An Adaptive nearest neighbor rule for classification

Nicholas Aldino Previtali MAT. 1046394

Sessione Estiva (24/07) 2019/2020

1. Contesto

Il lavoro si pone all'interno dell'ambito del machine learning, e in particolar modo discute una variante dell'algoritmo *k-nearest neighbor*. Il suo funzionamento si basa sulla somiglianza delle caratteristiche: più un'istanza è vicina a un data point, più il *k-NN* li considererà simili. Solitamente la somiglianza viene calcolata tramite la distanza euclidea (o un qualche altro tipo di distanza a seconda del problema in esame). Minore sarà la distanza e maggiore sarà la somiglianza tra data point e l'istanza da prevedere. Oltre alla distanza, l'algoritmo prevede di fissare un parametro *k*, scelto arbitrariamente, che identifica il numero di data points più vicini. L'algoritmo valuta le *k* minime distanze così ottenute. La classe che ottiene il maggior numero di queste distanze è scelta come previsione.

Il *K-NN* è uno strumento non parametrico, ossia non fa alcuna ipotesi sulla distribuzione dei dati che analizza. In altre parole, la struttura del modello è determinata dai dati ed è piuttosto utile, perché nel "mondo reale", la maggior parte dei dati non obbedisce ai tipici assunti teorici. Di conseguenza, *K-NN* potrebbe e probabilmente dovrebbe essere una delle prime scelte per uno studio di classificazione quando c'è poca o nessuna conoscenza precedente sulla distribuzione dei dati. In aggiunta, si può affermare che la fase di training è piuttosto veloce. La mancanza di generalizzazione fa sì che *K-NN* conservi tutti i dati di allenamento. Ciò significa che tutti i dati di addestramento (o la maggior parte di essi) sono necessari durante la fase di test.

Generalmente, il *k-NN* può essere utilizzato sia per problemi di classificazione sia per problemi di regressione. Nei problemi di classificazione, il KNN determina l'etichetta di classe prevista, valutando un tipo di distanza. Tra le più utilizzate, oltre alla distanza

euclidea, si ha la distanza di Hamming, la distanza di Manhattan o la distanza Minkowsky. Dopo avere calcolato la distanza, viene restituita l'etichetta della classe di maggioranza dell'insieme delle istanze k selezionate. Nella regressione, invece, il KNN sceglie il valore medio dei valori dei vicini più prossimi come valore previsto.

KNN può richiedere molta memoria o spazio per archiviare tutti i dati, ma esegue solo un calcolo (o impara) quando è necessaria una previsione (per questo motivo si dice che *l'algoritmo è pigro*). È inoltre possibile aggiornare e curare le istanze di allenamento nel tempo per mantenere accurate le previsioni.

C'è comunque da dire che l'idea di distanza o vicinanza può scomporre in dimensioni molto elevate (molte variabili di input) che possono influire negativamente sulle prestazioni dell'algoritmo sul problema di interesse. Questo fatto viene chiamato *curse of dimensionality.*

Importante è il potere predittivo, che dipende dal parametro k scelto inizialmente. Quando *k è piccolo*, stiamo limitando la regione di una determinata previsione e costringendo il nostro classificatore ad essere "più cieco" rispetto alla distribuzione generale.

Al contrario, un *k grande* riduce l'impatto della varianza causato da un errore casuale, ma corre il rischio di ignorare piccoli dettagli che potrebbero essere rilevanti. Una scelta adattiva di *k* potrebbe migliorare decisamente la situazione, andando ad aumentare l'accuratezza, riducendo di conseguenza possibili errori.

In questa variante il parametro *k* non viene quindi settato come parametro, bensì viene scelto in modo adattivo. La scelta di questo parametro non è casuale, dipende dalle

caratteristiche di ogni *neighborhood* e potrebbe quindi variare, anche di molto, da punto a punto. L'algoritmo utilizzerà valori abbastanza alti di *k* per fare la predizione delle classi dei punti in regioni affollate, quindi affette da rumore. Nel lavoro preso in esame viene dimostrato teoricamente e nella pratica come questa variante riesca ad approssimare molto bene (e talvolta con prestazioni superiori) il classico algoritmo *k-NN*. In particolare viene analizzato il rate di convergenza di questo classificatore adattivo in base ad una quantità denominata *advantage*. Questa analisi utilizza una variante del teorema di convergenza uniforme di Vapnik-Chervonenkis, il quale serve per produrre stime empiriche di probabilità condizionali.

2. Related Works

2.1 Paper related works

L. Devroye, L. Györfi, and G. Lugosi. A Probabilistic Theory of Pattern Recognition. Springer, 1996

Viene analizzato il capitolo 7 di questo libro, riguardante lo *slow rate of convergence.* Per i separatori lineari e molte altre famiglie parametriche di classificatori vengono discussi tassi di convergenza che valgono senza alcuna ipotesi sulla distribuzione di input μ o sulla funzione di aspettativa condizionale η. Si dimostra che non è vero per la stima non parametrica: sebbene in linea di principio sia possibile acquisire qualsiasi funzione target, il numero di campioni necessari per raggiungere un livello di precisione specifico dipenderà inevitabilmente da aspetti di questa funzione come la velocità con cui essa cambia. In particolar modo ci si è concentrati (1) sulla consistenza asintotica, idealmente senza ipotesi, e (2) sui tassi di convergenza sotto una varietà di ipotesi di *smoothness*.

