Relief feature selection

Základní myšlenka

Hlavní myšlenka relief algoritmů spočívá v odhadu vhodnosti atributů na základě toho, jak jsou si jednotlivé hodnoty v různých instancích blízké.

Mějme náhodný prvek z datové sady. Abychom mohli odhadovat vhodnost atributů, nalezneme nejbližší prvek se stejnou třídou jako náš náhodný prvek a nejbližší prvek s opačnou třídou. Těmto prvkům se také často říká „Nearest hit“ a „Nearest miss“.

Po nalezení těchto prvků porovnáváme jednotlivé atributy. Pokud jsou hodnoty pro daný atribut od sebe velmi vzdálené a jedná se o prvek se stejnou třídou, tak to znamená, že tento atribut nehraje příliš velkou roli v určení dané třídy, a tak se kvalita daného atributu sníží. Naopak, pokud budou hodnoty atributů od sebe vzdálené a bude se jednat o prvek s opačnou třídou, je to žádoucí, protože daný atribut od sebe rozlišuje tyto třídy, a tak se kvalita atributu zvýší. Tento proces je opakován m krát, kde m může být buďto počet instancí nebo uživatelem zvolené.

Relief extension

Výše zmíněný postup funguje pouze pro datové sady, kde jsou pouze 2 třídy objektů. Algoritmus sice je možné spustit pro datové sady s vícero třídami objektů, ale výsledky mohou být zkreslené.

Z tohoto důvodu byl původní algoritmus rozšířen tak, aby nebyl limitován jen na 2 třídy objektů, a také, aby dokázal pracovat s chybějícími daty nebo se zkreslenými daty.

Stejně jako dříve, je vybrán náhodný prvek, ale poté je nalezeno k nejbližších prvků ze stejné třídy jako náhodný prvek a k nejbližších prvků pro každou další třídu. Odhad kvality se určuje velmi podobně jako v původním algoritmu, akorát se navíc počítá ještě s pravděpodobnostmi pro výskyt dané třídy a průměrují se hodnoty z k nejbližších prvků.

Použitá implementace

ReliefJava

Tento algoritmus byl získán z githubu. Byl přepsán z původní Java implementace do C# a jedná se o základní verzi reliefu. Přepočet kvality atributu se určuje následující rovnicí:

$$Scores[j] = \frac{\sum_{j=0}^{a} -(hit(i,j)^2) + miss(i,j)^2}{n}$$

Kde n je počet instancí, a je počet atributů, i je konkrétní instance a j je konkrétní atribut

Tento výpočet je poměrně jednoduchý, a tak se výsledky často liší od ostatních algoritmů.

Pseudokód algoritmu:

```
Scores = List(a)
for i < n
      for j < a
            Scores[j] = Scores[j] – Hit(i, j)^2 + Miss(i, j)^2
for i < a
      Scores[i] = Scores[i] / n
```

Pseudokód funkce Miss:

```
Miss(sampleIndex, featureIndex):
	shortestDistance = MAX
	currentDistance = UNASSIGNED
	for i < n
		if class(data[sampleIndex]) != class(data[i])
			currentDistance =
				data[sampleIndex][featureIndex] – data[i][featureIndex]
			if currentDistance^2 < shortestDistance^2
				shortestDistance = currentDistance
	return shortestDistance
```

Pseudokód funkce Hit je téměř totožný, liší se pouze v podmínce:

```
if class(data[sampleIndex]) == class(data[i]) AND sampleIndex != i
```

Relief

Základní verze reliefu vycházející z pseudokódu dostupného zde. Zhruba 7× rychlejší než ReliefJava. Rovnice pro přepočet kvality atributu vypadá následovně:

$$Scores[j] = \sum_{j=0}^{a} -\frac{diff(A[j], i[j], h[j])}{n} + \frac{diff(A[j], i[j], m[j])}{n}$$

Kde n je počet instancí, a je počet atributů, i[j] je konkrétní atribut z instance i, A[j] je konkrétní atribut, h[j] je atribut z nearest hit instance a m[j] je atribut z nearest miss instance.

Funkce diff(a, i, j) vypadá následovně:

Pro numerické hodnoty: $= \frac{|i-j|}{\max(a)-\min(a)}$

Kde max(a) a min(a) jsou největší/nejmenší hodnoty ze všech instancí pro atribut a.

Pro nominální hodnoty funkce vrací 0, pokud se hodnoty rovnají a 1, pokud se hodnoty nerovnají. Poznámka: nominální atributy v současnosti v projektu nejsou implementované.

