraku vpraša učitelja.

Slika 2.1: Osnovni princip delovanja aktivnega učenja.

Input: T (total number of feedback iterations), U (pool of unlabeled instances), $init\_size$ (number of random feedback iterations)
Output: M_T (Model)
Algorithm:

$t = 0; U_0 = U; M_0 = NULL;$
while $t < init\_size$ **do**
  $(X_t, U_t) = random\_chooser(M_0, U_{t-1})$
  teacher assigns label Y_t to X_t
  $M_t = train\_classifier(\{(X_i, Y_i)|i = 1 \dots t\}, M_{t-1})$
  $t++$
end while
while $t < T$ **do**
  $(X_t, U_t) = chooser(M_{t-1}, U_{t-1})$
  teacher assigns label Y_t to X_t
  $M_t = train\_classifier(\{(X_i, Y_i)|i = 1 \dots t\}, M_{t-1})$
  $t++$
end while

Slika 2.2: Psevdokoda algoritma aktivnega učenja.

kator M_t na trenutni množici označenih primerov $\{(X_i, Y_i)|i = 1 \dots t\}$, ki se uporabi za ocenjevanje metode aktivnega učenja.

2.1 Izbiranje po neodločenosti

Ta metoda (angl. *uncertainty sampling*) je bila prvič opisana v članku [5] in temelji na intuitivni ideji, da je najbolje izbrati primere, za katere je trenutni klasifikator najmanj gotov pri napovedi razreda.

Pri tej metodi na vsakem koraku zgradimo klasifikator M_t na do sedaj označenih primerih in za vse neoznačene primere $X \in U_t$ z njim napovemo verjetnosti razreda Y. Izberemo tisti primer, za katerega velja:

$$\arg \min_{X \in U_t} \max_{y \in Y} P_{M_t}(Y = y|X).$$

Na vsaki iteraciji torej izberemo primer, za katerega klasifikator vrne najbolj neodločeno oceno verjetnosti razreda.

2.2 Zmanjševanje prihodnje napake

Roy in McCallum sta to metodo (angl. *Future error reduction*) predstavila v članku [6]. Ideja metode je v tem, da izbere tisti primer, ki bo najbolj zmanjšal pričakovano napako na testni množici, potem ko ga bomo dodali v učno množico.

Naj bo $P(Y|X)$ neznana pogojna distribucija čez vse vhode X in razrede $y \in Y$ in D množica označenih primerov. Če je $\hat{P}_D(Y = y|X)$ napoved klasifikatorja, zgrajenega na D, za pripadnost primera X razredu y, potem lahko pričakovano napako klasifikatorja ocenimo kot:

$$E_{\hat{P}_D} = \frac{1}{|U|} \sum_{X \in U} L(P(Y = y|X), \hat{P}_D(Y = y|X)),$$

kjer je L funkcija, ki meri ceno napačne napovedi distribucije razreda med klasifikatorjevo napovedjo $\hat{P}_D(Y = y|X)$ in pravo vrednostjo $P(Y = y|X)$.

Če za L vzamemo logaritemsko loss funkcijo: $L = \sum_{y \in Y} P(Y = y|X) \log \hat{P}_D(Y = y|X)$, dobimo:

$$E_{\hat{P}_D} = \frac{1}{|U|} \sum_{X \in U} \sum_{y \in Y} P(Y = y|X) \log \hat{P}_D(Y = y|X).$$

Prave distribucije $P(Y|X)$ ne poznamo, zato jo ocenimo s trenutnim klasifikatorjem, naučenim na množici D. Tako dobimo oceno za pričakovano napako:

$$\tilde{E}_{\hat{P}_D} = \frac{1}{|U|} \sum_{X \in U} \sum_{y \in Y} \hat{P}_D(Y = y|X) \log \hat{P}_D(Y = y|X).$$

Če hočemo zmanjšati varianco te ocene, lahko naredimo več učnih množic z bagging metodo, zgradimo klasifikator na vsaki izmed njih in na koncu povprečimo dobljeno napoved razreda.

Opazimo lahko, da je to v resnici povprečna entropija razredne spremenljivke neoznačenega primera:

$$\tilde{E}_{\hat{P}_D} = \frac{1}{|U|} \sum_{x \in U} H(\hat{P}_D(Y|X))).$$

Algoritem na vsakem koraku izbere tisti $X^* \in U_t$, za katerega velja:

$$\forall (X, Y) : \tilde{E}_{\hat{P}_{D+(X^*,Y^*)}} < \tilde{E}_{\hat{P}_{D+(X,Y)}}.$$

2.3 Ostale metode

Na področju aktivnega učenja obstaja še nekaj standarnih metod, ki jih nisem podrobneje opisal.

Prva je Zmanjševanje prostora hipotez (angl. *Version space reduction*), opisana v [2]. Pri tej metodi na vsakem koraku zgradimo dva klasifikatorja tako, da v enem primeru vse neoznačene primere vzamemo za pozitivne, v drugem pa kot negativne. Nato naključno izbiramo neoznačene primere in med njimi izberemo tiste, ki jih en klasifikator označi kot pozitivne, drugi pa kot negativne.

Še ena klasična metoda je Poizvedba z odborom (angl. *Query by Committee*), ki je opisana v članku [7]. Tukaj zgradimo več klasifikatorjev za učne množice, ki jih dobimo z bagging-om nad množico neoznačenih primerov. Za označevanje izberemo primer, pri katerem je nestrinjanje med klasifikatorji največje.

Druge metode aktivnega učenja po mojem mnenju niso dovolj razširjene in njihovo uporabnost je treba še potrditi.

2.4 Rezultati in analiza testiranja metod

Metode aktivnega učenja sem implementiral s pomočjo sistema za strojno učenje Orange [3]. Testiral sem jih na večih naborih podatkov.

Prvi nabor podatkov *descriptors* je bila množica kemijskih spojin, ki so jim bili dodani deskriptorji iz programskega paketa DRAGON (opis v [8]). Vsebuje 659 primerov s po 1232 atributi. Atributi so strukturni, geometrijski, topološki deskriptorji molekul, število sprehodov in poti po vezeh in atomih molekul, sosednosti robov, 2D avtokorelacije itn. Razred predstavlja toksičnost/netoksičnost spojine.

Ostali štirje nabori podatkov so bili izbrani izmed standardnih Orangovih naborov podatkov. Prvi je bil *adult_sample*, ki vsebuje podatke iz popisa prebivalstva. Ima 977 primerov s po 14 atributi, razred je binaren in predstavlja količino letnih prihodkov osebe: $\leq$ 50000\$, $>$ 50000\$. Nabor *brown_selected* opisuje podatke o raku. Ima 187 primerov, 79 atributov, razredna spremenljivka pa ima 3 vrednosti: *Proteas*, *Resp* in *Ribo*. Tretji je zelo znan nabor podatkov *iris*, ki opisuje 3 vrste rože perunike. Vsebuje 150 primerov s 4 atributi, razred je vrsta perunike: $Iris - setosa$, $Iris - versicolor$ in $Iris - virginica$. Zadnji nabor podatkov *zoo* vsebuje podatke o različnih vrstah živali. Atributi so preproste binarne spremenljivke (npr.: lasje, peruti, jajca, mleko, hrbtenjača, ...), razred pa je ena od sedmih skupin živali, ki jim primeri pripadajo:

sesalci, ptiči, plazilci, ribe, ...

Za testiranje opisanih metod sem uporabil metodo bootstrap. Z izbiranjem s ponovnim vstavljanjem sem naredil 10 učnih množic, vsakič sem testiral na primerih, ki jih nisem izbral (*out-of-bag samples*). Rezultati na grafih prikazujejo povprečje vseh desetih iteracij učenja in testiranja.

Klasifikator, ki sem ga uporabljal za testiranje, je klasifikacijsko drevo C4.5 s privzetimi parametri v Orangu. Za boljšo primerjavo sem želel za vse nabore podatkov izbrati enak klasifikator, zanj pa sem se odločil zaradi velike hitrosti. Testiranje aktivnega učenja se je namreč izkazalo za precej zamudno in računsko zahtevno. Vendar moram poudariti, da so rezultati z drugimi klasifikatorji lahko popolnoma različni. Sam sem poizkušal še s Orangovim klasifikacijskim drevesom, naivnim Bayesovim klasifikatorjem in metodo podpornih vektorjev. Testiral sem jih na naboru podatkov *descriptors*, kjer so vsi končni klasifikatorji (naučeni na vseh primerih) dosegali precej slabši AUC od C4.5 drevesa. Tako metoda aktivnega učenja s takim klasifikatorjem tudi teoretično ne bi mogla biti boljša od tiste s C4.5 drevesom.

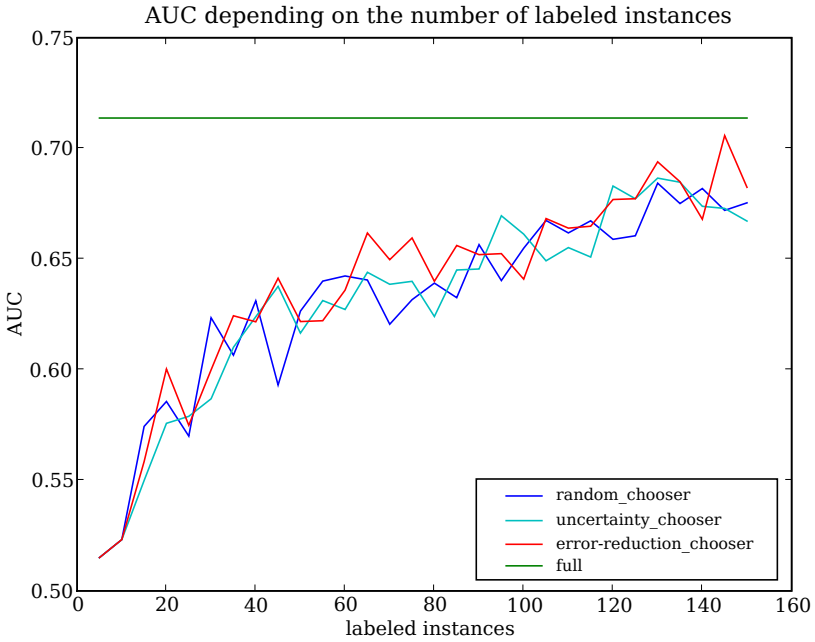
Za primerjavo sem kot metodo vključil tudi naključno izbiranje, ki na vsakem koraku aktivnega učenja nove primere za označevanje izbere naključno.

Parametri aktivnega učenja za prve štiri nabore podatkov so bili: $T = 30$ (število iteracij), $k = 5$ (število izbranih primerov na vsakem koraku) in $init\_iter = 2$ (število iteracij naključne inicializacije), za zadnji nabor podatkov *zoo* pa: $T = 30$, $k = 2$ in $init\_iter = 1$.

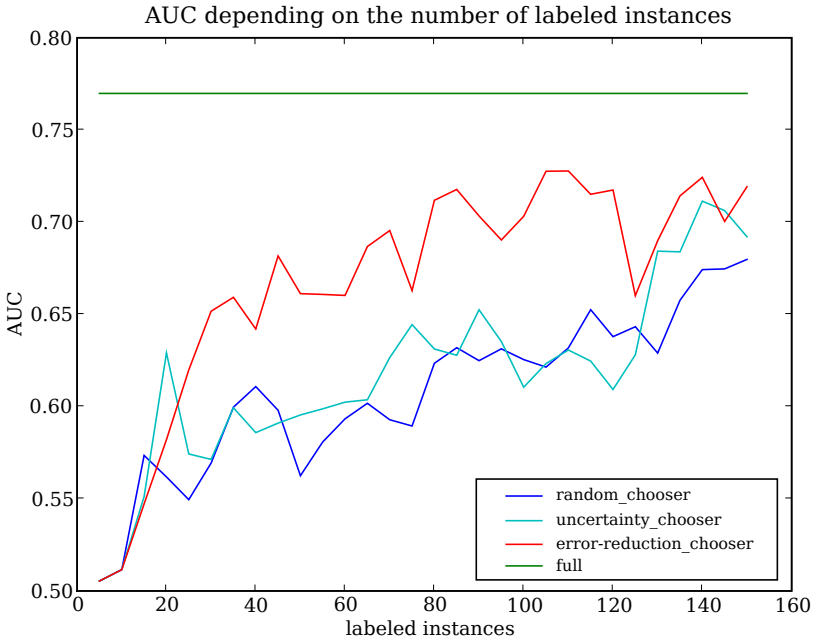
Slika 2.3(a) prikazuje rezultate na podatkovnem naboru *descriptors*. Vse tri metode se obnašajo enako dobro, nobena ni boljša od naključnega izbiranja. Tudi končni AUC za ta nabor podatkov je dokaj nizek (0.71). To kaže na to, da je sproti grajeno klasifikacijsko drevo dokaj slab napovedovalec verjetnosti razreda za neoznačene primere. Zato tudi obe nenaključni metodi nista preveč dobri.