T. Cover and P.E. Hart. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13:21–27, 1967.

Questo articolo discute nel dettaglio l'algoritmo *k-NN,* in particolar modo la nozione di consistenza asintotica, nel caso in cui *k* possa crescere con il crescere del numero di punti dati *n*. Il rischio del classificatore, indicato con *Rn*, è il suo tasso di errore sulla distribuzione P sottostante; questa è una variabile casuale che dipende dall'insieme di punti di training visti. Le pagine esaminate in questo articolo hanno mostrato che negli spazi metrici generali, supponendo che ogni x nel supporto di μ sia un punto di continuità di *η* o abbia $\mu(\{x\}) > 0$, il rischio atteso $\mathbb{E}(R_n)$ converge nel rischio ottimale di Bayes *R**, purché:

- $k \to \infty$
- $k/n \to 0$

C. Stone. Consistent non parametric regression. Annals of Statistics, 5:595–645, 1977

Viene discussa la regressione consistente non parametrica, in particolar modo per punti in uno spazio euclideo finito. Il risultato ottenuto viene utilizzato insieme a quelli di articoli più recenti che verranno citati di seguito. Questo risultato consiste nella dimostrazione della consistenza del classificatore *k-NN* senza fare assunzioni riguardo *η* o μ. Questa viene chiamata *consistenza universale* del classificatore, dove la sua probabilità di errore converge a quello Bayesiano, per qualsiasi distribuzione *(X, Y)*.

L. Devroye, L. Györfi, A. Krzyzak, and G. Lugosi. On the strong universal consistency of nearest neighbor regression function estimates. Annals of Statistics, 22:1371–1385,1994

Questo articolo viene utilizzato insieme al risultato di *Stone* citato precedentemente. Sono presentati alcuni risultati riguardo la consistenza della stima di regressione dell'algoritmo *k-NN.* Vengono presentati due risultati riguardanti la coerenza della stima di regressione del vicino k più vicino. Viene mostrato che tutte le modalità di convergenza in L1 (in probabilità, quasi sicuramente, complete) sono equivalenti se la variabile di regressione è limitata. Sotto la condizione aggiuntiva $\frac{k}{\log n} \to \infty$ viene ottenuta la *consistenza universale forte della stima*.

F. Cerou and A. Guyader. Nearest neighbor classification in infinite dimension. ESAIM: Probability and Statistics, 10:340–355, 2006.

Viene ripreso il risultato di *Stone*, mostrando in questo documento che questo risultato non è più valido negli spazi metrici generali. Tuttavia, se lo spazio metrico (F, d) è

separabile e se si presume una condizione di regolarità, il classificatore *k-NN* è debolmente consistente.

S. Kulkarni and S. Posner. Rates of convergence of nearest neighbor estimation under arbitrary sampling. IEEE Transactions on Information Theory, 41(4):1028–1039, 1995

Anche in questo articolo vengono discussi i tassi di convergenza per la stima del *k-NN,* i quali sono stabiliti in un quadro generale in termini di metriche di copertura dello spazio sottostante. Il primo risultato è trovare limiti superiori espliciti del campione finito per il classico problema di campionamento casuale indipendente e identicamente distribuito *(i.i.d.)* in un'impostazione di spazio metrico separabile.

L. Györfi. The rate of convergence of k-nn regression estimates and classification rules. IEEE Transactions on Information Theory, 27(3):362–364, 1981

E. Mammen and A.B. Tsybakov. Smooth discrimination analysis. The Annals of Statistics, 27(6):1808–1829, 1999.

J.-Y. Audibert and A.B. Tsybakov. Fast learning rates for plug-in classifiers. Annals of Statistics, 35(2):608–633, 2007

Questi tre articoli discutono ampiamente i tassi di convergenza, i quali sono stati ottenuti con la classificazione *k-NN* in varie condizioni di *smoothness*, tra cui le *condizioni di Holder* e le condizioni del *margine di Tsybakov*. Tali ipotesi sono diventate consuete nelle statistiche non parametriche, ma lasciano molto a desiderare.

Innanzitutto, non sono verificabili: non è possibile determinare empiricamente la *smoothness* fornita dai campioni. In secondo luogo, visualizzano la distribuzione sottostante *P* attraverso la minuscola finestra di due o tre parametri, oscurando quasi tutta la struttura rimanente della distribuzione che influenza anche il tasso di convergenza. Infine, poiché la stima non parametrica è spesso locale, esiste la possibilità di ottenere diversi tassi di convergenza in diverse regioni dello spazio di input: una possibilità che viene immediatamente sconfitta riducendo l'intero spazio a due costanti di *smoothness*.

K. Chaudhuri and S. Dasgupta. Rates of convergence for nearest neighbor classification. In Advances in Neural Information Processing Systems, pages 3437–3445. 2014.

In questo articolo viene analizzato il comportamento del *k-NN* negli spazi metrici e vengono forniti tassi di convergenza a campione finito e dipendenti dalla distribuzione con le sole ipotesi essenziali. Questi sono più generali dei limiti esistenti e consentono di stabilire la *coerenza universale* del *k-NN* in una gamma più ampia di spazi di dati rispetto a quanto precedentemente noto. Sono illustrati i limiti superiore e inferiore introducendo una nuova classe di *ssmoothness* personalizzata per *k-NN*. Si scopre, ad esempio, che sotto il margine di *Tsybakov* il tasso di convergenza del *k-NN* corrisponde ai limiti inferiori stabiliti per la classificazione non parametrica.

