Klasifikace

Pokyny k online schůzce

1. Používejte nejlépe desktopovou aplikaci Teams.

2. Pokud nekladete dotaz nebo se neúčastníte diskuze, mějte prosím vypnutý mikrofon.

3. Pokud jen trochu můžete, mějte puštěnou kameru. Vás výraz tváře pomůže vyučujícímu :)

4. Jak můžete položit dotaz:
   a. Zapněte si mikrofon a rovnou se zeptejte.
      nebo:
   b. Napište dotaz do chatu. nebo:
   c. Použijte tlačítko zvednout ruku.
      (Po vyvolání ruku sundejte)

Agenda

- MNIST dataset
- klasifikační metriky úspěchu
- binární klasifikátory
- klasifikace do více tříd
- analýza chyb klasifikátoru
- speciální případy klasifikace

Klasifikace vs. regrese

- zatím jsme se setkali s regresními úlohami
  - = predikce číselné hodnoty
  - algoritmy: lineární regrese, rozhodovací stromy, náhodné lesy

- klasifikace
  - = predikce třídy, do které daná instance patří
  - algoritmy: SGDClassifier, logistická regrese, rozhodovací stromy, náhodné lesy, kNN, SVM, ...

Dataset MNIST

- 70 tisíc obrázků ručně psaných číslic
  - včetně labelů (tj. která číslice to skutečně je)
- jeden je základních datasetů strojového učení, takové “hello world”
- rovnou k dispozici v Scikit-Learn
- viz jupyter notebook

Binární klasifikátor

- klasifikuje právě do dvou tříd - ano x ne
- v případě MNIST budeme pro začátek trénovat klasifikátor na rozpoznávání jedné konkrétní číslice (např. 5)
- použijeme SGDClassifier
  - Stochastic Gradient Descent
  - lineární klasifikátor
  - výhody:
    - schopný pracovat s velkými datasety
    - každá jedna trénovací instance se bere nezávisle, takže lze použít pro online learning

Metriky úspěchu

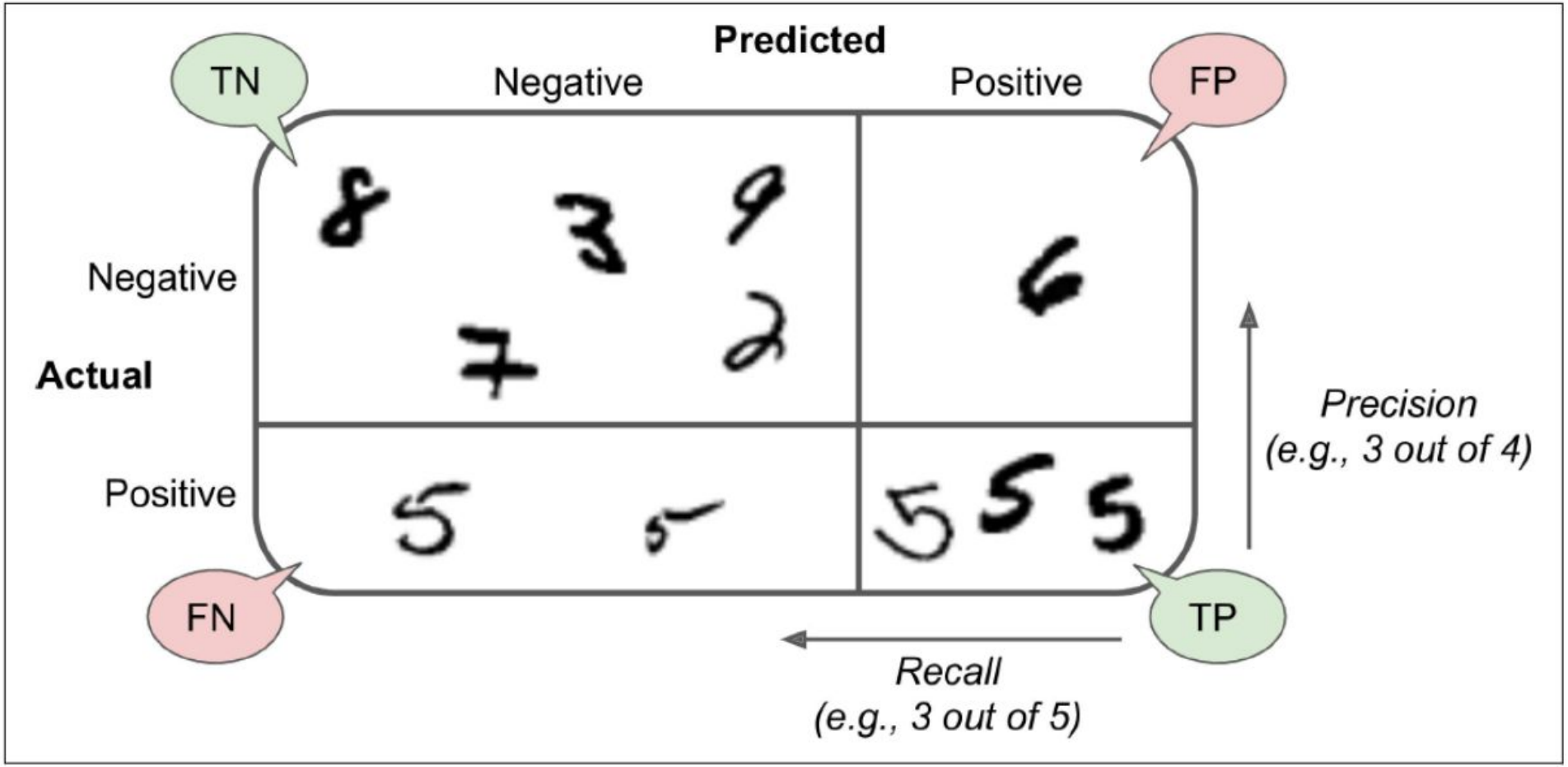
- evaluace klasifikátoru bývá komplikovanější než u regresoru

