

Tantusz Könyvek – Gépi tanulás

Rövid összefoglaló

A gépi tanulás egy hihetetlen technológia, amelyet ma már gyakrabban használsz, mint gondolnád, és amely még többre lesz képes a jövőben. A gépi tanulás érdekessége, hogy a Python könnyebbé teszi a feladatot, mint azt a legtöbben gondolnák, mivel rengeteg beépített és kiegészítő támogatást nyújt (könyvtárak, adathalmazok és egyéb erőforrások használatával). Ezt szem előtt tartva ebben a rövid összefoglalóban azokat az emlékeztetőket találod meg, amelyek a leggyakrabban szükségesek ahhoz, hogy gyors és könnyű élményben legyen részed a gépi tanulással.

Keresd meg az algoritmust, amelyre szükséged van

A gépi tanulásban számos algoritmust kell használnod a különböző feladatok elvégzéséhez. Nehéz lehet azonban megtalálni azt a konkrét algoritmust, amelyről többet szeretnél megtudni. Az alábbi táblázatban láthatod, hogy hol találsz információkat online a leggyakoribb algoritmusokkal kapcsolatban.

Algoritmus	Típus	Python/R URL-cím
Naiv Bayes-algoritmus	Felügyelt osztályozás, online tanulás	https://scikit-learn.org/stable/modules/naive_bayes.html
PCA	Felügyelet nélküli	https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
SVD	Felügyelet nélküli	https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html
K-közép	Felügyelet nélküli	https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
K-legközelebbi szomszéd	Felügyelt regresszió és osztályozás	https://scikit-learn.org/stable/modules/neighbors.html
Lineáris regresszió	Felügyelt regresszió, online tanulás	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html
Logisztikus regresszió	Felügyelt osztályozás, online tanulás	https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
Neurális hálózatok	Felügyelet nélküli és felügyelt regresszió és osztályozás	https://scikit-learn.org/dev/modules/neural_networks_supervised.html
Tartóvektor-gépek	Felügyelt regresszió és osztályozás	https://scikit-learn.org/stable/modules/svm.html
Adaboost	Felügyelt osztályozás	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
Gradiens növelés (Gradient Boosting)	Felügyelt regresszió és osztályozás	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
Véletlen erdő (Random Forest)	Felügyelt regresszió és osztályozás	https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Válaszd ki a megfelelő algoritmust

A *Tantusz Könyvek – Gépi tanulás* című kötetben rengeteg különböző algoritmussal foglalkozunk, és időnként úgy tűnhet, hogy soha nem fogynak el. Az alábbi táblázatban röviden összefoglaljuk a különböző algoritmusok erősségeit és gyengeségeit.

Algoritmus	Amire a legjobb	Előnyök	Hátrányok
Véletlen erdő (Random Forest)	<ul style="list-style-type: none"> » Szinte bármilyen gépi tanulási problémához megfelelő » Bioinformatika 	<ul style="list-style-type: none"> » Párhuzamosan is futtatható » Ritkán illeszt túl » Automatikusan kezeli a hiányzó értékeket, ha egy speciális számot rendelsz hozzájuk » Nincs szükség semmilyen változó átalakítására » Nincs szükség a paraméterek finomhangolására 	<ul style="list-style-type: none"> » Nehéz értelmezni » Gyengébb regressziónál, amikor a válaszértékek eloszlásának szélsőértékeinél lévő értékeket becsüljük meg » Többosztályos problémáknál a gyakoribb osztályok felé torzít
Gradiens növelés (Gradient Boosting)	<ul style="list-style-type: none"> » Szinte bármilyen gépi tanulási problémához megfelelő » Keresőmotorok (megoldja a rangsorolás megtanulásának problémáját) 	<ul style="list-style-type: none"> » A legtöbb nemlineáris függvényt képes közelíteni » A kategóriájában a legjobb előrejelző » Automatikusan kezeli a hiányzó értékeket » Nincs szükség semmilyen változó átalakítására 	<ul style="list-style-type: none"> » Hajlamos a túlillesztésre, ha túl sok iteráción keresztül futtatjuk » Érzékeny a zajos adatokra és a kiugró értékekre » A paraméterek finomhangolása nélkül nem működik a legjobban
Lineáris regresszió	<ul style="list-style-type: none"> » Alapvető előrejelzések » Ökonometria előjelzések » Marketingre adott válaszok modellezése 	<ul style="list-style-type: none"> » Egyszerűen értelmezhető és elmagyarázható » Ritkán illeszt túl » Az L1 & L2 regularizációval hatékonyan lehet jellemzőket kiválasztani » Gyorsan betanítható » A sztochasztikus változtatának köszönhetően könnyen betanítható big data adatokon 	<ul style="list-style-type: none"> » Kemény munka megoldani, hogy nemlineáris függvényekre illeszkedjen » Gondot okozhatnak neki a kiugró értékek
Tartóvektor-gépek	<ul style="list-style-type: none"> » Karakterfelismerés » Képfelismerés » Szövegosztályozás 	<ul style="list-style-type: none"> » Automatikus nemlineáris jellemzőalkotás » Képes összetett nemlineáris függvényeket közelíteni » Csak a példák egy részét (a tartóvektorokat) használja 	<ul style="list-style-type: none"> » Nehéz értelmezni, ha nemlineáris kerneleket alkalmazunk » Gondot okoz neki, ha túl sok a példa, 10 000 példa felett túl sokáig tart a betanítás
K-legközelebbi szomszéd	<ul style="list-style-type: none"> » Számítógépes látás » Többcímű címkézés » Ajánlórendszerek » Helyesírás-ellenőrzési problémák 	<ul style="list-style-type: none"> » Gyors, lusta betanítás » Természetes módon képes kezelni extrém többosztályos problémákat (pl. szövegek címkézését) 	<ul style="list-style-type: none"> » Az előrejelzési fázisban lassú és nehézkes » Előfordulhat, hogy a dimenzionalitás átka miatt nem tud helyesen előrejelezni