S. Kpotufe. k-nn regression adapts to local intrinsic dimension. In Neural Information Processing Systems, 2011.

Viene mostrato che la regressione di *k-NN* è anche adattativa alla dimensione intrinseca. In particolare, localmente per una query *x*, i tassi di convergenza dipendono solo dal modo in cui le masse di circonferenze centrate su x variano con il raggio. Inoltre, viene mostrato un modo semplice per scegliere $k = k(x)$ localmente su qualsiasi *x* in modo da raggiungere quasi la velocità minimax su *x* in termini di dimensione intrinseca sconosciuta in prossimità di *x*. E' il primo articolo che discute una variante adattiva del *k-NN.*

G.H. Chen and D. Shah. Explaining the Success of Nearest Neighbor Methods in Prediction. Foundations and Trends in Machine Learning. NOW Publishers, 2018

Viene ampiamente discusso l'algoritmo *k-NN,* con riferimento a possibili sue applicazioni mediche.

S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification: A survey of some recent advances. ESAIM: probability and statistics, 9:323–375, 2005

Descrizione dettagliata teorica e pratica nell'ambito di *pattern classification*. Si descrivono modello base, la minimizzazione del rischio empirico e di funzioni di costo, analisi di vincoli di margine.

Tabula Muris Consortium et al. Single-cell transcriptomics of 20 mouse organs creates a tabula muris. Nature, 562(7727):367, 2018

In questo lavoro viene presentato un compendio di dati trascrittomici a singola cellula dall'organismo modello Mus musculus che comprende oltre 100.000 cellule di 20 organi e tessuti. Questi dati rappresentano una nuova risorsa per la biologia cellulare, rivelano

l'espressione genica in popolazioni cellulari scarsamente caratterizzate e consentono il confronto diretto e controllato dell'espressione genica in tipi di cellule condivise tra tessuti, come i linfociti T e le cellule endoteliali da diverse posizioni anatomiche. Questo modello viene in parte utilizzato per trovare una possibile soluzione al *K-NN* e quindi anche alla sua versione adattiva.

K. Chaudhuri and S. Dasgupta. Rates of convergence for the cluster tree. In Advances in Neural Information Processing Systems, pages 343–351, 2010

In questo articolo si discutono procedure di clustering il cui output su un campione finito converge in "cluster naturali" della distribuzione sottostante f. Vi sono senza dubbio molti modi significativi per definire i cluster naturali. Qui vengono utilizzati alcuni primi lavori sul clustering associando i cluster a regioni connesse ad alta densità. In particolare, un cluster di densità f è qualsiasi componente collegato di $\{x: f(x) \geq \lambda\}$, per qualsiasi $\lambda > 0$. La raccolta di tutti questi cluster forma una gerarchia (infinita) chiamata albero dei cluster, descritto nel corso del lavoro.

W. Dong, M. Charikar, and K. Li. Efficient k-nearest neighbor graph construction forgeneric similarity measures. InProceedings of the 20th international conference onWorld wide web, pages 577–586. ACM, 2011.

Si discute la costruzione del *K-NNG (K-Nearby Neighbor Graph)*, una importante operazione con molte applicazioni in particolar modo nell'ambito dell'intelligenza artificiale. I metodi esistenti per la costruzione di K-NNG non si adattano bene per determinate misure di somiglianza. Viene presentato *NN-Descent*, un algoritmo semplice ma efficace per la costruzione approssimativa di *K-NNG* con misure di

somiglianza arbitrarie. Il metodo si basa sulla ricerca locale, ha un sovraccarico di spazio minimo e non si basa su alcun indice globale condiviso. Pertanto, è particolarmente adatto per applicazioni su larga scala in cui le strutture di dati devono essere distribuite sulla rete.

R.M. Dudley. Balls in Rk do not cut all subsets ofk+2points.Advances in Mathematics,31(3):306–308, 1979

Viene dimostrato che per ogni set *E* di $k + 2$ punti in R^k, $k = 1, 2,\ldots$, non tutti i subset di *E* hanno forma $B \cap E$ dove *E* è una circonferenza. Questa dimostrazione viene utilizzata nel perfezionamento dell'*AKNN.*

E. Fix and J. Hodges. Discriminatory analysis, nonparametric discrimination. USA F School of Aviation Medicine, Randolph Field, Texas, Project 21-49-004, Report 4, Con-tract AD41(128)-31, 1951.

Viene discusso come nel *supervised learning*, il numero sbilanciato di istanze tra le classi in un set di dati può rendere gli algoritmi per classificare un'istanza della classe di minoranza come una della classe di maggioranza. Con l'obiettivo di risolvere questo problema, l'algoritmo KNN fornisce una base per altri metodi di bilanciamento. Questi metodi di bilanciamento vengono rivisitati in questo lavoro e viene proposto un nuovo e semplice approccio al sottocampionamento KNN. Gli esperimenti hanno dimostrato che il metodo di sottocampionamento KNN ha superato gli altri metodi di campionamento. Il metodo proposto ha anche sovraperformato i risultati di altri studi e indica che la semplicità di KNN può essere utilizzata come base per algoritmi efficienti di apprendimento automatico.

J. Heinonen. Lectures on Analysis on Metric Spaces. Springer, 2001

Libri utilizzato per riprendere concetti matematici riguardanti *metric spaces* utilizzati poi in alcune dimostrazioni.

MNIST dataset. http://yann.lecun.com/exdb/mnist/, 1996

Mouse cell atlas dataset.ftp://ngs.sanger.ac.uk/production/teichmann/BBKNN/MouseAtlas.zip, 2018. Accessed: 2019-05-02

notMNIST dataset.http://yaroslavb.com/upload/notMNIST/, 2011. Accessed:2019-05-02.

M. Raab and A. Steger. Balls into bins - a simple and tight analysis. In Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings, pages 159–170,1998.