Zanimiv rezultat sem dobil na naboru *adult_sample*, ki je viden na grafu na sliki 2.3(b). Tukaj se je prepričljivo najbolje odrezala metoda Zmanjševanje prihodnje napake, ki AUC 0.70 doseže po 80 primerih, ostali metodi pa niti po 160 primerih. V tem primeru očitno C4.5 drevo dobro napoveduje verjetnosti razreda za neoznačene primere, kar v kombinaciji z zmanjševanjem entropije razredne spremenljivke vodi k optimalnemu aktivnemu učenju z ozirom na pričakovano napako na testni množici. Samo izbiranje tistih primerov, za katere klasifikator najmanj zanesljivo napoveduje razred, očitno tukaj ni dovolj dobro in je le nekoliko boljše od naključnega izbiranja.

Na naboru podatkov *brown_selected* krivulje učenja naraščajo izjemno hitro, kot je vidno na sliki 2.4(a). Iz začetnega AUC 0.57 po manj kot 20



(a) Krivulja naraščanja AUC glede na število označenih primerov za podatke *descriptors*.



(b) Krivulja naraščanja AUC glede na število označenih primerov za podatke *adult_sample*.

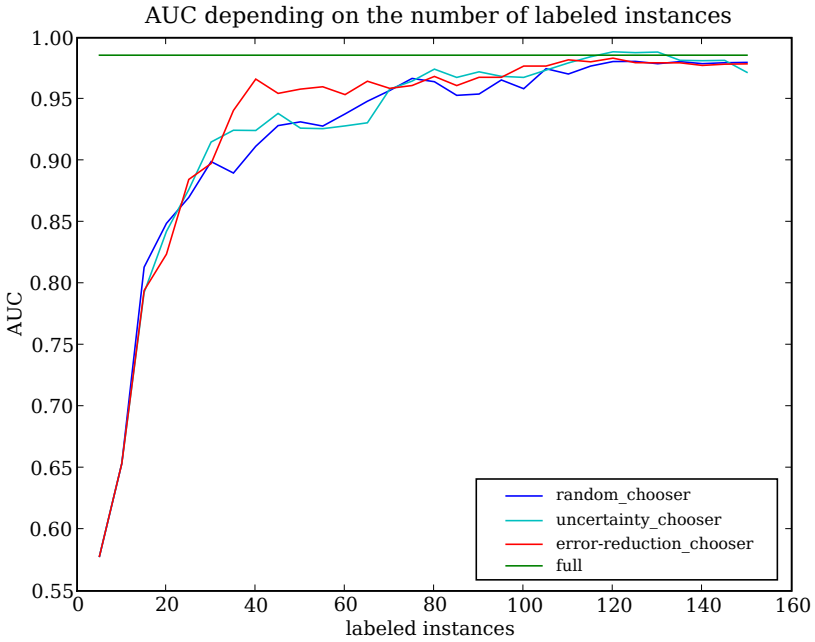
Slika 2.3: Rezultati testiranja različnih metod aktivnega učenja.

označenih primerih vse metode dosežejo AUC nad 0.80. Kasneje pa se med njimi pokažejo razlike in tudi tokrat je najboljša metoda Zmanjševanje prihodnje napake. Le-ta doseže skoraj končni AUC 0.97 po nekaj več kot 40 korakih, ostali metodi pa za tak AUC potrebujeta skoraj 80 korakov. Tukaj se je tudi pokazalo, da, ko aktivno učenje doseže AUC zelo blizu tistega končnega, krivulja začne počasi konvergirati k tej končni vrednosti.

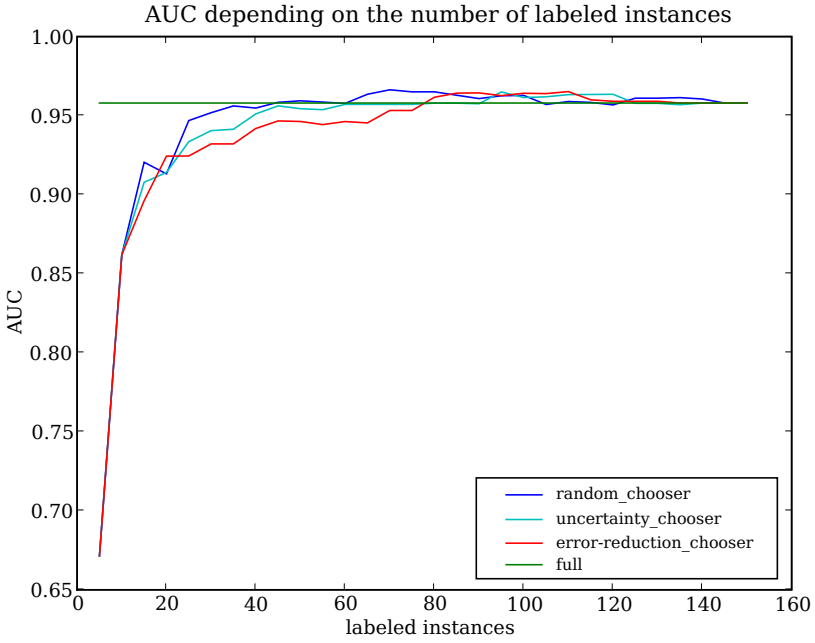
Podobno se je zgodilo tudi na naboru podatkov *iris*, kar prikazuje slika 2.4(b). Tukaj je bil začetni AUC še nekoliko višji od 0.67 in po 20 iteracijah je pri vseh metodah narasel nad 0.90. V nadaljevanju se je najbolje odrezala metoda naključnega izbiranja, a njena prednost ni bila velika. Verjetno gre pri tem za zelo specifično delovanje metod aktivnega učenja pri AUC-ju nad 0.90, ko točnosti na testni množici zaradi različnih anomalij (npr.: šuma, outlier-jev) z neko inteligentno metodo skoraj ni več mogoče izboljšati, naključno izbiranje pa ima pri tem lahko "srečno roko". Na koncu vse metode skonvergirajo h končnemu AUC-ju.

Rezultati na zadnjem naboru podatkov *zoo* so na sliki 2.4(c). Tako kot pri prejšnjih dveh naborih podatkov je tudi tukaj končni AUC skoraj 1.0. Metode iz začetnega AUC 0.5 že po nekaj več kot desetih primerih pridejo na AUC 0.9, od tam naprej pa počasi konvergirajo h končni vrednosti. Ker je imel ta nabor podatkov manj primerov, sem pri njem uporabil ustrezno nižji $k = 2$ in $init\_size = 1$ kot pri ostalih naborih. Rezultati so pokazali, da tukaj nobena metoda bistveno ne izstopa in da so vse enakovredne. Možna razlaga za to bi bila, da gre za podatke, pri katerih pametne metode aktivnega učenja svoje "pameti" ne morejo uporabiti in se zato ne odrežejo nič bolje od naključnega izbiranja.

Izvorna koda, ki sem jo uporabljal za testiranje metod aktivnega učenja, je podana v prilogi A.

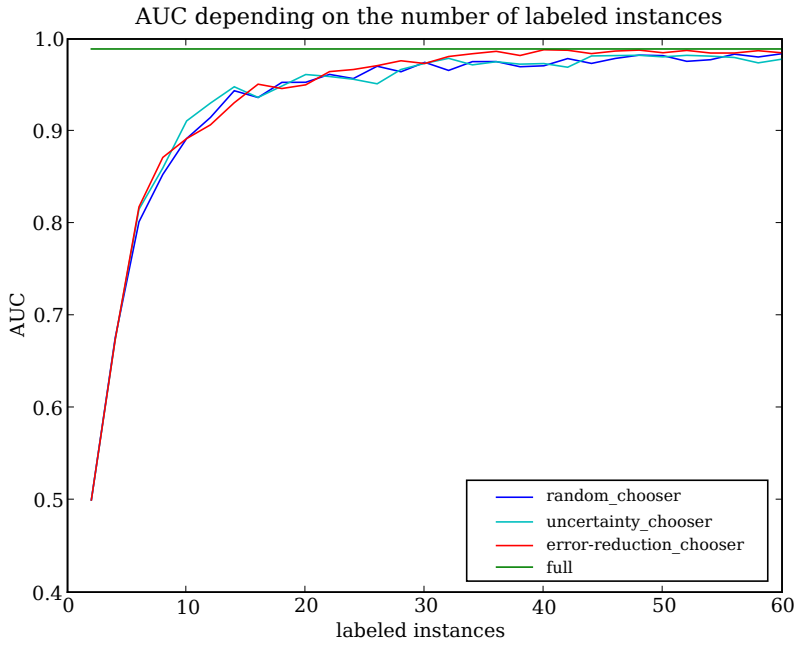


(a) Krivulja naraščanja AUC glede na število označenih primerov za podatke *brown_selected*.



(b) Krivulja naraščanja AUC glede na število označenih primerov za podatke *iris*.

Slika 2.4: Rezultati testiranja različnih metod aktivnega učenja.



(c) Krivulja naraščanja AUC glede na število označenih primerov za podatke *zoo*.

Slika 2.4 (nadalj.): Rezultati testiranja različnih metod aktivnega učenja.

Poglavje 3

Kvalitativno modeliranje v robotski domeni XPERO

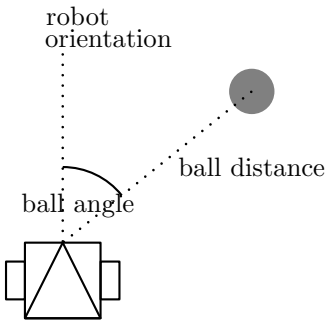
3.1 Kvalitativno modeliranje

Tradicionalno, kvantitativno modeliranje in simuliranje nam data natančne numerične rezultate. Toda običajno so te številke preveč natančne in dejansko potrebujemo samo kvalitativen odgovor. Kvalitativno modeliranje je veja umetne inteligence, ki se ukvarja s formalizacijo kvalitativnih problemov in razvojem algoritmov za njihovo reševanje.

Kvalitativno modeliranje je neke vrste abstrakcija kvantitativnih modelov. V tem procesu se del numeričnih informacij zanemari in namesto njih se uporabijo kvalitativni povzetki. Obstaja veliko načinov, kako abstrahirati podrobne numerične podatke. V našem primeru bomo namesto numeričnih vrednosti za odvod spremenljivke po času uporabili kvalitativno vrednost + za pozitivno spremembo in kvalitativno vrednost − za negativno spremembo spremenljivke.

Obstajata dve vrsti sistemov, statični in dinamični sistemi. Pri prvih se vrednosti spremeljivk s časom ne spreminjajo, pri drugih pa je potrebno upoštevati tudi njihov časovni potek. Med dinamične sisteme spada tudi naš primer robota z žogo. Za kvalitativno sklepanje v dinamičnih sistemih se lahko uporabi pristop z uporabo kvalitativnih diferencialnih enačb (*qualitative differential equations*, QDE). QDE so abstrakcija navadnih diferencialnih enačb in so osnova za algoritem QSIM, ki ga je prvič opisal Kuipers v [4] in je eden najbolj uporabljanih pristopov h kvalitativnemu sklepanju v dinamičnih sistemih.

Bolj pregledno in natačno je to področje opisal Bratko v svoji knjigi [1].



Slika 3.1: Robot in žoga. Robot opazuje dva parametra: razdaljo do žoge (*ball distance*) in kot med svojo orientacijo ter žogo (*ball angle*).

3.2 Opis robotske domene XPERO

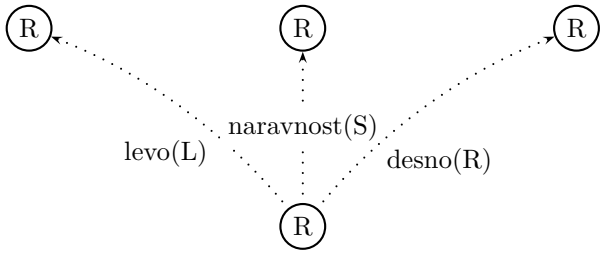
Naša robotska domena je povzeta po članku [9]. V njem je opisan primer uporabe aktivnega učenja za planiranje poskusov robota pri učenju svojega kvalitativnega modela sveta.

Robotsko domeno sestavljajo mobilni robot, žoga ter stropna kamera, ki v vsakem trenutku "vidi" celoten prostor. Robot uporablja to kamero za opazovanje svoje razdalje do žoge (*ball distance*, *bd*) ter kota med svojo orientacijo in žogo (*ball angle*, *ba*). Parametra *bd* in *ba* prikazuje tudi slika 3.1.