Pseudokód algoritmu:

```
Scores = List(a)
for i < n
	ri = data[random]
	hit = NearestHit(i)
	miss = NearestMiss(i)
	for j < a
		Scores[j] = Scores[j] – Diff(j, ri, hit) + Diff(j, ri, miss)
for i < a
	Scores[i] = Scores[i] / n
```

Pseudokód funkce Diff:

```
Diff(featureIndex, a, b):
	return Abs(a[featureIndex] – b[featureIndex]) /
		Max(data[featureIndex] – Min(data[featureIndex])
```

Pseudokód funkce NearestMiss:

```
NearestMiss(sampleIndex):
	shortestDistance = MAX
	currentDistance = UNASSIGNED
	index = UNASSIGNED
	for i < n
		if class(data[sampleIndex]) != class(data[i])
			currentDistance = 0
			for j < a
				currentDistance = currentDistance +
data[sampleIndex][featureIndex] – data[i][featureIndex]
				if currentDistance^2 < shortestDistance^2
					shortestDistance = currentDistance
					index = i
	return data[index]
```

Pseudokód funkce NearestHit je téměř totožný, liší se pouze v podmínce:

```
if class(data[sampleIndex]) == class(data[i]) AND sampleIndex != i
```

ReliefF

Rozšířená verze reliefu pracující s více třídami objektů, vycházející z pseudokódu zde. Zhruba stejně rychlý jako ReliefJava, i přes to, že je výpočetně daleko náročnější.
Rovnice pro přepočet kvality atributů vypadá následovně:

$$Scores[j] = \sum_{j=0}^{a} -\frac{diff(A[j], i[j], h[j])}{n \times k} + \sum_{c \neq class(i[j])} \frac{P(c)}{1 - P(class(i[j])} \sum_{m}^{k} \frac{diff(A[j], i[j], m[j, c])}{n \times k}$$

Kde n je počet instancí, a je počet atributů, i[j] je konkrétní atribut z instance i, A[j] je konkrétní atribut, h[j] je atribut z nearest hit instance, m[j,c] je atribut z nearest miss instance pro třídu c a P(c) je pravděpodobnost výskytu instance s třídou c.

Funkce diff(a, i, j) vypadá následovně:

Pro numerické hodnoty: $= \frac{|i-j|}{\max(a) - \min(a)}$

Kde max(a) a min(a) jsou největší/nejmenší hodnoty ze všech instancí pro atribut a.

Pro nominální hodnoty funkce vrací 0, pokud se hodnoty rovnají a 1, pokud se hodnoty nerovnají.

Pseudokód algoritmu:

```
Scores = List(a)
for i < n
	ri = data[random]
	hits = NearestHits(i, k)
	misses = NearestMisses(i, k)
	for j < a
		hDiff = 0
		missDiff = 0
		for l < k
			hDiff = hDiff + Diff(j, ri, hits[l]) / (n * k)
```

```
            for l < classes, class(l) != class(ri)
                  prob = classes[l] / 1 – classes[ri]
                  tmp = 0
                  for o < misses, class(misses[o]) == class(l)
                        tmp = tmp + Diff(j, ri, misses[o] / (n * k)
                  missDiff += prob * tmp
            Scores[j] = Scores[j] – hitsDiff + missDiff
```

Pseudokód funkce NearestMisses:

```
NearestMisses(sampleIndex, k):
      currentDistance = UNASSIGNED
      distances = KeyValues(data.length)
      for i < n
            if class(data[sampleIndex]) != class(data[i])
                  currentDistance = 0
                  for j < a
                        currentDistance = currentDistance +
data[sampleIndex][j] – data[i][j]
                  distances.push(i, Abs(currentDistance))
      return first k smallest values from distances
```

Pseudokód funkce NearestHits je téměř totožný, liší se pouze v podmínce:

```
if class(data[sampleIndex]) == class(data[i] AND sampleIndex != i)
```

ReliefConsistent a ReliefFConsistent

Tyto algoritmy byly upraveny tak, aby nevybíraly náhodný prvek, ale postupně vyhodnotily každou instanci, což vede ke konzistenci výsledků, zatímco algoritmy Relief a ReliefF mohou při opakovaném spuštění vracet rozdílné výsledky, což není žádoucí.

Srovnání výsledků

Pro demonstraci algoritmů byl použit datový soubor WineQT.csv získaný z webového portálu kaggle.com.

Calculation time: 00:00:02.5890709
Algorithm: ReliefJava

Best Column	Score
total sulfur dioxide	0,000293396256332372

Column	Score
total sulfur dioxide	0,000293396256332372
fixed acidity	-6,43372472167975E-05
density	-6,94745732183287E-05
residual sugar	-7,46382354298038E-05
free sulfur dioxide	-8,10282430445804E-05
volatile acidity	-0,000124598857963171
pH	-0,000129804284105506
alcohol	-0,000147017290162858
citric acid	-0,000293350831146107
sulphates	-0,000309061729433772
chlorides	-0,000713233829820927

Calculation time: 00:00:01.1329688
Algorithm: Relief

Best Column	Score
alcohol	0,0801197927182179

Column	Score
alcohol	0,0801197927182179
citric acid	0,0543307086614173
volatile acidity	0,0413745370869737
free sulfur dioxide	0,038860814040036
fixed acidity	0,0357156682848272
density	0,0354060918596635
pH	0,0308278394334567
sulphates	0,0307730994703506
total sulfur dioxide	0,0293938522702331
chlorides	0,0198201435171188
residual sugar	0,0113466124953559