Il problema delle *balls into bins* è un classico problema nella teoria della probabilità che ha molte applicazioni nell'informatica. Il problema riguarda m palle e n caselle (o "bidoni"). Ogni volta, una singola palla viene posizionata in uno dei contenitori. Dopo che tutte le palle sono nei cassonetti, osserviamo il numero di palline in ogni cestino; chiamiamo questo numero *load* sul cestino e si vuole sapere qual è il carico massimo su un singolo cestino. In questo caso si studia il problema legandolo all'algoritmo K-NN, sfruttando il concetto di *features.*

V.N. Vapnik and A.Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. Theory of Probability & Its Applications, 16(2):264–280,1971

2.2 Adaptive K-NN related works (non presenti nel paper)

Li Baoli, Lu Qin, Yu Shiwen. An adaptive k-nearest neighbor text categorization strategy. ACM Transactions on Asian Language Information Processing. 2004

k è il parametro più importante in un sistema di categorizzazione del testo basato sull'algoritmo *kNN*. Per classificare un nuovo documento, vengono determinati per primi i documenti k-più vicini nel training set. La previsione delle categorie per questo documento può quindi essere effettuata in base alla distribuzione delle categorie tra i k vicini più vicini. In generale, la distribuzione delle classi in un training set non è uniforme; alcune classi possono avere più campioni di altre. Le prestazioni del sistema sono molto sensibili alla scelta del parametro k. Ed è molto probabile che un valore k fisso comporti una distorsione per grandi categorie e non farà pieno uso delle informazioni nel training set. Per far fronte a questi problemi, in questo articolo viene proposta una strategia *kNN* migliorata, in cui vengono utilizzati diversi numeri di vicini più vicini per categorie diverse anziché un numero fisso in tutte le categorie. Verranno utilizzati più campioni (vicini più vicini) per decidere se un documento di prova deve essere classificato in una categoria che ha più campioni nel training set. I numeri dei vicini più vicini selezionati per le diverse categorie sono adattativi alla loro dimensione del campione nel training set. La strategia

è particolarmente applicabile e promettente per i casi in cui non è possibile stimare il parametro k e quando la distribuzione delle classi di un training set è distorta.

Link: https://dl.acm.org/doi/abs/10.1145/1039621.1039623

Stefanos Ougiaroglou, Alexandros Nanopoulos, Apostolos N. Papadopoulos, Yannis Manolopoulos, Tatjana Welzer-Druzovec. Adaptive k-Nearest-Neighbor Classification Using a Dynamic Number of Nearest Neighbors. ADBIS 2007: Advances in Databases and Information Systems pp 66-82. 2007

Se la dimensione del set di dati è grande, la ricerca sequenziale o binaria di NN è inapplicabile a causa dell'aumento dei costi di calcolo. Pertanto, gli schemi di indicizzazione vengono spesso utilizzati per accelerare il processo di classificazione. Se il numero richiesto di vicini più vicini è elevato, l'uso di un indice potrebbe non essere adeguato per ottenere prestazioni elevate. In questo documento, viene dimostrato che l'esecuzione dell'algoritmo *kNN* può essere interrotta se alcuni criteri sono soddisfatti. In questo modo, una decisione può essere presa senza il calcolo di tutti i *k* vicini più vicini di un nuovo oggetto. Tre diverse euristiche sono studiate per migliorare l'algoritmo del vicino più vicino con una capacità di interruzione anticipata. Tali euristiche mirano a: (i) ridurre il più possibile i costi di calcolo e I / O e (ii) mantenere l'accuratezza della classificazione ad alto livello. I risultati sperimentali basati su set di dati della vita reale illustrano l'applicabilità del metodo proposto per ottenere prestazioni migliori rispetto ai metodi esistenti.

Link: https://link.springer.com/chapter/10.1007/978-3-540-75185-4_7

J Wang, P Neskovic, LN Cooper. Improving nearest neghbor rule with a simple adaptive distance measure. Pattern Recognition Letters, 2007 – Elsevier

kNN è uno degli algoritmi di classificazione dei modelli più semplici e più attraenti. Tuttavia, deve affrontare serie sfide quando modelli di classi diverse si sovrappongono in alcune aree dello spazio. In passato, molti ricercatori hanno sviluppato varie metriche adattive o discriminanti per migliorarne le prestazioni. In questo documento, viene dimostrato che una misura adattativa della distanza estremamente semplice migliora significativamente le prestazioni della regola del *kNN*.

3. Descrizione

3.1 Introduzione all'algoritmo

Introduciamo una regola adattiva di classificazione nearest neighbor. Dato un training set con etichette {± 1}, la sua previsione in corrispondenza di un punto ignoto *x* si basa sui punti di training più vicini a *x*, piuttosto che sulla regola del *kNN*. Tuttavia, il valore di *k* che si utilizza può variare da query a query. In particolare, se ci sono *n* punti di allenamento, quindi per qualsiasi query *x*, viene cercato il *k* più piccolo per il quale i *k* punti più vicini a *x* hanno etichette la cui media è maggiore di *Δ(n, K)*, in questo caso la predizione è +1, altrimenti se la media è minore di - *Δ(n, K)* la predizione sarà -1. Se non esiste tale *k* viene ritornato il valore *"?"*.

In questo caso,

$$\Delta(n,k) \sim \sqrt{(\log n)/k}$$

corrisponde a un intervallo di confidenza per l'etichetta media nell'area intorno alla query.