- K-násobná křížová validace s využitím metriky správnost
  - K-fold cross-validation, metrika accuracy
  - rozdělí trénovací data na K hromádek, trénuje na K-1 hromádkách a ověřuje na zbývající
  - metoda cross_val_score() viz jupyter

- správnost
  - udává podíl správných klasifikací
  - nevhodná metrika, pokud jsou třídy v datasetu nevyvážené

Matice záměn (Confusion Matrix)

- též nazývána jako konfusní nebo chybová matice
- počítá, kolikrát byly instance třídy A chybně klasifikovány jako B
- potřebujeme znát hodnoty všech predikcí a cílové hodnoty
  - mohli bychom využít test set, ale ten bychom si měli nechat opravdu až úplně na konec před spuštěním modelu
  - použijeme tedy opět cross-validaci, nicméně potřebujeme hodnoty predikcí
    - funkce cross_val_predict(), viz jupyter
    - dostaneme “čisté” predikce pro každou instanci, tj. natrénovaný model tuto instanci před klasifikací neviděl
- matici záměn nám jednoduše vytvoří funkce confusion_matrix()



Zdroj: (Géron 2019)

Přesnost a úplnost

- přesnost (precision)
  - tj. správnost pozitivních predikcí
  - přesnost = TP / (TP+FP)
- úplnost (recall)
  - tj. poměr pozitivních instancí nalezených klasifikátorem (sensitivity, True Positive Rate)
  - úplnost = TP / (TP+FN)
- pozor:
  - dosáhnout 100% přesnosti lze jednoduše tak, že vybereme jednu instanci, o které jsme si celkem jistí, že patří do cílové třídy a označíme ji tak, zatímco ostatní instance označíme jako false
  - dosáhnout 100% úplnosti lze jednoduše tak, že všechny instance označíme, že patří do cílové třídy
  - proto se přesnost a úplnost používají téměř výhradně dohromady

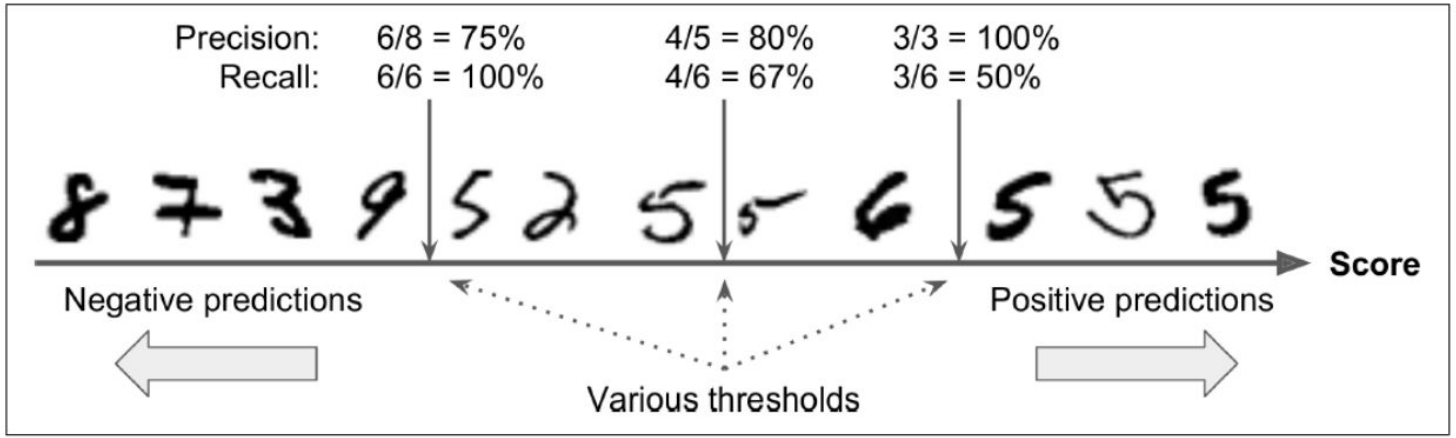
F1 skóre

- jedna číselná hodnota vzniklá z precision a recall vhodná pro porovnání klasifikátorů
- = harmonický průměr precision a recall (malé hodnoty mají velký vliv, tj. nízké precision nebo recall způsobí nízký celkový výsledek)
- F_1 = 2 * (precision * recall) / (precision + recall)
  preferuje klasifikátory, kde precision a recall dosahují podobných hodnot