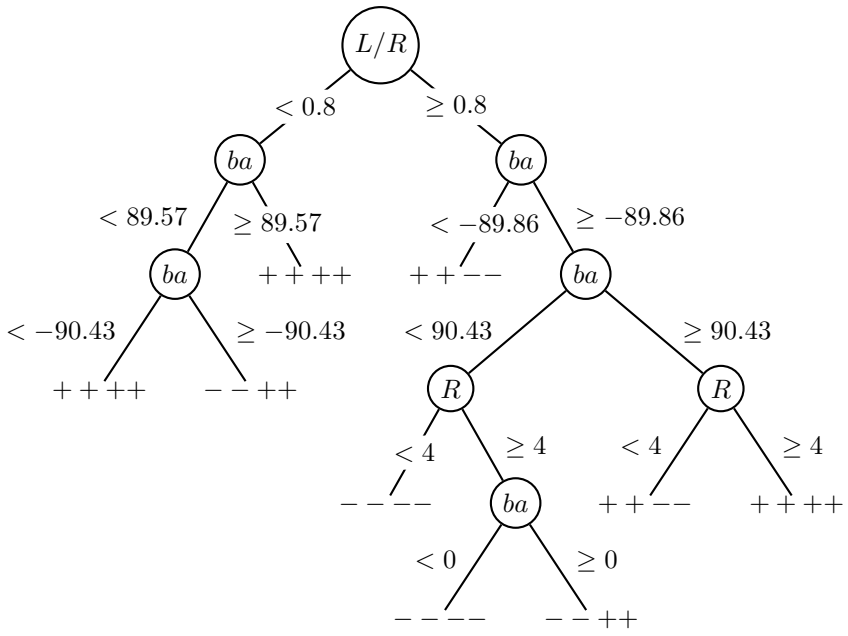
Robot se premika z nastavljanjem hitrosti levega in desnega kolesa (L in R). V našem primeru sta L in R vedno pozitivna. Robot je bil omejen tako, da je lahko izbiral le med hitrostima 3 in 5. To pomeni, da se robot lahko premakne naravnost naprej ($L = R = 5$), zavije desno ($L = 5$, $R = 3$) ali levo ($L = 3$, $R = 5$). Premikanje robota je prikazano na sliki 3.2.

Robot se ne "zaveda" nobenega koordinatnega sistema. Vse, kar pozna, so akcije, ki jih izvaja (L in R), ter opazovanja iz kamere (*bd* in *ba*).

Glavni cilj, za katerega hočemo, da ga robot doseže, je, da se nauči kvalitativnega modela, ki opisuje povezavo med robotovimi akcijami in opazovanji. Z gostim vzorčenjem celotnega prostora zgoraj omenjenih spremenljivk in učenjem kvalitativnega klasifikacijskega drevesa na teh podatkih smo dobili "idealen" model naše domene. Ta model ni bil nikjer uporabljen, služil je le našemu vpogledu v to, kar bi se moral robot sčasoma naučiti. To "idealno" kvalitativno klasifikacijsko drevo za našo domeno je prikazano na sliki 3.3. Podrobna razlaga drevesa je podana v podpoglavju 3.3.



Slika 3.2: Robotove akcije. Robot se lahko premika naravnost, levo ali desno.



Slika 3.3: T. i. "idealno" kvalitativno klasifikacijsko drevo robotske domene.

Dejanski eksperimenti z robotom in žogo so se izvajali v za ta namen pripravljenem simulatorju Simon, ki je del sistema za strojno učenje Orange [3].

3.3 Učenje kvalitativnega modela z eksperimenti

Gradnja kvalitativnega klasifikacijskega drevesa v naši domeni je drugačen način kvalitativnega modeliranja od tradicionalnega pristopa, kjer za definicijo modela poskrbi človek. V našem sistemu imamo celo robota, ki sam vpliva na časovni potek izvajanja akcij in meritev. Naša naloga je torej drugačna.

Robota želimo opremiti z raziskovalno strategijo, ki bo robotu omogočila, da se nauči modela brez zunanjega (našega) posredovanja. Nadalje želimo, da se robot nauči kvalitativnega modela, da bodo njegovi izsledki človeškemu opazovalcu lažje razumljivi. Robot je omejen na eksperimentiranje in izvajanje prej omenjenih treh akcij. Pri premikanju in snovanju novih poskusov si lahko pomaga s kvalitativnimi modeli, ki jih je predhodno sam izgradil. V našem kontekstu poskus pomeni izbiro akcije, premik robota v skladu s to akcijo ter opazovanje rezultatov akcije. Rezultat akcije je sprememba atributov *bd* in *ba*, pri čemer nas zanima samo predznak te spremembe.

Robotova motivacija za učenje je del vgrajenega algoritma učenja. Ideja za to je preprosta: ker se mora robot zanašati na svoj napovedni model, želi čim bolj optimizirati modelovo napovedno točnost. Izboljšanje modela v času zahteva zbiranje novih opazovalnih podatkov, še posebej tistih, ki so najboljši za izboljšanje napovednega modela. Če gre vse po načrtu, bo robot po določenem času zgradil model, katerega napovedi bodo vedno točne. Robot se bo torej naučil vse o svojih akcijah in njihovih vplivih na dano okolje.

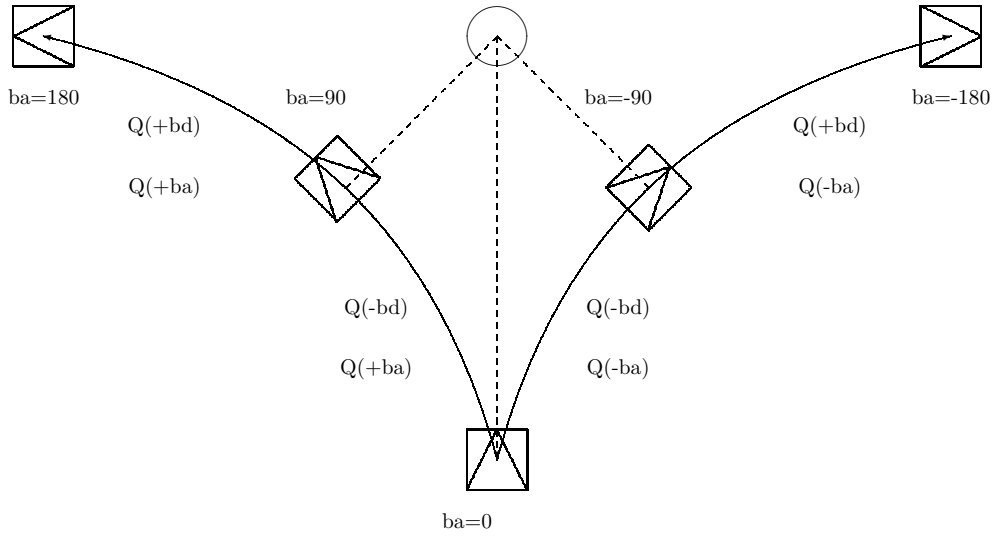
3.4 Opis kvalitativnega modela

Robot naj bi se naučil povezave med svojimi akcijami in opazovanji. V našem preprostem primeru ima robot dve spremenljivki, ki določata akcijo (L in R), ter dve spremeljivki, ki ju opazuje (*bd* in *ba*). Definirajmo relacijo Q:

$$y = Q(+x),$$

ki pomeni: parcialni odvod y po x je pozitiven.

Robot bi se torej moral naučiti, da veljajo zveze: $bd = Q(sS_L)$, $bd = Q(sS_R)$, $ba = Q(sS_L)$ in $ba = Q(sS_R)$, kjer s predstavlja predznak + ali −,

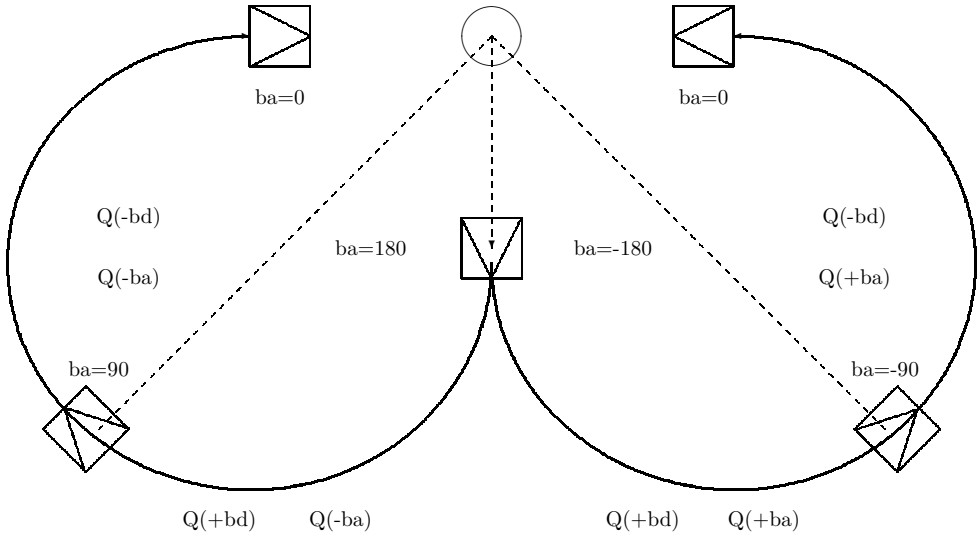


Slika 3.4: Vrednosti razreda C ($Q(+bd)Q(+ba)$, $Q(+bd)Q(-ba)$, $Q(-bd)Q(+ba)$ in $Q(-bd)Q(-ba)$), ko se robot premika v levo ali desno stran, če robot začne pri kotu med svojo orientacijo in žogo $ba = 0$.

L hitrost levega kolesa, S_L pot levega kolesa (med njima velja zveza $L = \dot{S}_L$), R hitrost desnega kolesa in S_R pot desnega kolesa (med njima velja zveza $R = \dot{S}_R$).

Za lažje branje bomo v nadaljevanju uporabili krajšo notacijo, npr. "$++--$", ki podaja samo predznake s v takem vrstnem redu, kot so podane zgornje enačbe. Oznaka "$++--$" torej pomeni: $bd = Q(+S_L)$, $bd = Q(+S_R)$, $ba = Q(-S_L)$ in $ba = Q(-S_R)$. Z besedami to pomeni: razdalja med robotom in žogo se povečuje, ko se S_L in S_R povečujeta (oz. sta odvoda $L, R > 0$), in kot med robotom in žogo se zmanjšuje, ko se S_L in S_R povečujeta. Razred C naše robotske domene bomo definirali kot urejeno četvorko predznakov, ki smo jih pravkar opisali. Različna območja vrednosti razreda C so prikazana na slikah 3.4 in 3.5.

Kvalitativni modeli, ki se jih robot nauči, so v obliki kvalitativnih klasifikacijskih dreves. K osnovnim atributom L, R, bd, ba in razredu C je s pomočjo algoritma Padé dodan na novo zgrajen atribut L/R. Atribut L/R je dobljen z uporabo verižnega pravila odvajanja, tj. deljenjem odvodov poti vsakega kolesa po času. Opisuje kvalitativno relacijo med hitrostima obeh koles robota in lahko, kot se bo kasneje izkazalo, pojasni leve in desne zavoje. Uporaba



Slika 3.5: Vrednosti razreda C ($Q(+bd)Q(+ba)$, $Q(+bd)Q(-ba)$, $Q(-bd)Q(+ba)$ in $Q(-bd)Q(-ba)$), ko se robot premika v levo ali desno stran, če robot začne pri kotu med svojo orientacijo in žogo $ba = 180$ oz. $ba = -180$.

verižnega pravila za konstrukcijo atributov je že uveljavljen postopek in ni vezan samo na našo robotsko domeno.

Poglavje 4

Aktivno učenje z robotom XPERO

4.1 Adaptacija metod aktivnega učenja za robotsko domeno

Standardno aktivno učenje temelji na predpostavki, da ima metoda aktivnega učenja na vsakem trenutku na voljo vse neoznačene primere. Druga podobna možnost je, da lahko metoda nove primere, za katere želi izvedeti razredno oznako, kar sama skonstruira. V naši robotski domeni pa so stvari popolnoma drugačne, saj lahko robot na vsakem koraku izbira le med tremi akcijami. To pomeni, da so algoritmu aktivnega učenja na voljo le trije novi neoznačeni primeri, med katerimi mora izbrati enega. Tudi razredne oznake primera algoritem ne dobi tako, da zanjo povpraša vsevednega učitelja. Metoda aktivnega učenja izbere neoznačen primer in s tem tudi naslednjo akcijo, kot jo določa izbrani primer. Po opravljeni akciji robot izmeri kvalitativno spremembo svoje razdalje do žoge ter kota med sabo in žogo. Te kvalitativne spremembe pa ravno ustrezajo razredu C prejšnjega neoznačenega primera.

Pomembna razlika med standardnim aktivnim učenjem in aktivnim učenjem v naši robotski domeni je torej časovno-prostorska omejenost izbiranja novih primerov, ki je vezana na trenutni položaj robota v prostoru. Metode aktivnega učenja, opisane v poglavju 2, te omejitve ne poznajo, zato jih je bilo potrebno za robotsko domeno ustrezno prilagoditi.