Studiamo questa regola nel quadro statistico standard in cui tutti i dati sono i.i.d. presi da una distribuzione sottostante sconosciuta P in *X*↠*Y*, dove *X* è lo spazio dei dati e *Y* è lo spazio della label. Consideriamo *X* uno spazio metrico separabile, con la funzione di distanza *d:* $X \times X \to \mathbb{R}$ e prendiamo $y = \{\pm 1\}$. Possiamo decomporre *P* nella distribuzione marginale *μ* su *X* e l'aspettativa condizionale della label per ogni *x*: se *(X, Y)* rappresenta un'estrazione random da *P,* definiamo $\eta(x) = \mathbb{E}(Y|X = x)$. In questa terminologia, il classificatore ottimo di Bayes è la regola g*: X ⇥ {±1} data da:

$$g^*(x) = \begin{cases} sign(\eta(x)) & \eta(x) \neq 0 \\ -1 \; o + 1, & \eta(x) = 0 \end{cases} \qquad (1)$$

e il suo error rate è il Bayes risk, R* = $\frac{1}{2}E_{x\sim\mu}[1-|\eta(X)|]$. È noto che vari schemi di classificazione non parametrici presentano tassi di errore che convergono asintoticamente a R*.

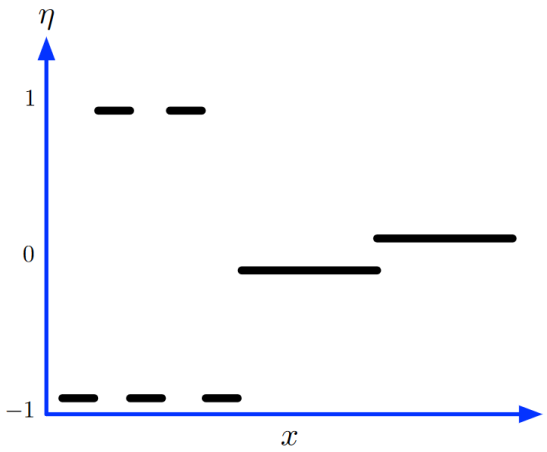


Figura 1: per i valori di *x* presenti nella metà a sinistra dell'intervallo, il pointwise bias *η(x)* è vicino a +1 o -1, e quindi un basso valore di *k* porta ad una buona predizione. Un valore di *k* maggiore non porterà ad una predizione accurata, poiché si potrebbe andare in regioni vicine ma con label differente. Al contrario per i valori di *x* presenti nella metà destra dell'intervallo proposto.

In questo documento, ci si concentra sulla coerenza e ai tassi di convergenza. In particolare, si scopre che la regola adattiva del *kNN* è anche asintoticamente coerente (con le stesse condizioni tecniche) mentre converge a un ritmo che è circa buono, e talvolta significativamente migliore, quello di *kNN* in qualsiasi altro algoritmo.

Intuitivamente, uno dei vantaggi di *kNN* rispetto ai classificatori non parametrici che utilizzano un raggio di larghezza di banda fisso, come ad esempio la *finestra di Parzen* o

gli stimatori della densità del kernel, è che *kNN* si adatta automaticamente alla variazione della distribuzione marginale μ: nelle regioni con grande μ, i *k* vicini più vicini si trovano vicino al punto di ricerca, mentre nelle regioni con piccoli μ, i k vicini più vicini possono essere più lontani. La regola adattativa *NN* che si propone va oltre: si adatta anche alla variazione di μ. In alcune regioni dello spazio di input, dove μ è vicino a 0, una previsione accurata avrebbe bisogno di *k* di grandi dimensioni. In altre regioni, dove μ è vicino a 0 una piccola *k* sarebbe sufficiente e, in effetti, una *k* più grande potrebbe essere dannosa perché le regioni vicine potrebbero essere etichettate diversamente. Un classificatore *kNN* è costretto a scegliere un singolo valore di *k* che viene scambiato tra queste due contingenze. La regola *NN* adattativa, tuttavia, può scegliere la *k* giusta in ciascuna zona separatamente.

Questo stimatore consente di fornire tassi di convergenza più stretti e più trasparenti di quelli normalmente ottenuti nelle statistiche non parametriche. In particolare, per qualsiasi punto *x* nello spazio dell'istanza *X*, viene definita una nozione del vantaggio in *x*, indicato con *adv(x)*, che è piuttosto simile a un margine locale. Viene mostrato che la previsione in *x* è molto probabile che sia corretta una volta che il numero di punti di training supera *O(1/adv(x))*.

3.2 Setup

Si consideri lo spazio delle istanze come spazio metrico separabile (X, d) e lo spazio delle labels deve essere Y = {± 1}. Si presume che tutti i dati vengano estratti i.i.d. da una distribuzione sconosciuta fissa P su X⇥Y.

Definiamo μ che descrive la distribuzione marginale su X: se (X, Y) è una estrazione casuale da P, allora:

$$\mu(S) = \Pr(X \in S)$$

per ogni set misurabile $S\ contenuto\ in\ X$. Per ogni $x \in X$, il *bias* di *Y* dato *x* è:

$$\eta(x) = E(Y|X = x) \in [-1, +1]$$

Similmente, per ogni set misurabile con $\mu(S) > 0$, il bias di Y dato $X \in S$ è:

$$\eta(S) = E(Y|X \in S) = \frac{1}{\mu(S)} \int_S \eta(x) d\mu(x)$$

Il rischio del classificatore $g: X \to \{-1, +1, ?\}$ è la probabilità che sia incorretto in coppie $(X, Y) \sim P$:

$$R(g) = P(\{(x, y): g(x) \neq y\}) \qquad (2)$$

Il classificatore ottimo di Bayes g^* come visto dalla (1) dipende solo da η, ma il suo rischio R* dipende da μ. Per un classificatore g_n basato su *n* punti di training da P, saremmo interessati a sapere se $R(gn)$ converge a R^*, e anche a conoscere il rate con il quale questo valore converge. L'algoritmo e l'analisi dipendono fortemente dalle masse di probabilità e dai bias delle circonferenze in X.