- čím chceme vyšší precision, tím menší získáme recall a naopak (nejde uměle zvyšovat oboje)
- ne vždy nutně chceme podobné hodnoty precision a recall
  - např. klasifikátor videí, která jsou “bezpečná” pro děti - chceme vysoké precision (tj. nestane se, že dítě uvidí škodlivé video), nízké recall nám nevadí (neuvidí všechna bezpečná videa)
  - např. detekce podezřelých transakcí - chceme vysoké recall (aby neproklouzla škodlivá transakce), nižší precision zas tolik nevadí (falešné poplachy jsou řešitelné)

Precision/Recall Tradeoff

- kompromis mezi přesností a úplností vychází z principu, jak obecně fungují klasifikátory
  - klasifikátor spočítá skóre na základě rozhodovací funkce
  - pokud je skóre vyšší než určitá prahová hodnota (threshold), přiřadí instanci do pozitivní třídy a naopak
  - podle toho, jak budeme posouvat práh, tak můžeme ovlivnit precision resp. recall



Zdroj: (Géron 2019)

ROC křivka

- (receiver operating characteristic, operační charakteristika přijímače)
- zobrazuje True Positive Rate (=recall) oproti False Positive Rate (poměr negativních instancí chybně klasifikovaných jako pozitivní)
    - False Positive Rate = 1 - True Negative Rate (tj. poměr negativních instancí správně klasifikovaných jako negativní)
- viz jupyter
- cílem je, aby křivka byla co nejdál od úhlopříčky (= náhodný klasifikátor)
    - porovnat dva klasifikátory lze tedy tak, že se spočte plocha pod křivkou (AUC)
    - perfektní klasifikátor má ROC AUC 1, náhodný klasifikátor pak ROC AUC 0,5

Kdy jakou metriku použít

- správnost (Accuracy)
  - pokud jsou třídy zastoupeny v datasetu rovnoměrně, pak lze pro jednoduchost použít
- přesnost/úplnost (Precision/Recall)
  - pokud je pozitivní třída v datasetu vzácná nebo pokud nám záleží více na FP než na FN
- ROC AUC
  - v ostatních případech

- v případě MNIST binární klasifikátoru bychom zvolili P/R (ROC AUC vypadá velmi dobře, ale to je zejména díky nevyváženosti tříd)

Klasifikace do více tříd (Multiclass Classification)

- multiclass klasifikátory dokáží klasifikovat do více než dvou tříd
- některé algoritmy (Random Forest, Naive Bayes) dokážou rovnou pracovat jako multiclass
- jiné algoritmy (SVM, lineární klasifikátory) jsou z podstaty binární
- existují však různé postupy, jak vyřešit multiclass klasifikaci pomocí sady binárních klasifikátorů
    - one-versus-all
    - one-versus-one

One-versus-All (OvA)

- pro každou třídu se natrénuje klasifikátor rozpoznávající danou třídu oproti zbytku datasetu (jako jsme si ukazovali)
  - klasifikátor nul, jedniček, dvojek, ...
- pro klasifikaci se vezmou decision score všech klasifikátorů a vybere se ta třída, jejíž klasifikátor má nejvyšší hodnotu score
- z podstaty je třeba trénovat na celém datasetu

One-versus-One (OvO)

- natrénuje se klasifikátor pro každou dvojici tříd
  - nuly oproti jedničkám, nuly oproti dvojkám, jedničky oproti dvojkám, ...
- pro N tříd je potřeba N * (N-1) / 2 klasifikátorů (pro MNIST tedy 45)
- klasifikace se provede tak, že vyhrává ta třída, která “vyhraje nejvíce duelů”
- výhodou je, že každý klasifikátor stačí trénovat jen na té části datasetu, která se týká dané dvojice tříd
  - některé algoritmy, např. SVM, mají problém s velkými datasety, takže pro ně je tato metoda vhodnější
  - natrénovat mnoho klasifikátorů na malém setu je rychlejší než několik málo na velkém

Zdroje

- Coelho, L. P.; Richert, W. (2013) Building machine learning systems with Python. Birmingham: Packt Publishing. ISBN 978-1-78216-140-0.

- Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition. O'Reilly Media, Inc. ISBN 9781492032649.

- Chollet, F. (2019) Deep Learning v jazyku Python. Knihovny Keras, TensorFlow. Grada Publishing, a.s. ISBN 978-80-247-3100-1.

- Segaran, T. (2007) Programming collective intelligence: building smart web 2.0 applications. Beijing: O'Reilly Media. ISBN 0-596-52932-5.