4.1.1 Izbiranje po neodločenosti

Pri tej metodi posebne adaptacije za robotsko domeno niso potrebne. Pričakujemo, da se bo metoda precej slabo odrezala in ne bo nikoli odkrila idealnega napovednega modela. Po začetnem naključnem izbiranju akcij bo robot zgradil zelo slab približek napovednega modela, s pomočjo katerega se bo naprej odločal, katero akcijo naj izbere. Ker robot na začetku niti ne pozna vseh vrednosti razreda, bo naivno mislil, da že ima dober napovedni model svoje domene. Ta algoritem tudi nima vgrajene nobene motivacije robota po učenju. Za robota bodo torej v večini primerov vse akcije enako nezanimive in se bo med njimi odločal naključno, to pa ne bo prineslo dobrega rezultata.

4.1.2 Zmanjševanje prihodnje napake

Ta metoda, kjer v resnici zmanjšujemo povprečno entropijo razredne spremenljivke neoznačenega primera, ima večji potencial. Predvsem je pomembna ugotovitev, da robot sicer res lahko izbira samo med tremi neoznačenimi primeri, ki predstavljajo možnosti za naslednjo robotovo akcijo, vendar lahko kot neoznačene primere pri ocenjevanju zmanjšanja povprečne entropije razredne spremenljivke uporablja katerikoli primer iz celotnega prostora. Dovolj je torej, da robot dobro vzorči svoj atributni prostor in s tem dobi uravnoteženo in popolno bazo primerov za celotno robotsko domeno.

Pričakujemo, da bo robot dosegel boljši rezultat kot pri zgornji metodi, vendar bo imel še vedno probleme pri izbiri ustreznih novih primerov. Robotovi premiki glede na celoten prostor so namreč zelo majhni, zato pričakujemo, da razlika med napovedjo napake za eno akcijo in napovedjo napake za drugo akcijo ne bo velika. Tako se bo ta metoda prelevila v naključno izbiranje. Vseeno ima metoda nek potencial, ki bi lahko dal boljše rezultate. Metoda namreč optimizira tisto, kar kasneje preverjamo s testno množico, tj. klasifikacijsko točnost oz. napako na gosto vzorčenem celotnem prostoru atributov.

4.1.3 Zmanjševanje prihodnje napake s povečanjem velikosti koraka

Naslednja metoda je izpeljanka osnovne metode zmanjševanja prihodnje napake. Zdi se, da so razlike med posameznimi akcijami v določenem položaju robota premajhne, da bi lahko pametno izbrali najprimernejšo. Zato dobimo intuitivno idejo, da bi lahko povečali velikost robotovega koraka. To pomeni, da robot lahko naenkrat naredi le k premikov v levo, k premikov naravnost ali

k premikov v desno.

Na vsakem koraku aktivnega učenja izberemo eno od skupin premikov levo, naravnost ali desno, ki jo označimo z $S = \{X_i | i = 1 \dots k\}$. Za vsako skupino S vsem njenim primerom X_i dodelimo isto razredno oznako Y. Za to imamo 4 različne možnosti.

Tako dobimo 4 množice $T_r = \{(X_i, Y) | i = 1 \dots k\}$, kjer je T_r r-ta označitev množice S.

Naj bosta D trenutna množica označenih primerov in $\hat{P}_D$ napoved verjetnosti razreda, ki jo da model, zgrajen na množici D. Za vsako skupino S lahko ocenimo napako kot:

$$E_{\hat{P}_{D \cup S}} = \sum_{r=1}^{4} \tilde{E}_{\hat{P}_{D \cup T_r}},$$

kjer $\tilde{E}_{\hat{P}_{D \cup T_r}}$ predstavlja povprečno entropijo napovedne verjetnosti razredne spremenljivke neoznačenih primerov:

$$\tilde{E}_{\hat{P}_{D \cup T_r}} = \frac{1}{|U|} \sum_{x \in U} H(\hat{P}_{D \cup T_r}(Y|X))).$$

Oznaka $\hat{P}_{D \cup T_r}$ predstavlja verjetnost napovedi razreda, ki jo da model, zgrajen na uniji množice D in r-te oznake množice S. Algoritem na vsakem koraku izbere tisto množico S^*, za katero velja:

$$\forall S : E_{\hat{P}_{D \cup S^*}} < E_{\hat{P}_{D \cup S}}.$$

Pričakujemo, da bo robot dosegal boljše rezultate kot pri osnovni metodi zmanjševanja napake, saj sedaj robot svoje akcije planira do neke globine naprej. Prav tako bo prisotna večja vztrajnost robota, ki je potrebna, da le-ta odkrije nove dele prostora stanj. Robot je namreč omejen s tem, da mora določeno akcijo ponoviti k-krat. Slabost tega modela je njegovo neupoštevanje verjetnosti posameznih označitev množic S, kot tudi omejitev, da morajo imeti vsi elementi iz množice S enako oznako razreda. Upamo, da bo algoritem te slabosti nadoknadil s svojo večjo hitrostjo na račun teh poenostavitev.

4.1.4 Zmanjševanje prihodnje napake z gledanjem v globino

Ta metoda je posplošitev metode Zmanjševanje prihodnje napake s povečanjem velikosti koraka. Pri slednji smo opazovali le tri skupine premikov: k korakov

v levo, naravnost ali desno. Namesto tega ta metoda uporabi splošnejši pristop. Robot lahko izbira med vsemi možnimi potmi z začetkom v njegovem trenutnem stanju in dolžino k. Takih poti je 3^k.

Vsaki izmed teh 3^k poti pripada množica primerov $S = \{X_i | i = 1 \dots k\}$. Vsaki množici S lahko priredimo 4^k različnih oznak razredov, da dobimo množico $T_r = \{(X_i, Y_i) | i = 1 \dots k\}$, kjer T_r predstavlja r-to označitev množice S.

Naj bo D trenutna množica označenih primerov in $\hat{P}_D$ napoved verjetnosti razreda, ki jo da model, zgrajen na množici D. Za vsako možno pot S lahko ocenimo napako kot:

$$E_{\hat{P}_{D \cup S}} = \sum_{r=1}^{4^k} \hat{P}_D(T_r) \tilde{E}_{\hat{P}_{D \cup T_r}},$$

kjer $\hat{P}_D(T_r)$ predstavlja verjetnost r-te označitve množice S na podlagi napovedi trenutnega modela $\hat{P}_D$ in $\tilde{E}_{\hat{P}_{D \cup T_r}}$ predstavlja povprečno entropijo razredne spremenljivke neoznačenega primera:

$$\tilde{E}_{\hat{P}_{D \cup T_r}} = \frac{1}{|U|} \sum_{x \in U} H(\hat{P}_{D \cup T_r}(Y|X))).$$

Oznaka $\hat{P}_{D \cup T_r}$ predstavlja verjetnost napovedi razreda, ki jo da model, zgrajen na uniji množice D in r-te oznake množice S. Algoritem na vsakem koraku izbere tisto pot S^*, za katero velja:

$$\forall S : E_{\hat{P}_{D \cup S^*}} < E_{\hat{P}_{D \cup S}}.$$

Ta metoda nam obeta, da bo robot z njo dosegel še boljše rezultate kot s prejšnjimi metodami. Robot sedaj izbira svoje akcije bolj preudarno in planira njihov potek. Planiranje akcij izvaja tako, da preiskuje celoten prostor možnih akcij in stanj, v katera bi z njimi prišel. Za vsako pot nato oceni pričakovano zmanjšanje entropije napovedi razreda na neoznačeni množici. Poleg tega na podlagi trenutnega modela oceni še verjetnost posamezne oznake poti in s tem uteženo gleda njeno potencialno korist. Na koncu izbere tisto pot, ki v kombinaciji verjetnosti in zmanjšanja entropije prinese največjo korist. Slabost tega pristopa je gotovo njegova eksponentna časovna zahtevnost v odvisnosti od globine preiskovanja k.

4.2 Eksperimenti

Metode aktivnega učenja v naši robotski domeni sem ovrednotil z eksperimenti. Najprej sem opravil niz eksperimentov, kjer je robot lahko poljubno izbiral

med vsemi primeri oz. med vsemi akcijami in stanji. Nato sem opravil še eksperimente, ki so v celoti odražali fizične omejitve robota pri izbiri novih primerov. Robot je lahko izbiral samo med akcijami, ki jih je lahko izvedel v svojem trenutnem položaju. Delovanje robota sem simuliral s simulatorjem Simon, ki je del sistema Orange.

4.2.1 Eksperimenti brez časovno-prostorske omejitve

Najprej sem želel preizkusiti, kako se metode aktivnega učenja obnašajo v naši robotski domeni brez dodatnih fizičnih omejitev o izbiranju primerov oz. poskusov.

V ta namen sem naredil reprezentativen vzorec prostora atributov. V kvadratnem prostoru 5000×5000 enot sem enakomerno izbral 10 x koordinat in za vsako izmed njih še 10 y koordinat. Skupaj sem dobil 100 možnih položajev robota. Potem sem za vsak tak položaj na xy mreži enakomerno izbral 12 različnih orientacij robota. Tako sem dobil 1200 možnih $(x, y, orientacija)$ stanj robota. Za vsakega izmed njih so možne 3 akcije oz. 3 novi primeri. Tako sem na koncu dobil vzorec s 3600 primeri, enakomerno razporejenimi po celotnem prostoru akcij in stanj.

S takim naborom podatkov sem simuliral robota, ki ima sposobnost teleportiranja. Robot lahko po vsaki akciji izbere poljubno novo točko na mreži, kjer se želi nahajati. Nato se teleportira na želeno mesto in tam izvede izbrano akcijo. S tem sem pretvoril robotsko domeno v običajno domeno, kjer imajo metode aktivnega učenja v vsakem trenutku dostop do vseh neoznačenih primerov. Pričakujemo, da se bodo metode aktivnega učenja na tem naboru podatkov obnašale tako, kot so se v prejšnjih primerih iz poglavja 2.

Žoga je bila fiksno postavljena na sredino kvadratnega prostora.

4.2.2 Eksperimenti v simulatorju Simon

Drugi del eksperimentov sem opravil v simulatorju Simon, kjer je robot moral upoštevati vse prostorsko-časovne omejitve glede izvajanja akcij za pridobivanje novih primerov.

Robot se je lahko premikal po kvadratnem prostoru 5000×5000 enot. Žogo sem ponovno postavil na sredino prostora. Robotu sem vsakič določil x in y koordinato naključno, razporejeno po Gaussovi distribuciji s srednjo vrednostjo (μ) 2500 in standardno deviacijo (σ) 200. Orientacija je bila prav tako izbrana naključno z Gaussovo distribucijo $\mu = 0$ in $\sigma = 180$.

Po začetnem številu naključnih akcij je sledila izbira ustreznih akcij z metodo aktivnega učenja. Pri metodah zmanjševanja prihodnje napake je imel robot na voljo tudi reprezentativen vzorec primerov iz celotnega prostora atributov, kjer so bili primeri seveda brez razrednih oznak. Lahko privzamemo, da si je robot sam sposoben zgraditi tak nabor primerov z vnašanjem ustreznih naključnih vrednosti za vrednosti atributov.

Pričakujemo, da bo robot porabil več časa in primerov, da se bo naučil tako dobrih napovednih modelov kot pri eksperimentih brez časovno-prostorskih omejitev.

4.3 Rezultati

V podpoglavjih so podani rezultati eksperimentov za robota, ki se lahko teleportira, ter za robota, ki upošteva fizične omejitve izbiranja novih poskusov. V tabeli 4.1 je podana legenda oznak, ki sem jih na grafih uporabljal za metode aktivnega učenja.

random_chooser	naključno izbiranje
uncertainty_chooser	Izbiranje po neodločenosti
error-reduction_chooser	Zmanjševanje prihodnje napake
zbm2008	raziskovalna strategija iz članka [9]
er-step2	Zmanjševanje prihodnje napake s povečanjem velikosti koraka ($k = 2$)
er-step3	Zmanjševanje prihodnje napake s povečanjem velikosti koraka ($k = 3$)
er-step5	Zmanjševanje prihodnje napake s povečanjem velikosti koraka ($k = 5$)
er-step10	Zmanjševanje prihodnje napake s povečanjem velikosti koraka ($k = 10$)
error-reduction_chooser	Zmanjševanje prihodnje napake z gledanjem v globino ($k = 2$)
full	t. i. "idealno" kvalitativno klasifikacijsko drevo

Tabela 4.1: Oznake metod aktivnega učenja, ki sem jih uporabljal na grafih.

4.3.1 Eksperimenti brez časovno-prostorske omejitve

Pri teh eksperimentih sem za testiranje metod aktivnega učenja uporabil metodo bootstrap. Z izbiranjem in ponovnim vstavljanjem sem naredil 10 učnih množic, vsakič sem testiral na primerih, ki jih nisem izbral (*out-of-bag samples*). Rezultati na grafih prikazujejo povprečje vseh desetih iteracij učenja in testiranja.

Klasifikator, ki sem ga uporabil za testiranje, je Orangovo klasifikacijsko drevo s parametri: $measure = "infoGain"$, $maxMajority = 0.95$. To drevo

sem izbral, ker ga kasneje uporabljajo tudi metode, ki sem jih testiral v simulatorju Simon. Meril sem klasifikatorjevo klasifikacijsko točnost.