Per $x \in X\ e\ r \geq 0$, si definisce $B(x, r)$ la circonferenza di raggio r più vicina centrata in x,

$$B(x,r) = \{z \in X : d(x,z) < r\}$$

Per $0 \leq p \leq 1$, definiamo $r_p(x)$ il minor raggio r tale che $B(x,r)$ abbia probabilità di massa almeno p, ovvero:

$$r_p(x) = \inf\{r \geq 0 : \mu(B(x,r)) \geq p\} \qquad (3)$$

Segue che $\mu(B(x, r_p(x))) \geq p$

Il supporto della distribuzione marginale μ gioca un ruolo importante nelle prove di convergenza e viene formalmente definito come:

$$supp(\mu) = \{x \in X : \mu(B(x,r)) > 0 \; per \; ogni \; r > 0\}$$

3.3 L'algoritmo A-KNN

Viene dato all'algoritmo un *labeled training set* $(x_1, y_1) \ldots (x_n, y_n) \in X \times Y$. Basandosi su questi punti, l'algoritmo è in grado di computare stime empiriche delle probabilità e dei bias delle diverse circonferenze.

Per ogni set $S \; contenuto \; in \; X$ vengono definiti contatore empirico e probabilità di massa come:

$$\#_n(S) = |\{i : x_i \in S\}| \qquad (4)$$

$$\mu_n(S) = \frac{\sum_{i:xi \in S} y_i}{\#_n(S)} \qquad (5)$$

AKNN fa una predizione di x incrementando una circonferenza centrata in x fino a quando questa ha bias significativo. Successivamente si sceglie la label corrispondente.

In alcuni casi questo non accade, quindi viene ritornato il valore ?. Viene definito il classificatore AKNN:

$$g_n: X \to \{-1, +1, ?\}$$

Verrà successivamente discussa una variante di tale algoritmo, con un intervallo di confidenza modificato:

$$\Delta(n, k, \delta) = c_1 \sqrt{\frac{d_0 \log n + \log(\frac{1}{\delta})}{k}} \quad (7)$$

Dove d_0 è la dimensione della famiglia delle circonferenze in (X, d) e la costante universale c_1 verrà dettagliata più avanti nel Lemma 7.

Comparando l'algoritmo descritto sotto al *kNN* standard, potrebbe sembrare che sia stato solo rimpiazzato il parametro k con un altro paramentro δ, ma non è così, infatti δ è il parametro di confidenza abituale della statistica e della teoria dell'apprendimento: fornisce un limite superiore alla probabilità di fallimento dell'algoritmo. Ad esempio, può essere impostato su 0,05. L'algoritmo fa infinite scelte di parametri - imposta k per ciascun punto di query - e richiede solo una singola probabilità di errore che gli consente di sapere con quale aggressività impostare i suoi intervalli di confidenza.

Di seguito viene riassunto e schematizzato il funzionamento dell'algoritmo AKNN.

Dati:

- *Un training set* $(x_1, y_1) \dots (x_n, y_n) \in X \times \{\pm 1\}$.
- *Un parametro di confidenza* $0 < \delta < 1$

Per fare una predizione di $x \in X$:

- *Per ogni intero k, sia* $B_k(x)$ *la più piccola circonferenza centrata in* x *che contenga esattamente* k *punti di training. Se molteplici punti hanno la stessa distanza da* x*, potrebbero esserci valori di* k *per cui* $B_k(x)$ *non sia definita, questo algoritmo li ignora.*
- *Si trova il più piccolo* $0 < k < 1$ *per il quale* $B_k(x)$ *ha un bias significativo: questo è*

  $\left|\eta_n\left(B_k(x)\right)\right| > \Delta(n,k,\delta)$*, dove* $\Delta(n,k,\delta) = c_1\sqrt{\frac{d_0 \log n + \log(\frac{1}{\delta})}{k}}$ (6)
- *Se tale circonferenza esiste, viene ritornato il label* $sign(\eta_n\left(B_k(x)\right))$
- *Altrimenti se tale circonferenza non esiste, viene ritornato* "? "*.*

3.4 Rates di convergenza e pointwise advantage

Gli autori forniscono ora i rates di convergenza a campione finito per la regola adattiva del *kNN*. Per semplicità vengono forniti rates di convergenza specifici per qualsiasi punto di query x e che dipendono da una nozione adeguata del "margine" della distribuzione P attorno a x.

Si considerano $p, \gamma > 0$ qualsiasi. Facendo riferimento alla (3), si definisce un punto $x \in X$ come (p,γ)-salient se la definizione seguente vale per $s = \pm 1$:

$$s\eta(x) > 0 \; and \; s\eta(B(x,r)) > 0 \; per \; ogni \; r \in \left[0, r_p(x)\right) \; and \; s\eta\left(B(x, r_p(x))\right) \geq \gamma$$

Questo significa che $g^*(x) = s$ e che i bias di tutte le circonferenze di raggio $\leq r_p(x)$ centrate in x hanno lo stesso segno di s, inoltre questi bias hanno valore assoluto

almeno γ. Un punto x può soddisfare questa definizione per una varietà di coppie (p, γ). L'$advantage$ di x risulta essere il più grande valore di $p\gamma^2$ tra tutte le coppie:

$$advantage(x) = \begin{cases} \sup\{p\gamma^2 : x \; is \; (p,\gamma) - \text{salient}\} & \eta(x) \neq 0 \\ 0 & \eta(x) = 0 \end{cases} \quad (8)$$

Viene poi dimostrato dagli autori che sotto certe condizioni, quasi tutte le x con $\eta(x) \neq 0$ hanno $advantage$ positivo.