Adaboost	» Arcfelismerés	» Automatikusan kezeli a hiányzó értékeket » Nincs szükség semmilyen változó átalakítására » Nem jut könnyen túlillesztésig » Kevés paramétert kell finomhangolni » Számos különböző gyenge tanuló algoritmust képes hasznosítani	» Érzékeny a zajos adatokra és a kiugró értékekre » Soha nem a legjobb előrejelzést adja az osztályra
Naiv Bayes-algoritmus	» Arcfelismerés » Érzelelemzés » Kéretlen (spam) üzenetek felismerése » Szövegosztályozás	» Könnyen és gyorsan megvalósítható, nem igényel túl sok memóriát, és online tanuláshoz is használható » Könnyen értelmezhető » Figyelembe veszi a korábbi ismereteket	» Erős és irreális feltételezések a jellemzők függetlenségéről » A ritka előfordulásokat nem tudja megbecsülni » Gondot okoznak neki a jelentéktelen jellemzők
Neurális hálózatok	» Képfelismerés » Nyelvfelismerés és fordítás » Beszédfelismerés » Vizuális felismerés	» Bármilyen nemlineáris függvényt képes közelíteni » Ellenáll a kiugró értékekkel szemben » Kép-, szöveg- és hangadatokat is képes kezelni	» Definiálni kell hozzá a hálózat felépítését » Nehezen hangolható a túl sok paraméter miatt, és a hálózat felépítését is definiálni kell » Nehéz értelmezni » Könnyen túlilleszthető
Logisztikus regresszió	» Az eredmények valószínűség szerinti sorba rendezése » Marketingre adott válaszok modellezése	» Egyszerűen értelmezhető és elmagyarázható » Ritkán illeszt túl » Az L1 & L2 regularizációval hatékonyan lehet jellemzőket kiválasztani » A legjobb algoritmus egy esemény valószínűségének előrejelzésére » Gyorsan betanítható » A sztochasztikus változatának köszönhetően könnyen betanítható big data adatokon	» Kemény munka megoldani, hogy nemlineáris függvényekre illeszkedjen » Gondot okozhatnak neki a kiugró értékek
SVD	» Ajánlórendszerek	» Képes értelmesen átstrukturálni az adatokat	» Nehéz értelmezni, hogy az adatokat miért az adott módon strukturálta át
PCA	» Kollinearitás eltávolítása » Az adathalmaz dimenziószámának csökkentése	» Képes csökkenteni az adatok dimenziószámát	» Erős lineáris feltételezéseket foglal magában (a komponensek a jellemzők súlyozott összegei)
K-közép	» Szegmentálás	» Gyorsan talál klasztereket » Több dimenzióban is képes felismerni a kiugró értékeket	» Gondot okoz neki a multikollinearitás » A klaszterek gömb alakúak, más formájú csoportokat nem tud felismerni » Instabil megoldások, az inicializálástól függ

Használd a megfelelő csomagot

Amikor a Pythont használod, előnyödre válik, hogy nem kell újból feltalálnád a spanyol viaszt, ami az algoritmusokat illeti. Elérhető olyan csomag, amely megfelel a konkrét igényeidnek – csak tudnod kell, hogy melyiket használd. A következő listában a Python gyakran használt csomagjait találod. Ha bármilyen algoritmussal kapcsolatos feladatot szeretnél elvégezni, egyszerűen csak töltsd be az adott feladathoz szükséges csomagot a programozási környezetedbe.