Slika 4.1(a) prikazuje rezultate testiranja s parametri: $T = 60, k = 5, init\_iter = 2$, slika 4.1(b) pa prikazuje rezultate testiranja s parametri: $T = 30, k = 10, init\_iter = 2$. Parametri so podrobneje opisani v poglavju 2.

Za primerjavo je kot metoda vključeno tudi naključno izbiranje, ki na vsakem koraku aktivnega učenja nove primere za označevanje izbere naključno. Podana je tudi krivulja z oznako $full$, ki predstavlja točnost klasifikatorja, naučenega na učni množici, ki je v celoti označena. To predstavlja zgornjo mejo, ki jo lahko dosežejo metode aktivnega učenja.

4.3.2 Eksperimenti v simulatorju Simon

Pri teh eksperimentih sem poskuse pognal za 3 različna naključna semena (*random seeds*). Modele, ki jih je gradil robot, sem vsakič testiral na vzorčni množici 3600 primerov, ki je bila opisana v podpoglavju 4.2. Robot je svoj model osveževal vsakih 5 iteracij. Na začetku je 100 iteracij akcije izbiral naključno.

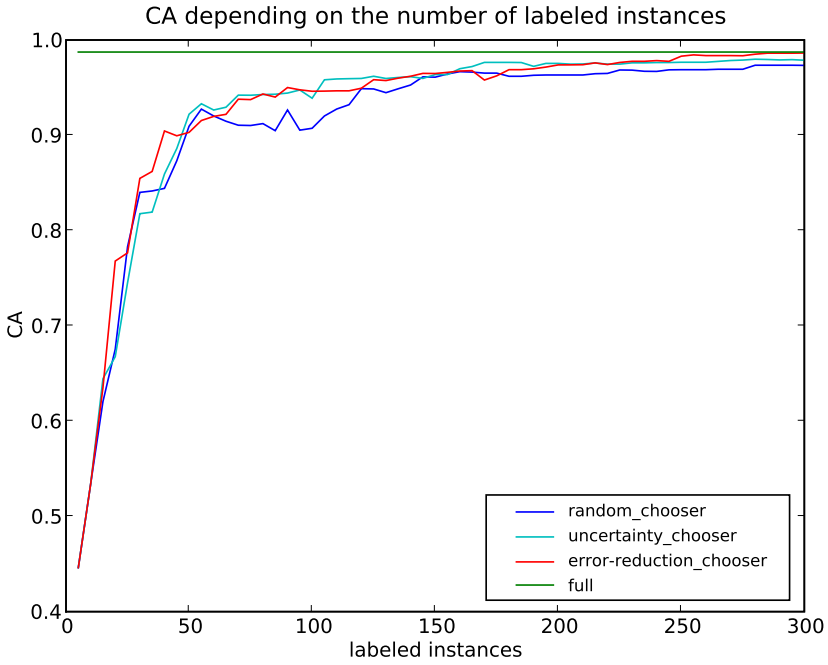
Klasifikator, ki ga je robot uporabljal za gradnjo modela, je bilo Orangovo klasifikacijsko drevo s parametri: $measure = "infoGain"$, $maxMajority = 0.95$. To drevo se je izkazalo za uspešno že v članku [9] in omogoča lažjo primerjavo metod. Poleg tega je klasifikacijsko drevo tak napovedni model, ki ljudem omogoča vpogled v znanje, ki se ga je naučil robot. To pa je pri naši robotski domeni tudi zelo pomembno.

Rezultate testiranj za 3 različne začetne položaje in orientacije robota prikazuje slika 4.2. Za primerjavo je bila dodana tudi raziskovalna metoda robota, ki je opisana v članku [9] in jo predstavlja oznaka $zbm2008$. Na grafu sta še krivulja, ki jo dobimo z naključnim izbiranjem naslednjih robotovih akcij, ter krivulja, ki predstavlja klasifikacijsko točnost "idealnega" kvalitativnega klasifikacijskega drevesa.

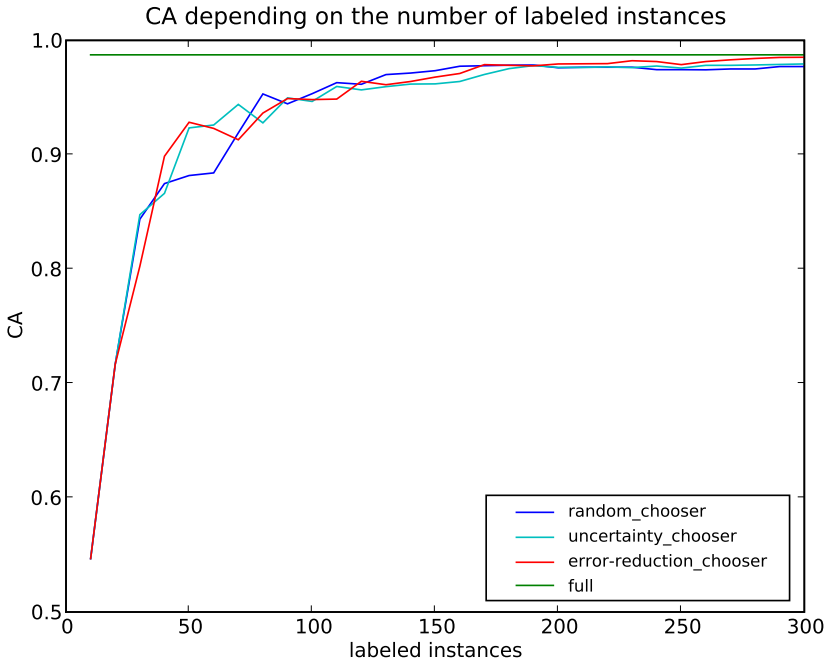
Na sliki 4.3 so podani rezultati za metodo Zmanjševanje prihodnje napake s povečanjem velikosti koraka. Testiral sem velikosti koraka $k = 2$, 3, 5 in 10. Za primerjavo sem na graf vključil še naraščanje CA običajne metode Zmanjševanje prihodnje napake. Pri vseh metodah je robot začel iz enakega položaja in z enako orientacijo.

Metodo Zmanjševanje prihodnje napake z gledanjem v globino sem testiral v primerjavi z običajno verzijo te metode. Rezultate prikazuje slika 4.4. Zaradi velike časovne zahtevnosti metode sem metodo testiral samo za globino $k = 2$.

Za lažje razumevanje različnih strategij robota so na sliki 4.5 podane tipične

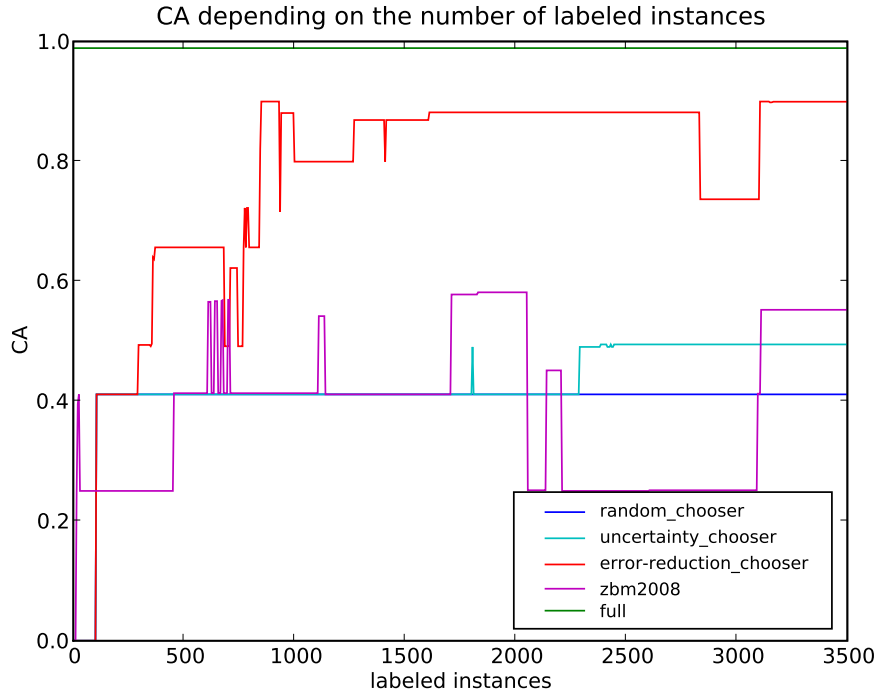


(a) Krivulja naraščanja CA glede na število označenih primerov. V vsaki od 60 iteracij metoda izbere po 5 primerov.

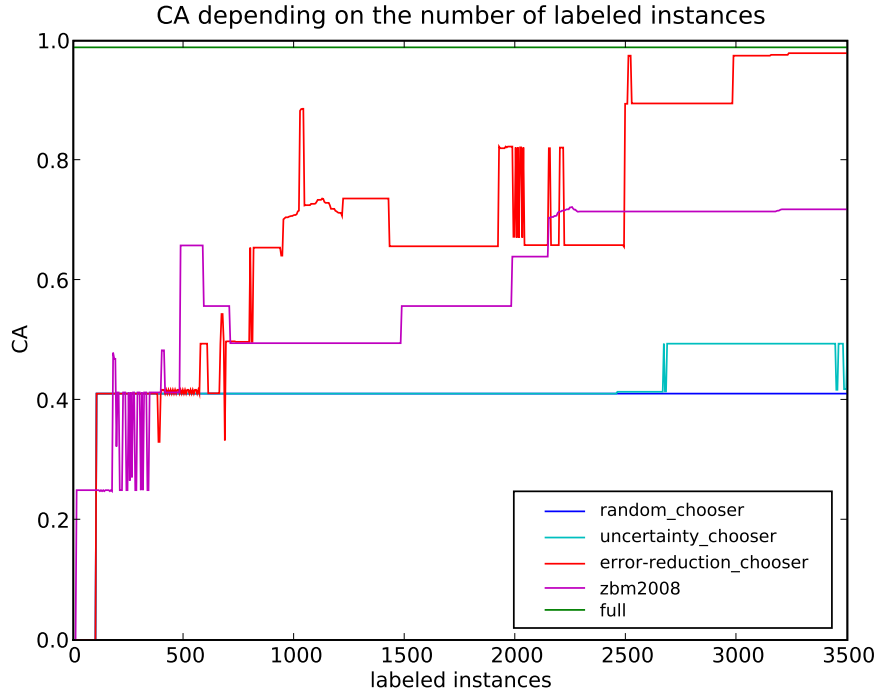


(b) Krivulja naraščanja CA glede na število označenih primerov. V vsaki od 30 iteracij metoda izbere po 10 primerov.

Slika 4.1: Rezultati testiranja različnih metod aktivnega učenja brez časovno-prostorskih omejitev.

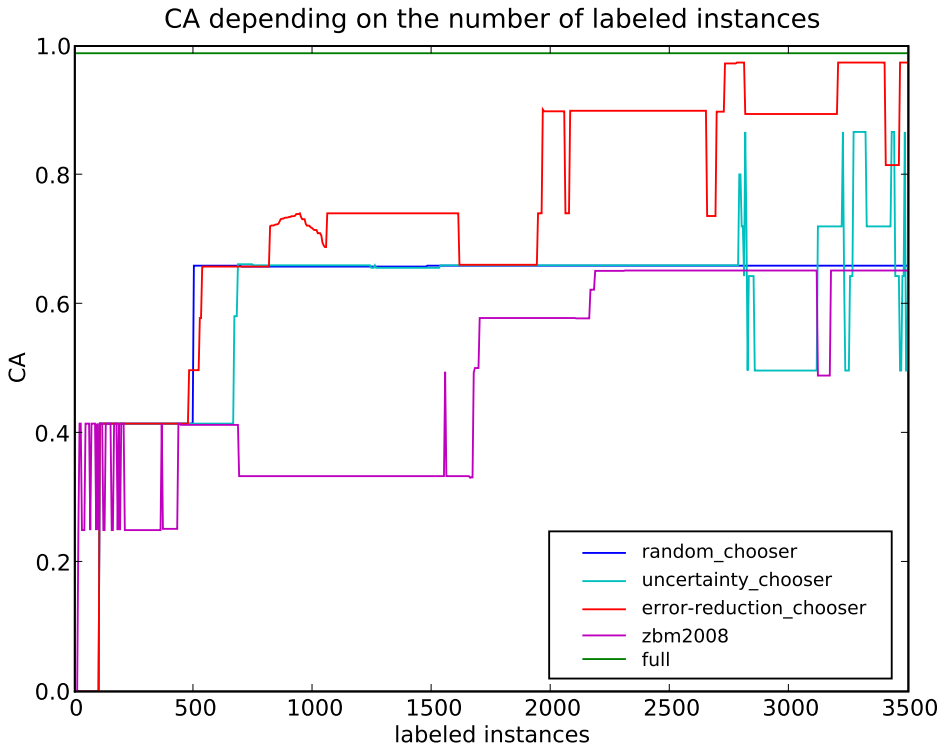


(a) Naraščanje CA v odvisnosti od števila označenih primerov. Naključno seme je 32.