3.5 Limiti del campione finito basati sull'advantage

Gli autori dichiarano ora due limiti di generalizzazione per il classificatore AKNN. Il primo vale in senso puntuale - limita la probabilità di errore in un punto specifico x - mentre il secondo è il tipo di limite di convergenza uniforme che è più standard nella teoria dell'apprendimento.

Il prossimo teorema mostra che per ogni punto x, se la dimensione del campione soddisfa $n \gtrapprox 1/adv(x)$ allora la label di x potrebbe essere $g^*(x)$ (ovvero il classificatore ottimo di Bayes). Viene provata la convergenza puntuale di $g(x)$ a $g^*(x)$ ad un rate che è sensibile alla geometria locale di x.

Teorema 1 (pointwise convergence rate). *Esiste una costante assoluta $C > 0$ per la quale vale la seguente affermazione. Sia $0 < \delta < 1$ il parametro di confidenza dell'algoritmo AKNN, e si supponga che l'algoritmo sia utilizzato per definire un classificatore g_n basato su n training points scelti i.i.d. da P. Per ogni punto $x \in supp(\mu)$ se*

$$n \geq \frac{C}{adv(x)} \max\left(\log \frac{1}{adv(x)}, log \frac{1}{\delta}\right)$$

Allora con probililità di almeno $1-\delta$ *abbiamo che* $g_n(x) = g^*(x)$.

Se si assume anche che la famiglia delle circonferenze nello spazio ha dimensione finita d_0 allora si può rafforzare questo teorema con alta probabilità simultaneamente per tutti gli $x \in supp(\mu)$. Ciò viene ottenuto mediante una versione modificata dell'algoritmo che usa (7) come intervallo di confidenza al posto di (6).

Teorema 2 (uniform convergence rate). *Si supponga che il set di circonferenze in* (X, d) *abbia dimensione finita* d_0 *e che l'algoritmo sopra descritto (3.3) usi come intervallo di confidenza (7) al posto di (6). Allora, con probabilità di almeno* $1-\delta$*, il classificatore risultante* $g_n(x)$ *soddisfa la seguente: per ogni ounto* $x \in supp(x)$*, se*

$$n \geq \frac{C}{adv(x)} \max\left(\log \frac{1}{adv(x)}, log\, \frac{1}{\delta}\right)$$

allora $g_n(x) = g^*(x)$.

Un punto chiave per dimostrare questi due teoremi è quello di identificare un subset di X che potrebbe essere classificato correttamente per un numero di training points n.

Ciò segue lo schema approssimativo di *[K. Chaudhuri and S. Dasgupta. Rates of convergence for nearest neighbor classification. In Advances in Neural Information Processing Systems, pages 3437–3445. 2014.]*, che ha dato i tassi di convergenza per *kNN,* ma ci sono due differenze notevoli. In primo luogo, si osserva che gli insiemi ottenuti in quel lavoro precedente (per *kNN*) sono, approssimativamente, sottoinsiemi di quelli che si ottengono per la nuova procedura AKNN. In secondo luogo, la prova dell'impostazione degli autori è notevolmente più snella.

3.6 Confronto con kNN

Si consideri un $a \geq 0$, sia X_a l'insieme di tutti i punti con $advantage > a$:

$$X_a = \{\mathrm{x} \in \mathrm{supp}(\mu)\colon \mathrm{adv}(\mathrm{x}) > \mathrm{a}\} \qquad (9)$$

In particolare X_0 considera tutti i punti con $advantage$ positivo.

Dal Teorema 1, punti in X_a possono essere classificati correttamente quando il numero dei training points è $\Omega(\frac{1}{a})$ dove $\Omega(*)$ ignora i termini logaritmici. In contrasto, il lavoro *[K. Chaudhuri and S. Dasgupta. Rates of convergence for nearest neighbor classification. In Advances in Neural Information Processing Systems, pages 3437–3445. 2014.]* ha mostrato che con n training points il classificatore *kNN* può classificare correttamente il seguente set di punti:

$$X'_{n,k} = \left\{x \in supp(\mu)\colon \eta(x) > 0, \eta\big(B(x,r)\big) \geq k^{-\frac{1}{2}}\, per\ ogni\ 0 < r < r_{\frac{k}{n}}(x)\right\}$$

$$\cup \{x \in supp(\mu)\colon \eta(x) < 0, \eta\big(B(x,r)\big) \leq -k^{-\frac{1}{2}}\, per\ ogni\ 0 < r < r_{\frac{k}{n}}(x)\}$$

Questi punti sono $\left(\frac{k}{n}, k^{-\frac{1}{2}}\right) - salient$ e hanno quindi $advantage$ di almeno $1/n$, infatti:

$$\bigcup_{1 \leq k \leq n} X'_{n,k}\ contenente\ X_{1/n}$$

In questo senso, AKNN riesce a performare circa uguale a scelte simultanee di k. Questa definizione non è precisa, in quanto sono presenti fattori logaritmici. Gli autori mostrano questa lacuna negli esperimenti da loro condotti.

3.7 Consistenza universale

In questa sezione viene studiata la convergenza di $R(g_n)$ al rischio Bayesiano R^* come numero dei punti n incrementati. Uno stimatore è descritto come *universalmente consistente* in uno spazio di misura metrico (X, d, μ) se possiede lo stesso comportamento limitato per tutte le funzioni di aspettativa condizionale η.