Calculation time: 00:00:01.1016377
Algorithm: ReliefConsistent

Best Column	Score
alcohol	0,078829889853512

Column	Score
alcohol	0,078829889853512
citric acid	0,0539282589676291
volatile acidity	0,042555040209015
free sulfur dioxide	0,0380185685744506
density	0,0357728778763158
total sulfur dioxide	0,0337605767476946
fixed acidity	0,0334006921701158
sulphates	0,0324862086849922
pH	0,0314202850627923
chlorides	0,0164286642800702
residual sugar	0,0163592564628051

Calculation time: 00:00:01.8113620
Algorithm: ReliefF

Best Column	Score
alcohol	0,0515090170101813

Column	Score
alcohol	0,0515090170101813
total sulfur dioxide	0,0191002984618737
citric acid	0,0157368427333504
volatile acidity	0,0152105060991459
density	0,0135586201256381
sulphates	0,0125196165365191
free sulfur dioxide	0,0114517124227154
fixed acidity	0,010068510737395
chlorides	0,00603659064745375
pH	0,001618319094172
residual sugar	0,000814195805026149

Calculation time: 00:00:01.8238476
Algorithm: ReliefFConsistent

Best Column	Score
alcohol	0,0470180887266448

Column	Score
alcohol	0,0470180887266448
total sulfur dioxide	0,0180424502744196
volatile acidity	0,0179424047065012
citric acid	0,0141835662796969
density	0,0135408073785621
sulphates	0,0131079337466254
free sulfur dioxide	0,00952709296468448
fixed acidity	0,00933317321615565
pH	0,00449461014319422
chlorides	0,00346134755882111
residual sugar	0,00216007553345157

Můžeme pozorovat, že kromě algoritmu ReliefJava se všechny shodly na nejlepším prvku, a zároveň odlišnosti v dalších atributech jsou si velice blízké, zatímco rozdíly v porovnání s ReliefJava jsou poměrně razantní, což je způsobeno právě odlišným poměrem ohodnocování kvality atributů.

Dále můžeme pozorovat, že konzistentní algoritmy se neshodují s jejich nekonzistentními verzemi, což je pochopitelné, jelikož se porovnávaný prvek volil náhodně. Pokud bychom vyhodnocení opakovali, dostaneme jiné, avšak podobné výsledky.

Calculation time: 00:00:02.8858381
Algorithm: ReliefJava

Best Column	Score
total sulfur dioxide	0,000293396256332372

Column	Score
total sulfur dioxide	0,000293396256332372
fixed acidity	-6,43372472167975E-05
density	-6,94745732183287E-05
residual sugar	-7,46382354298038E-05
free sulfur dioxide	-8,10282430445804E-05
volatile acidity	-0,000124598857963171
pH	-0,000129804284105506
alcohol	-0,000147017290162858
citric acid	-0,000293350831146107
sulphates	-0,000309061729433772
chlorides	-0,000713233829820927

Calculation time: 00:00:01.3521197
Algorithm: Relief

Best Column	Score
alcohol	0,0798797586199161

Column	Score
alcohol	0,0798797586199161
citric acid	0,052992125984252
volatile acidity	0,043453900454224
fixed acidity	0,0365208773682051
density	0,0363773360929011
sulphates	0,0357709777295802
free sulfur dioxide	0,0348258706467662
total sulfur dioxide	0,0304480491175353
pH	0,0287336130227816
chlorides	0,0245845730051691
residual sugar	0,0179442466951905

Calculation time: 00:00:01.4960650
Algorithm: ReliefConsistent

Best Column	Score
alcohol	0,078829889853512

Column	Score
alcohol	0,078829889853512
citric acid	0,0539282589676291
volatile acidity	0,042555040209015
free sulfur dioxide	0,0380185685744506
density	0,0357728778763158
total sulfur dioxide	0,0337605767476946
fixed acidity	0,0334006921701158
sulphates	0,0324862086849922
pH	0,0314202850627923
chlorides	0,0164286642800702
residual sugar	0,0163592564628051

Calculation time: 00:00:02.4352075
Algorithm: ReliefF

Best Column	Score
alcohol	0,0443163178256122

Column	Score
alcohol	0,0443163178256122
volatile acidity	0,0178838873599592
total sulfur dioxide	0,0167036442606977
density	0,0154489530161145
citric acid	0,0142108233910281
sulphates	0,0141227864689177
fixed acidity	0,0110427788370933
free sulfur dioxide	0,0101849383003415
chlorides	0,00475914167587223
pH	0,00362753519543137
residual sugar	0,0021284287243609

Calculation time: 00:00:02.4331859
Algorithm: ReliefFConsistent

Best Column	Score
alcohol	0,0470180887266448

Column	Score
alcohol	0,0470180887266448
total sulfur dioxide	0,0180424502744196
volatile acidity	0,0179424047065012
citric acid	0,0141835662796969
density	0,0135408073785621
sulphates	0,0131079337466254
free sulfur dioxide	0,00952709296468448
fixed acidity	0,00933317321615565
pH	0,00449461014319422
chlorides	0,00346134755882111
residual sugar	0,00216007553345157