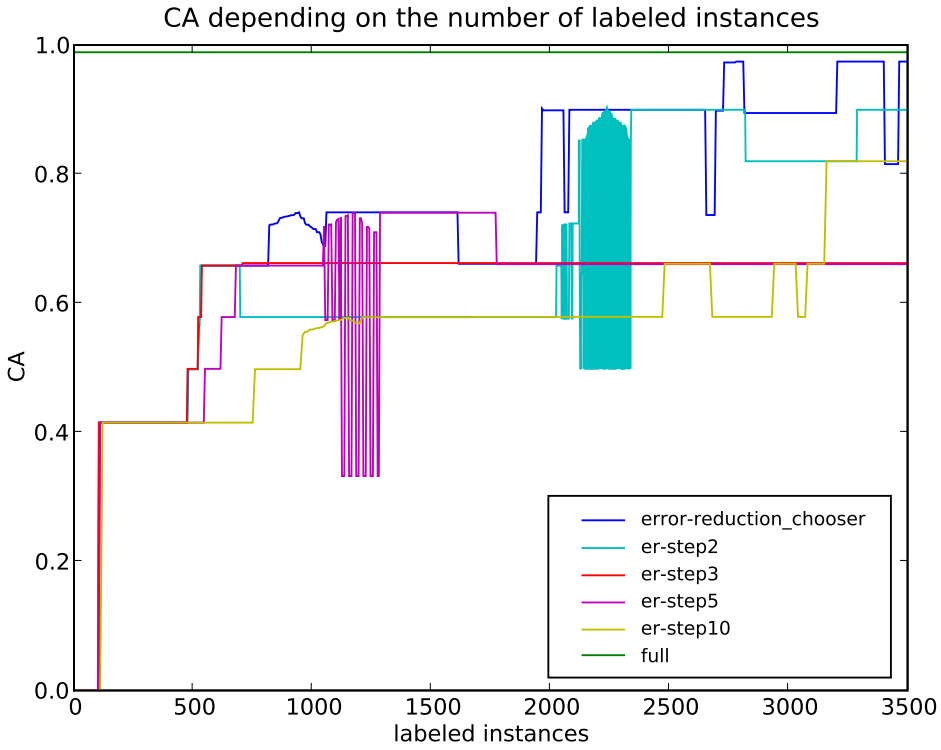
(b) Naraščanje CA v odvisnosti od števila označenih primerov. Naključno seme je 33.

Slika 4.2: Rezultati testiranja metod aktivnega učenja z upoštevanjem fizičnih omejitev. Podgrafi prikazujejo rezultate za tri različne začetne položaje robota.

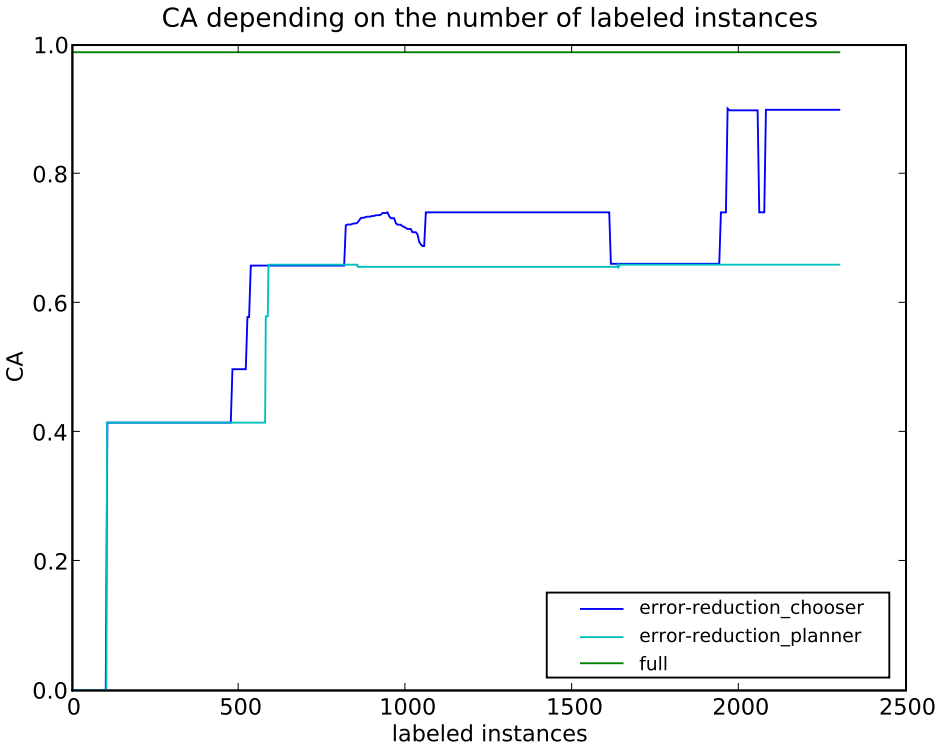


(c) Naraščanje CA v odvisnosti od števila označenih primerov. Naključno seme je 34.

Slika 4.2 (nadalj.): Rezultati testiranja metod aktivnega učenja z upoštevanjem fizičnih omejitev.Podgrafi prikazujejo rezultate za tri različne začetne položaje robota.



Slika 4.3: Naraščanje CA v odvisnosti od števila označenih primerov za metode z velikostmi koraka $k = 2$, 3, 5 in 10. Naključno seme je 34.



Slika 4.4: Naraščanje CA v odvisnosti od števila označenih primerov za metodo z gledanjem v globino za $k = 2$. Naključno seme je 34.

sledi gibanja robota za posamezno metodo.

4.4 Analiza rezultatov

V tem razdelku je podana moja interpretacija eksperimentov, ki sem jih opravil v robotski domeni.

4.4.1 Eksperimenti brez časovno-prostorske omejitve

Rezultati na sliki 4.1 kažejo, da se vse metode aktivnega učenja na robotski domeni brez časovno-prostorskih omejitev obnašajo enako kot na ostalih naborih podatkov iz poglavja 2. Če metode na vsaki iteraciji izberejo po 5 novih primerov, se za malenkost bolje od ostalih dveh metod obnese metoda Zmanjševanje prihodnje napake. Vendar so razlike med vsemi tremi metodami zanemarljive.

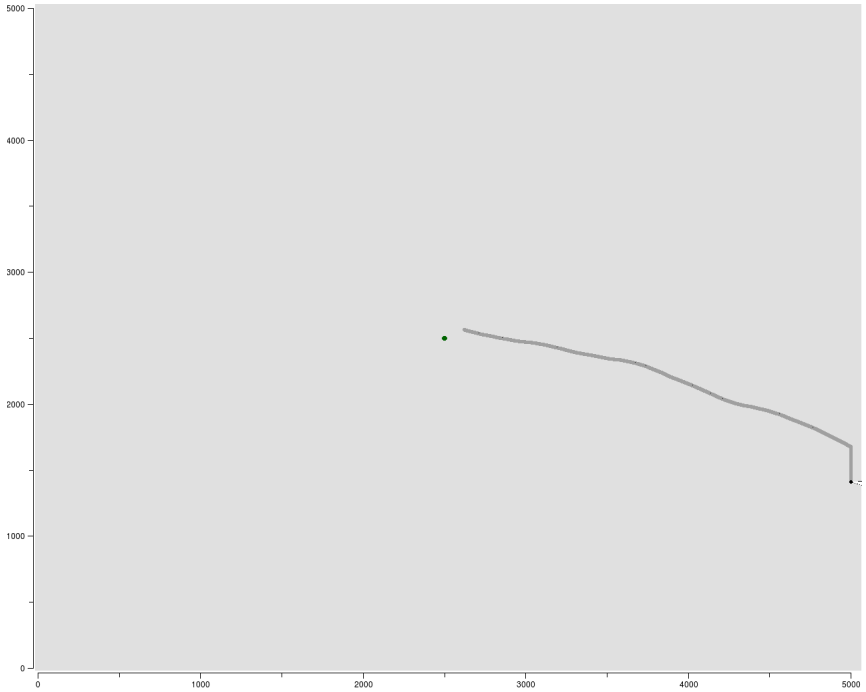
Podobno je tudi v primeru, ko metode na vsakem koraku izbirajo po 10 novih primerov. Na grafu je posebno le območje med 40 in 70 označenimi primeri, ko sta obe metodi aktivnega učenja $\approx 5\%$ boljši od naključnega izbiranja.

Pri obeh poskusih so vse metode zelo hitro dosegle zelo visok CA. Večinoma je zadostovalo že 60 primerov in metode so zgradile t. i. "idealno" kvalitativno klasifikacijsko drevo. Vse kaže na to, da je naša robotska domena sestavljena tako, da primerov ni potrebno zelo "pametno" izbirati, ampak je dovolj, da z naključnim izbiranjem dobro pokrijemo celoten prostor primerov. Ker je tudi prostor primerov enakomerno vzorčen, z njegovim pokrivanjem metode nimajo posebnih težav.

Ta rezultat predstavlja spodbudo za naprej, saj dokazuje, da lahko že z majhnim številom primerov zgradimo t. i. "idealno" klasifikacijsko drevo. Res je, da so ti primeri razporejeni po celotnem prostoru atributov in ne upoštevajo fizičnih omejitev premikanja robota, vendar pričakujemo, da nam bo uspelo tudi z robotom, ki bo upošteval vse omejitve.

4.4.2 Eksperimenti v simulatorju Simon

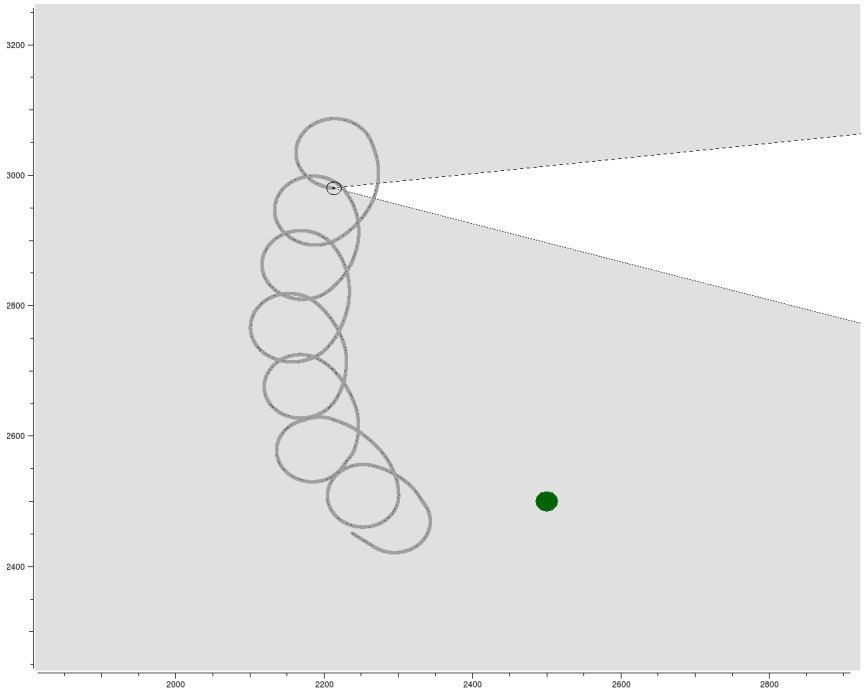
Rezultate z robotom, ki upošteva vse zakonitosti in omejitve svoje domene, prikazuje slika 4.2. Ti rezultati so veliko presenečenje. Zelo nadpovprečno se je odrezala metoda Zmanjševanje prihodnje napake, ki je z velikim naskokom premagala vse ostale metode aktivnega učenja, vključno z metodo iz članka [9]. Omeniti velja, da se rezultati metode iz članka [9] ne ujemajo s tamkajšnjimi



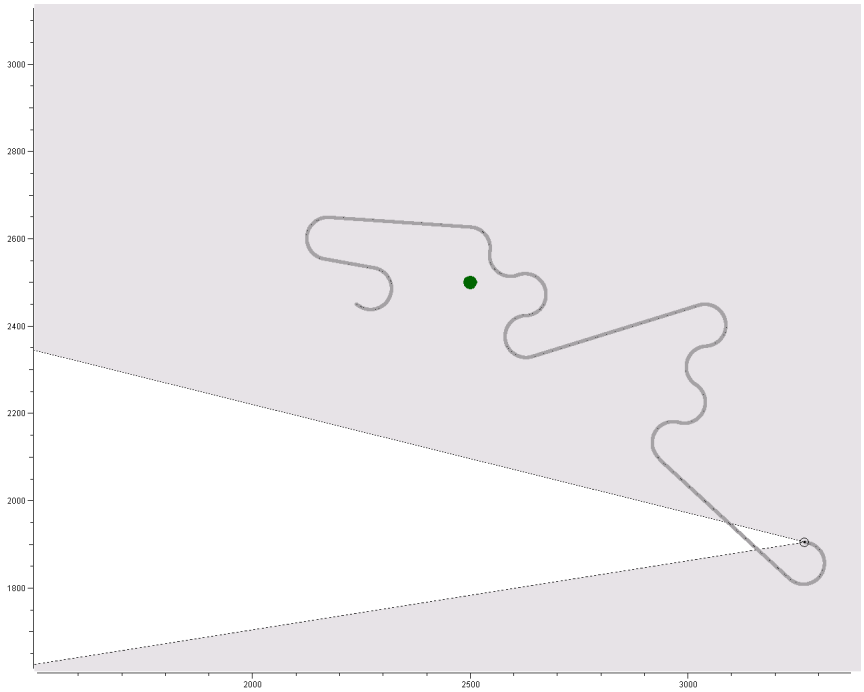
(a) Naključna metoda.