- » **Adaboost:** `sklearn.ensemble.AdaBoostClassifier` és `sklearn.ensemble.AdaBoostRegressor`
- » **Gradiens növelés (Gradient Boosting):** `sklearn.ensemble.GradientBoostingClassifier` és `sklearn.ensemble.GradientBoostingRegressor`
- » **K-közép:** `sklearn.cluster.KMeans` és `sklearn.cluster.MinibatchKMeans`
- » **K-legközelebbi szomszéd:** `sklearn.neighbors.KNeighborsClassifier` és `sklearn.neighbors.KNeighborsRegressor`
- » **Lineáris regresszió:** `sklearn.linear_model.LinearRegression`, `sklearn.linear_model.Ridge`, `sklearn.linear_model.Lasso`, `sklearn.linear_model.ElasticNet` és `sklearn.linear_model.SGDRegressor`
- » **Logisztikus regresszió:** `sklearn.linear_model.LogisticRegression` és `sklearn.linear_model.SGDClassifier`
- » **Naiv Bayes-algoritmus:** `sklearn.naive_bayes.GaussianNB`, `sklearn.naive_bayes.MultinomialNB` és `sklearn.naive_bayes.BernoulliNB`
- » **Neurális hálózatok:** `tensorflow.keras`
- » **Főkomponens-analízis (PCA):** `sklearn.decomposition.PCA`
- » **Véletlen erdő (Random Forest):** `sklearn.ensemble.RandomForestClassifier`, `sklearn.ensemble.RandomForestRegressor`, `sklearn.ensemble.ExtraTreesClassifier` és `sklearn.ensemble.ExtraTreesRegressor`
- » **Tartóvektor-gépek (SVM-ek):** `sklearn.svm.SVC`, `sklearn.svm.LinearSVC`, `sklearn.svm.NuSVC`, `sklearn.svm.SVR`, `sklearn.svm.LinearSVR`, `sklearn.svm.NuSVR` és `sklearn.svm.OneClassSVM`
- » **Szinguláris értékbontás (SVD):** `sklearn.decomposition.TruncatedSVD` és `sklearn.decomposition.NMF`

Különböztessd meg a tanulás típusait

Az algoritmusok állítólag tanulnak, de azt fontos tudnod, hogy hogyan tanulnak, mert egészen biztosan nem úgy, ahogy az emberek. A tanulásnak számos különböző fajtája van, az algoritmustól és annak céljaitól függően. A gépi tanulási algoritmusokat céljuk alapján három fő csoportra oszthatjuk:

- » **Felügyelt tanulás:** Erről akkor beszélünk, amikor egy algoritmus példaadatokról és a hozzájuk tartozó célválaszokról tanul, amelyek állhatnak numerikus értékekből vagy szöveges besorolásokból, például osztályokból vagy címkékből, hogy később, amikor új példákkal találkozunk, előre tudja jelezni a helyes választ. A felügyelt megközelítés valóban hasonlít a tanár felügyelete mellett történő emberi tanuláshoz. A tanár jó példákat ad a tanulónak, hogy megjegyezze azokat, a tanuló pedig általános szabályokat vezet le ezekből a konkrét példákból.
- » **Felügyelet nélküli tanulás:** Erről akkor beszélünk, amikor egy algoritmus egyszerű példákból tanul, mindenféle kapcsolódó válasz nélkül, az algoritmusra bízva, hogy önállóan határozza meg a mintázatokat az adatokban. Az ilyen típusú algoritmusok hajlamosak az adatokat más formába átstrukturálni, például új adatjellemzőkké, amelyek egy osztályt vagy néhány új értéket ábrázolhatnak, és hasznosak lehetnek a további elemzéshez vagy egy előrejelző modell betanításához.
- » **Megerősítéssel tanulás:** Erről akkor beszélünk, amikor egymás után mutatunk az algoritmusnak olyan példákat, amelyeknek nincsenek címkéi, pont úgy, mint a felügyelet nélküli tanulásnál. Itt azonban minden egyes példához pozitív vagy negatív visszajelzést adunk az algoritmus által javasolt megoldás alapján. A megerősítéssel tanulás olyan alkalmazási területekhez kapcsolódik, amelyeken az algoritmusnak döntéseket kell hoznia (tehát az eredmény előíró, és nem csak leíró jellegű, mint a felügyelet nélküli tanulásnál), és a döntések következményekkel járnak.