# Introduction to Data Mining



## 4. domača naloga: logistična regresija

**1. (30 točk)** Implementirajte logistično regresijo. Uporabite priloženo ogrodje ter ga dopolnite z:

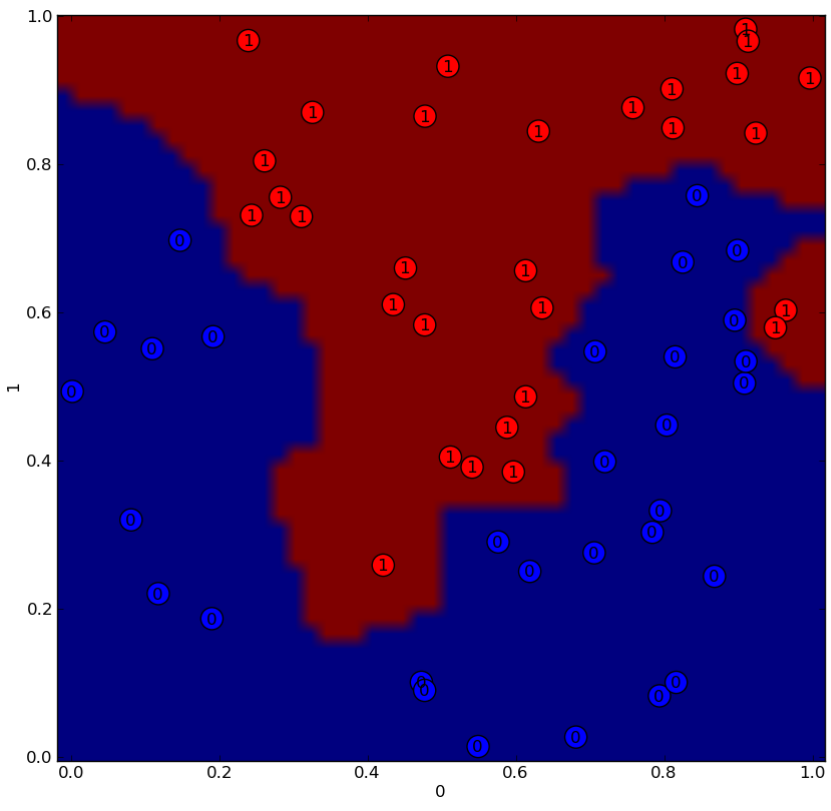
- izračunom verjetnosti za posamezni primer, *h*,
- cenilno funkcijo, *cost*, in
- gradientom cenilne funkcije, *grad*.

Opazili boste, da sta gradnja napovednega modela in napovedovanje razreda posameznim primerom ločeni: razred *LogRegLearner* iz učnih podatkov zgradi napovedni model tipa *LogRegClassifier,* ki lahko nato za poljuben vektor značilk napove verjetnosti obeh razredov. Ogrodje rešuje optimizacijski problem s funkcijo fmin_l_bfgs_b (preprost primer uporabe), zato morate le implementirati *cost* in *grad*.

Preverjanje:

- Ker vemo, da lahko cost in grad množimo s poljubno konstanto, prilagodite vašo rešitev testom. Najverjetneje bo treba deliti s številom primerov. Datoteko s testi shranite v direktorij z vašo rešitvijo (vašo rešitev poimenujte solution.py).
- Če gradnjo modela logistične regresije brez regularizacije poženete na celotnih podatkih reg.data, vam mora zgrajen model vse primere uvrstiti prav tako, kot je zapisano v reg.data.

**2. (20 točk)** Uporabite kodo za izris napovedi. Na celotnem podatkovnem naboru reg.data (brez regularizacije) dobite spodnji izris (če ga ne, popravite logistično regresijo), kjer točke označujejo primere posameznih razredov, barva ozadja pa verjetnost napovedi v tistem delu prostora. Na abscisi je vrednost prve značilke (indeks 0), na ordinati pa druge (indeks 1).




Raziščite vpliv regularizacije na napovedi, tako da spreminjate vrednosti parametra `lambda`. Izrišite tri zanimive stopnje regularizacije. Katera je najboljša in zakaj?

**3. (25 točk)** Implementirajte k-kratno prečno preverjanje kot funkcijo `test_cv(learner, X, y, k)`, ki vrne napovedi za vse primere (napovedi so verjetnosti za oba razreda) v enakem zaporedju kot so v X, le da nikoli ne uporablja istih primerov za napovedi in učenje. Razvijte še mero napovedne točnosti kot funkcijo `CA(real, predictions)`. Funkciji naj se uporabljata takole:

```
learner = LogRegLearner(lambda_=0.)
res = test_cv(le, X, y)
print "Tocnost:", CA(y, res) #argumenta sta pravi razredi, napovedani
```

Poročajte o točnosti za širok nabor vrednosti lambda, kjer točnost merite:

- s 5-kratnim prečnim preverjanjem in (funkcija `test_cv`) in
- z gradnjo modela in napovedovanjem istih (vseh) primerov (funkcija `test_learning`).

**4. (25 točk)** Ustvarite dve poljubni skupini slik, recimo skupino stolov in skupino miz (vsaj 15 v vsaki skupini; poskrbite, da dodate tudi slike, za katere se vam zdi, da bo ločevanje med razredoma težje). Z orodjem Orange in dodatkom Image Analytics slike pretvorite v vektorje števk (https://youtu.be/Iu8g2Twjn9U) in rezultate shranite v datoteko (gradniki Import Images -> Image Embedding -> Save Data). Z vašo logistično regresijo iz solution.py slike nato poskusite ločevati. Katera vrednost stopnje regularizacije je najbolj primerna za vaše slike? Preglejte širok razpon vrednosti, uporabite prečno preverjanje in poročajte o rezultatih. Pri katerih slikah se vaš model obnese najslabše?

**POMEMBNO**: Pri razvoju metod vam bodo pomagali testi.

**Dodatno (+20 točk)**: Uspešnost klasifikacije merite še s površino pod krivuljo ROC. Implementirajte jo kot funkcijo AUC(real, predictions). Implementirajte jo sami, brez uporabe že izdelanih ROC ali AUC.

**Oddaja**: Oddajte .zip arhiv, ki vsebuje:

1. Datoteko solution.py, kjer ste implementirali manjkajoče funkcije pri logistični regresiji in testiranju. Datoteka mora delovati s testi.
2. Datoteko rezultati.py, kjer ste implementirali izpise in izrise za 2., 3., in 4. podnalogi. V datoteko ne kopirajte kode iz solution.py, ampak izdelane funkcije le uvozite.
3. Izbrane slike in datoteko ali datoteke, kjer so slike predstavljene z vektorji števil (iz 4. dela podnaloge).
4. Poročilo.

Poročilo naj bo napisano s predpisano predlogo.  Pri pisanju poročila ne uporabljajte razdelkov iz predloge, ampak vključite zgolj naslednje razdelke:

1. **Regularizacija.** Vključite izrise za tri zanimive stopnje regularizacije in komentar, kot ga zahteva 2. podnaloga.
2. **Točnosti.** Poročajte točnosti za različne stopnje regularizacije za obe metodi testiranja (v sklopu 3. podnaloge).
3. **Klasifikacija slik**. V poročilo vstavite izbrane slike, točnosti pri različnih stopnjah regularizacije in nekaj za klasifikacijo najtežavnejših slik (zraven pripišite verjetnosti napovedi).

Pri domači nalogi boste potrebovali knjižnjice numpy, scipy in matplotlib.

◄ 3. domača naloga: napovedovanje prihodov avtobusov LPP

Jump to...

5. domača naloga: priporočilni sistemi ►