(b) Izbiranje po neodločenosti.

Slika 4.5: Tipične sledi gibanja robota za posamezno metodo.



(c) Zmanjševanje prihodnje napake.



(d) Raziskovalna strategija opisana v [9].

Slika 4.5 (nadalj.): Tipične sledi gibanja robota za posamezno metodo.

rezultati. Ne izključujem možnosti, da sem za testiranje uporabil napačno različico implementacije te metode.

Na sliki 4.2(a) metoda Zmanjševanje prihodnje napake doseže CA 90% že po manj kot 1000 iteracijah algoritma. Tako visok CA z nekaj krajšimi odstopanji ohrani vse do konca, medtem ko druge metode v tem primeru nikoli ne presežejo CA 60%.

V drugem primeru na sliki 4.2(b) je rezultat nekoliko drugačen. Pri 500. iteraciji je v vodstvo prišla metoda iz članka [9], vendar se je pri 800. iteraciji slika spet obrnila in od takrat naprej je bila najboljša metoda Zmanjševanja prihodnje napake. V tem primeru je CA nad 90% dosegla šele po 2500 iteracijah, kar pa je gotovo tudi dober rezultat. Tukaj se tudi prvič pozna razlika med metodo iz članka [9] in preostalima metodama. Naj dodamo še, da je v tem primeru metoda Zmanjševanje prihodnje napake po 3000 iteracijah zgradila "idealno" kvalitativno klasifikacijsko drevo domene.

Rezultati so v zadnjem primeru na sliki 4.2(c) podobni. Še vedno je Metoda zmanjševanje prihodnje napake daleč najboljša, zamenjani pa sta vlogi naključne in metode izbiranja po neodločenosti z metodo iz članka [9]. CA nad 90% doseže najboljša metoda že po 2000 korakih, nekoliko nižjega od 90% pa prvič doseže tudi metoda izbiranja po neodločenosti in sicer po 3300 korakih.

Pomembni so tudi rezultati na novo razvite metode Zmanjševanje prihodnje napake s povečanjem velikosti koraka, ki so prikazani na sliki 4.3. Najprej je treba nekaj reči o nenavadno pobarvanem vijoličnem in zelenem delu grafa. Do tega je prišlo, ker sta metodi s korakoma 2 in 5 zelo hitro menjavali svoja klasifikacijska drevesa. Na videz sta bili obe drevesi, slabše z atributom *bd* v vozliščih in boljše s pravim atributom *ba* v vozliščih, na množici do tedaj označenih primerov skoraj enako dobri. Robot tega, da se bo táko drevo tako slabo odrezalo na zunanjem preverjanju, ne more vnaprej predvideti. Sicer je iz grafa še razvidno, da se metode glede na število korakov k kar precej razlikujejo. Še najbližje sta si po obnašanju običajna metoda Zmanjševanje prihodnje napake in metoda z dolžino koraka 2. Ostale metode z daljšimi dolžinami koraka so se odrezale slabše, le še metoda z dožino koraka 10 je dosegla CA višji od 80%, in sicer po 3100 iteracijah. Kaže torej, da povečevanje koraka metodi Zmanjševanje prihodnje napake nič ne koristi.

Nazadnje sem testiral še metodo Zmanjševanje prihodnje napake z gledanjem v globino, kar je vidno na sliki 4.4. Metoda je gledala vse možne poti dolžine 2, vse možne oznake za vsako pot, utežene z verjetnostjo napovedi trenutnega klasifikatorja. Za primerjavo sem zraven vključil še najboljšo standardno metodo aktivnega učenja Zmanjševanje prihodnje napake. Rezultati nove metode so zelo podobni rezultatom standardne metode, a jih nikoli ne

presežejo. Tudi CA, ki ga metoda doseže po 2300 korakih, je nizek in manjši kot 70%. Pričakovali bi, da bo metoda na račun svoje kompleksnosti porabila manjše število iteracij za doseganje enako velike klasifikacijske točnosti kot standardne metode. Vendar temu ni tako. S tem je povezana tudi slabost te metode. Ima namreč eksponentno časovno zahtevnost v odvisnosti od dolžine poti k, zato njeno poganjanje že za $k = 3$ na mojem prenosnem računalniku časovno ni bilo izvedljivo.

Mogoče še bolj zanimivo je opazovanje sledi poti robota za različne algoritme aktivnega učenja na sliki 4.5. Najmanj presenetljivi sta sledi naključnega izbiranja in izbiranja po neodločenosti. V obeh primerih robot bolj ali manj sledi ravni črti, ki je odvisna le od njegove začetne orientacije v prostoru. Naključno izbiranje na dolgi rok izbere toliko premikov v levo kot premikov v desno, ki se medsebojno izničijo. Zato se robot premika naravnost do roba prostora, kjer ne more nikamor več. Pri izbiranju po neodločenosti pa se zgodi to, da robot na začetku nima dobrega napovednega modela, s katerim bi izbral najbolj negotov primer, zato je metoda prisiljena k naključnemu izbiranju med primeri. Naključno izbiranje primerov robota na dolgi rok pelje naravnost naprej in robot ne odkrije novih primerov, ki bi mu pomagali pri izboljšanju svojega napovednega modela. Ta t. i. "začarani krog" se nikoli ne konča in robot primere izbira praktično naključno.

Veliko bolj presenetljiva je slika 4.5(c), ki je sled robota z uporabo metode Zmanjševanje prihodnje napake. Robotova sled riše skoraj popolne pentljaste vzorce. Zdi se, da ima robot vgrajeno posebno komponento, ki skrbi za estetiko njegovega gibanja. Verjetno gre pri tem pojavu za to, da robot s svojo strategijo ugotovi, da se mu splačajo krožni premiki. Pri krožnem premikanju se namreč močno spreminja atribut *ba*, ki je eden glavnih atributov iskanega klasifikacijskega drevesa. To torej pomeni, da robot s spletanjem pentelj hitro izboljšuje svoj napovedni model. Tukaj velja poudariti, da je ta rezultat še toliko bolj presenetljiv, ker ta metoda ne planira svojih potez vnaprej in se vsakič sproti odloča za naslednjo akcijo.

Slika 4.5(d) dobro odraža strategijo, ki jo uporablja robot iz članka [9]. Deli slike, na katerih se robot premika naravnost, levo ali desno, predstavljajo t. i. vztrajnostni (*persistent*) del algoritma, spremembe smeri pa naključni del algoritma. Iz slike je razvidno, da robot skuša dobro pokriti prostor atributov ter da je vanj vgrajena motivacija po učenju oz. eksperimentiranju.

Poglavje 5

Sklepne ugotovitve

Pri pregledu pomembnejših pristopov k aktivnemu učenju se je pokazalo, da s temi metodami na nekaterih naborih podatkov hitreje dosežemo boljši AUC, kot če bi primere izbirali naključno. Vendar se je hkrati pokazalo, da v veliko primerih tega ne moremo doseči.

Po transformaciji naše robotske domene v nabor primerov, ustreznih za standardne metode aktivnega učenja, se je pokazalo, da bi s pametnim izbiranjem primerov lahko hitro dosegli visok CA. To je bila iztočnica za adaptacijo metod aktivnega učenja za delo v časovno-prostorsko odvisni robotski domeni.

Izkazalo se je, da daje standardna metoda aktivnega učenja Zmanjševanje prihodnje napake odlične rezultate v naši robotski domeni. To je zelo presenetljivo, saj je metodi v vsaki iteraciji na voljo le omejeno število neoznačenih primerov, ki predstavljajo možne akcije robota v njegovem trenutnem stanju. Da ne gre za naključje, potrjujejo tudi slike sledi premikanja robota, kjer robot iz različnih začetnih pozicij vedno ustvari nekakšen pentljasti vzorec.

Druga ugotovitev je povezana z metodami, ki pri izbiri poskusov uporabljajo planiranje. Obe predlagani metodi, Zmanjševanje prihodnje napake s povečanjem velikosti koraka in Zmanjševanje prihodnje napake z gledanjem v globino, sta sicer dokaj dobri, a v primerjavi z osnovno metodo njuni rezultati zbledijo. Izkaže se, da povečevanje vztrajnosti s podaljševanjem koraka v izbrani smeri metodi ne koristi. Prav tako preiskovanje prostora možnih premikov in oznak razreda ne obrodi sadov. Poleg tega je ta metoda še tako počasna, da je za praktično premikanje robota neuporabna.

V prihodnje bi lahko naredili še bolj sistematično povezavo področja aktivnega strojnega učenja s področjem planiranja eksperimentov. Lahko bi se zgledovali po raziskovalnih strategijah, ki jih uporabljajo živa bitja v naravi in jih poizkušali eksperimentalno in teoretično ovrednotiti. Zanimivo bi bilo tudi,

če bi namesto simulatorja uporabili pravega robota v kompleksnejši domeni.

Dodatek A

Implementacija splošnih metod aktivnega učenja

```
1  import random, time, math
2  import orange, orngTest, orngStat, orngEnsemble
3  import pylab
4  from matplotlib.font_manager import FontProperties
5  random.seed(23)
6
7  def add(list1, list2):
8      """Returns a component-wise sum of lists list1 and list2."""
9      return map(lambda x, y: x + y, list1, list2)
10
11 def bootstrap_indices(n):
12     """Makes indices for bootstrap sample from bag of cardinality
           n. Returns
13     bootstrap sample indices and out-of-bag indices.
14     """
15     indices = []
16     out_of_bag = n*[1] #na zacetku so vsi primeri out-of-bag
17     for i in range(n):
18         r = random.randint(0, n-1)
19         indices.append(r)
20         out_of_bag[r] = 0
21     return indices, [i for i in range(n) if out_of_bag[i] == 1]
22
23 def make_bootstrap_sample(data, bootstrap_sample, out_of_bag):
24     """Constructs orange ExampleTable train with indices from
           bootstrap_sample
25     and orange ExampleTable test with indices from out_of_bag.
26     """
27     train = orange.ExampleTable(data.domain)
```

```
28      train.extend([data[i] for i in bootstrap_sample])
29      test = orange.ExampleTable(data.domain)
30      test.extend([data[i] for i in out_of_bag])
31      return train, test
32
33  def experiment(data_file, attr_id, learner=orange.TreeLearner, \
34                 disc_intervals=0, iter=10):
35      """Performs an active learning simulation on data set given in
            data_file.
36          It uses bootstrap validation method.
37      - iter: number of bootstrapped data sets
38      - attr_id: id of an attribute that uniquely identifies an
            instance in data
39      - learner: learner to use
40      - disc_intervals: number of intervals for equal-frequency
            discretization
41          = 0, no discretization
42          = 1, # intervals is equal to the square root of the number
                of instances
43          (so called proportional k-interval discretization)
44          = x > 1, # intervals is x
45      """
46      t = time.clock()
47      print "Reading data... %fs" % (time.clock()-t)
48      data = orange.ExampleTable(data_file)
49      if attr_id == None:
50          attr_id = "id"
51          meta_id = orange.IntVariable(attr_id)
52          id = orange.newmetaid()
53          data.domain.addmeta(id, meta_id)
54          for i, ex in enumerate(data):
55              ex[meta_id] = i
56      oracle = Oracle(data, attr_id)
57      if (disc_intervals > 0):
58          print "Discretizing data... %fs" % (time.clock()-t)
59          #WARNING! V resnici podatkov ne bi smeli diskretizirati
                pred BS preverjanjem
60          #To naredimo zaradi hitrejsega racunanja.
61          if (disc_intervals == 1):
62              disc_intervals = int(len(data)**(0.5))
63          data = orange.Preprocessor_discretize(data, \
64                      method=orange.EquiNDiscretization(
                            numberOfIntervals=disc_intervals))
65      T = 30
66      k = 5
67      init_iter = 2
68      CAs_random = [0]*T
```

```
69      AUCs_random = [0]*T
70      CAs_uncertainty = [0]*T
71      AUCs_uncertainty = [0]*T
72      CAs_errreduct = [0]*T
73      AUCs_errreduct = [0]*T
74      CA_full = 0
75      AUC_full = 0
76      for i in range(iter):
77          print "Iteration %i:" % i
78          print "Constructing bootstrap sample... %fs" % (time.clock
                ()-t)
79          bootstrap_sample, out_of_bag = bootstrap_indices(len(data)
                )
80          train, test = make_bootstrap_sample(data, bootstrap_sample
                , out_of_bag)
81          # learn and test the learner on full data set (for
                reference)
82          res = orngTest.learnAndTestOnTestData([learner], train,
                test)
83          CA_full += orngStat.CA(res)[0]
84          AUC_full += orngStat.AUC(res)[0]
85          print "full learner, CA: %s, AUC: %s" % (CA_full, AUC_full
                )
86          # delete class labels for train data set
87          delete_class_label(train)
88          # random active learning
89          CAs_i, AUCs_i = active_learning(T, train, init_iter,
                learner, random_chooser, k, oracle, test)
90          CAs_random = add(CAs_random, CAs_i)
91          AUCs_random = add(AUCs_random, AUCs_i)
92          # uncertainty active learning
93          CAs_i, AUCs_i = active_learning(T, train, init_iter,
                learner, uncertainty_chooser, k, oracle, test)
94          CAs_uncertainty = add(CAs_uncertainty, CAs_i)
95          AUCs_uncertainty = add(AUCs_uncertainty, AUCs_i)
96          # error-reduction active learning
97          CAs_i, AUCs_i = active_learning(T, train, init_iter,
                learner, error_reduction_chooser, k, oracle, test)
98          CAs_errreduct = add(CAs_errreduct, CAs_i)
99          AUCs_errreduct = add(AUCs_errreduct, AUCs_i)
100     # average the results over all iterations
101     CAs_random = map(lambda x: 1.*x/iter, CAs_random)
102     AUCs_random = map(lambda x: 1.*x/iter, AUCs_random)
103     CAs_uncertainty = map(lambda x: 1.*x/iter, CAs_uncertainty)
104     AUCs_uncertainty = map(lambda x: 1.*x/iter, AUCs_uncertainty)
105     CAs_errreduct = map(lambda x: 1.*x/iter, CAs_errreduct)
106     AUCs_errreduct = map(lambda x: 1.*x/iter, AUCs_errreduct)
```

```
107     CA_full = 1.*CA_full/iter
108     AUC_full = 1.*AUC_full/iter
109     # plot the results
110     pylab.figure(1)
111     pylab.plot(range(k, (T+1)*k, k), AUCs_random, 'b')
112     pylab.plot(range(k, (T+1)*k, k), AUCs_uncertainty, 'c')
113     pylab.plot(range(k, (T+1)*k, k), AUCs_errreduct, 'r')
114     pylab.plot(range(k, (T+1)*k, k), [AUC_full]*T, 'g')
115     pylab.legend(('random_chooser', 'uncertainty_chooser', 'error-
            reduction_chooser', 'full'), 'lower right', \
116                  prop = FontProperties(size='small'))
117     pylab.xlabel('labeled instances')
118     pylab.ylabel('AUC')
119     pylab.title('AUC depending on the number of labeled instances'
            )
120     pylab.savefig('../data/AUCs.pdf')
121     pylab.figure(2)
122     pylab.plot(range(k, (T+1)*k, k), CAs_random, 'b')
123     pylab.plot(range(k, (T+1)*k, k), CAs_uncertainty, 'c')
124     pylab.plot(range(k, (T+1)*k, k), CAs_errreduct, 'r')
125     pylab.plot(range(k, (T+1)*k, k), [CA_full]*T, 'g')
126     pylab.legend(('random_chooser', 'uncertainty_chooser', 'error-
            reduction_chooser', 'full'), 'lower right', \
127                  prop = FontProperties(size='small'))
128     pylab.xlabel('labeled instances')
129     pylab.ylabel('CA')
130     pylab.title('CA depending on the number of labeled instances')
131     pylab.savefig('../data/CAs.pdf')
132
133 def delete_class_label(data):
134     """Changes class labels for all instances in data to '?'."""
135     for ex in data:
136         ex.setclass('?')
137
138 class Oracle:
139     """An oracle for class prediction of instances in data.
            Attr_id is the name
140     of the meta attribute that uniquely defines an instance in
            data.
141     """
142     def __init__(self, data, attr_id):
143         self.attr_id = attr_id
144         self.dict = {}
145         for example in data:
146             self.dict[example.getmeta(self.attr_id)] = example.
                    getclass()
147
```

```
148     def __call__(self, example):
149         print example
150         return self.dict[example.getmeta(self.attr_id)]
151 
152 def active_learning(T, U, init_iter, learner, chooser, k, oracle,
        test):
153     """Performs standard active learning algorithm. Returns the
            trained model.
154     - T: total number of feedback iterations
155     - U: pool of unlabeled instances
156     - init_size: number of initial random feedback iterations
157     - learner: learning algorithm used for learning
158     - chooser: active learning algorithm, which will select k
            unlabeled
159         instances for labeling in the next iteration of the
                algorithm
160     - k: number of unlabeled instances chooser chooses in each
            iteration
161     - oracle: function that assigns correct class to the instance
            it is given
162     - test: set of test instances
163     """
164     t = 0 # iteration number
165     U_t = U # unlabeled instances at t-th iteration
166     L_t = orange.ExampleTable(U.domain) # labeled instances at t-
            th iteration
167     M_t = None # model at t-th iteration
168     CAs = [0]*T
169     AUCs = [0]*T
170 
171     # initialization
172     while t < init_iter:
173         # randomly choose new instances
174         (S_t, U_t) = random_chooser(U_t, k)
175         # label new instances
176         for ex in S_t:
177             ex[U.domain.classVar.name] = oracle(ex)
178             L_t.append(ex)
179         # build classification model
180         M_t = learner(L_t)
181         # test classification model
182         res = orngTest.testOnData([M_t], test)
183         CAs[t] = orngStat.CA(res)[0]
184         AUCs[t] = orngStat.AUC(res)[0]
185         print "iter: %i, CA: %s, AUC: %s" % (t, CAs[t], AUCs[t])
186         # next iteration
187         t += 1
```

```
188     # active learning
189     while t < T:
190         # choose the most appropriate new instances
191         (S_t, U_t) = chooser(U_t, k, M_t, L_t, learner)
192         # label new instances
193         for ex in S_t:
194             ex[U.domain.classVar.name] = oracle(ex)
195             L_t.append(ex)
196         # build classification model
197         M_t = learner(L_t)
198         # test classification model
199         res = orngTest.testOnData([M_t], test)
200         CAs[t] = orngStat.CA(res)[0]
201         AUCs[t] = orngStat.AUC(res)[0]
202         print "iter: %i, CA: %s, AUC: %s" % (t, CAs[t], AUCs[t])
203         # next iteration
204         t += 1
205     return CAs, AUCs
206
207 def random_chooser(U, k, M=None, L=None, learner=None):
208     """Chooses k instances from unlabeled instance set U randomly
            (model M,
209     labeled set L and learner are passed as an argument for
            compatibility with
210     other chooser functions, but they are never used).
211     Returns the list of selected instances and reduced set of
            unlabeled
212     instances.
213     """
214     indices2 = orange.MakeRandomIndices2(p0=min(len(U), k))
215     ind = indices2(U)
216     selected = U.select(ind, 0)
217     unlabeled = U.select(ind, 1)
218     return selected, unlabeled
219
220 def uncertainty_chooser(U, k, M, L=None, learner=None):
221     """Chooses k instances from unlabeled instance set U, for
            which the model M
222     is most uncertain about (labeled set L and learner are passed
            as an argument
223     for compatibility with other chooser functions, but they are
            never used).
224     Returns the list of selected instances and reduced set of
            unlabeled
225     instances.
226     """
```

```
227     # store sequential number and uncertainty for each tested
            example
228     results = [(i, min(M(ex, orange.GetProbabilities))) for i, ex
            in enumerate(U)]
229     # sort tested examples according to higher uncertainty
230     results.sort(cmp = lambda x, y: cmp(x[1], y[1]), reverse =
            True)
231     # select min(k, len(U)) most uncertain examples
232     selectedIndices = [r[0] for i, r in enumerate(results) if i <
            k]
233     # build indices for selection from unlabeled set U
234     ind = [1 if i in selectedIndices else 0 for i in range(len(U))
            ]
235     # make two new data sets
236     selected = U.select(ind, 1)
237     unlabeled = U.select(ind, 0)
238     return selected, unlabeled
239
240 def log2(x):
241     """Returns log2(x) if x > 0 and 0 otherwise. """
242     return math.log(x, 2) if x > 0 else 0
243
244 def H(ps):
245     """Calculates the entropy of probability vector ps (vector ps
            is normalized
246     so that sum(ps) == 1).
247     """
248     s = sum(ps)
249     ps = [1.*p/s for p in ps]
250     return -sum([p * log2(p) for p in ps])
251
252 def expected_error_log_loss(U, M):
253     """Calculates the expected error of all possible labelings for
             each instance
254     in unlabeled set U using model M. Error is measured using
            entropy H.
255     """
256     error = 0
257     for ex in U:
258         error += H(M(ex, orange.GetProbabilities))
259     return error
260
261 def insert_sorted(elem, list, cmp):
262     """Inserts elem in sorted list using cmp for comparing two
            elements. The
263     largest element gets deleted and the new list is returned.
264     """
```

```
265     j = 0
266     while (j < len(list) and cmp(elem, list[j]) == 1):
267         j += 1
268     temp = elem
269     while (j < len(list)):
270         list[j], temp = temp, list[j]
271         j += 1
272     return list
273
274 def error_reduction_chooser(U, k, M, L, learner):
275     """Chooses an instance from unlabeled instance set U that
            maximizes the
276     reduction in the total predicted label entropy (entropy of
            class variable on
277     all unlabeled instances in U).
278     Returns the list of selected instances and reduced set of
            unlabeled
279     instances.
280     """
281     pool_size = 50
282     # table of tuples (error, example, example_index)
283     min_error = []
284     P_indices2 = orange.MakeRandomIndices2(p0=min(len(U),
            pool_size))
285     P_ind = P_indices2(U)
286     P = U.select(P_ind, 0)
287     out_of_P = U.select(P_ind, 1)
288     for i, ex in enumerate(P):
289         error_i = 0
290         for classValue in P.domain.classVar.values:
291             ex.setclass(classValue)
292             # add ex to labeled set L
293             L.append(ex)
294             M_i = learner(L)
295             error_i += expected_error_log_loss(U, M_i)
296             # remove ex from labeled set L
297             L.__delitem__(-1)
298         if (i < k):
299             min_error.append((error_i, ex, i))
300             if (i == k-1):
301                 min_error.sort(cmp = lambda x, y: cmp(x[0], y[0]))
302         elif (error_i < min_error[-1][0]):
303             insert_sorted((error_i, ex, i), min_error, \
304                           cmp = lambda x, y: cmp(x[0], y[0]))
305     for error, ex, i in min_error:
306         print "error: %f, index: %i, example: %s" % (error, i, ex)
307     # select min(k, len(U)) most uncertain examples
```

```
308     selectedIndices = [x[2] for x in min_error]
309     # build indices for selection from unlabeled set U
310     ind = [1 if i in selectedIndices else 0 for i in range(len(P))
            ]
311     # make two new data sets
312     selected = P.select(ind, 1)
313     unlabeled = P.select(ind, 0)
314     # add all unlabeled instances that were not in the pool P
315     unlabeled.extend(out_of_P)
316     return selected, unlabeled
317
318 #lnr = orange.SVMLearner()
319 #lnr.probability = True
320 #lnr = orange.BayesLearner()
321 #lnr = orngEnsemble.RandomForestLearner(trees=50)
322 lnr = orange.C45Learner()
323 experiment("../data/descriptors.tab", "No.", lnr, disc_intervals
        =0, iter=10);
```

Slike

Literatura

[1] I. Bratko: *Prolog Programming for Artificial Intelligence*, pogl. 20, str. 520–554. Addison Wesley Publishing Company, 2001.

[2] D. Cohn, L. Atlas in R. Ladner: *Improving generalization with active learning.* Machine Learning, 15(2):201–221, 1994.

[3] J. Demsar, B. Zupan, G. Leban in T. Curk: *Orange: From Experimental Machine Learning to Interactive Data Mining.* Lecture Notes in Computer Science, str. 537–539, 2004.

[4] B. Kuipers: *Qualitative Simulation.* Artificial Intelligence, 29(3):289–338, 1986.

[5] D. Lewis in W. Gale: *A sequential algorithm for training text classifiers.* Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval, str. 3–12, 1994.

[6] N. Roy in A. McCallum: *Toward optimal active learning through sampling estimation of error reduction.* Proc. 18th International Conf. on Machine Learning, str. 441–448, 2001.

[7] H. Seung, M. Opper in H. Sompolinsky: *Query by committee.* Proceedings of the fifth annual workshop on Computational learning theory, str. 287–294, 1992.

[8] R. Todeschini in V. Consonni: *DRAGON software for the Calculation of Molecular Descriptors.* Web version, 3, 2003.

[9] J. Zabkar, I. Bratko in A. Mohan: *Learning Qualitative Models by an Autonomous Robot.* Proceedings of the 22th International Workshop on Qualitative Reasoning, 2008.

Izjava

Izjavljam, da sem diplomsko nalogo izdelal samostojno pod vodstvom mentorja akad. prof. dr. Ivana Bratka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Ljubljana, 8. 9. 2008 Tadej Janež