Un precedente lavoro *[K. Chaudhuri and S. Dasgupta. Rates of convergence for nearest neighbor classification. In Advances in Neural Information Processing Systems, pages 3437–3445. 2014.]*, ha stabilito la consistenza universale del *kNN* per $\frac{k}{n} \to 0$ $e \frac{k}{\log n} \to \infty$ in ogni spazio di misura metrico che soddisfi la condizione di differenziazione di Lebesgue, che consiste per ogni f misurabile limitata tale che $f: X \to Real$ e per quasi tutte le $x \in X$,

$$\lim_{r \downarrow 0} \frac{1}{\mu(B(x,r))} \int_{B(x,r)} f \, d\mu = f(x) \qquad (10)$$

Ciò risulta vero in ogni spazio dimensionale finito normalizzato.

Gli autori mostrano ora che questa stessa condizione implica la consistenza universale per l'*AKNN*. Per prima cosa, implica che quasi ogni punto possiede $advantage$ positivo.

Lemma 3. *Si supponga lo spazio di misura metrico* (X, d, μ) *che soddisfi la condizione (10). Quindi, per ogni aspettativa condizionale* η*, il set di punti*

$$\{x \in X: \eta(x) \neq 0, adv(x) = 0\}$$

ha zero μ*-measure.*

Dimostrazione. Sia $X' \subseteq X$ l'insieme di punti $x \in supp(\mu)$ per i quali la condizione (10) risulti vera per $f = \eta$, ovvero $\lim_{r \downarrow 0} \mu\big(B(x,r)\big) = \eta(x)$. Dato che $\mu\big(supp(\mu)\big) = 1$, segue che $\mu(X') = 1$.

Si prenda un qualsiasi $x \in X'$ con $\eta(x) \neq 0$; senza perdite di generalità, $\eta(x) > 0$. Dalla (10), esiste $r_0 > 0$ tale che

$$\eta\big(B(x,r)\big) \geq \frac{\eta(x)}{2} \; per \; ogni \; 0 \leq r \leq r_0$$

Quindi x risulta $(p,\varphi) - salient \; per \; p = \mu\big(B(x,r_0)\big) > 0 \; e \; \varphi = \eta(x)/2,$ e ha $advantage$ positivo.

La consistenza universale è una conseguenza di questo Lemma.

Teorema 4. *Si supponga lo spazio di misura metrico (X,d,μ) che soddisfi la condizione (10). Sia (δ_n) una sequenza in [0,1] con*

(1) $\sum_n \delta_n < \infty$

(2) $\lim_{n\to\infty} (\log(\frac{1}{\delta_n}))/n = 0$

Sia il classificatore $g_{n,\delta_n}: X \to \{-1,+1,?\}$ il risultato dell'applicazione dell'algoritmo AKNN con n punti scelti i.i.d. da P e con parametro di confidenza δ_n. Definendo $R_n = R(g_{n,\delta_n})$ il rischio di g_{n,δ_n}, si ha che $R_n \to R^*$ quasi sicuramente.

3.8 Convergenza uniforme di misure empiriche condizionali

Un punto chiave dell'analisi degli autori è la convergenza uniforme legata a stime empiriche della probabilità condizionale.

Sia P una distribuzione su X e siano $\mathcal{A},\mathcal{B}$ due collezioni di eventi. Si considerino n samples indipendenti da P, chiamati $x_1 \dots x_n$. Si vuole stimare $P(A|B)$ contemporaneamente per ogni $\mathcal{A} \in A, \mathcal{B} \in B$. La stima empirica risulta la seguente:

$$P_n(A|B) = \frac{\sum_i 1_{[x_i \in A \cap B]}}{\sum_i 1_{[x_i \in B]}}$$

Gli autori provano ora a dimostrare quando queste stime danno una buona approssimazione. Il caso $\mathcal{B} = X$ viene gestito dalla classica teoria VC. Si assume in questa sezione che $\mathcal{A}, \mathcal{B}$ abbiano dimensioni finite, e definiamo d_0 limite superiore per entrambe $\mathcal{A}, \mathcal{B}$.

Per dimostrare queste affermazioni si suppongano solo il casi in cui $\mathcal{A}, \mathcal{B}$ contengano solo un evento, ovvero $\mathcal{A} = \{A\},\ \mathcal{B} = \{B\}$ e sia inoltre $\#_n(B) = \sum_i 1_{[x_i \in B]}$.Un limite di Chernoff indica che condizionando sull'evento tal per cui $\#_n(B) > 0$, risulta la seguente equazione con probabilità di almeno $1 - \delta$

$$|P(A|B) - P_n(A|B)| \leq \sqrt{\frac{2\log(\frac{1}{\delta})}{\#_n(B)}} \qquad (11)$$

Questo limite dipende da $\#_n(B)$ e quindi risulta dipendente dai dati. Per derivarlo, si usa questa condizionata su $x_i \in B$, l'evento $x_i \in A$ ha probabilità $P(A|B)$, quindi una variabile random $'\#_n(B) * p_n(A|B)'$ ha distribuzione binomiale con parametri $\#_n(B)$ e $P(A|B)$.

L'idea è che più samples appartengono a B, più si è certi con la stima empirica. Inoltre si vuole provare la seguente affermazione, riguardante una versione uniforme della (11):

Con probabilità di almeno $1 - \delta$,

$$(\forall A \in \mathcal{A})(B \in \mathcal{B})\colon |P(A|B) - P_n(A|B)| \leq O(\sqrt{\frac{d_0 \log(\frac{1}{\delta})}{\#_n(B)}})$$

dove $\#_n(B) = \sum_i 1_{[x_i \in B]}$.

Purtroppo gli autori dimostrano che questa affermazione ideale risulta falsa. Come esempio si consideri lo spazio delle probabilità definite disegnando uniformemente $x \sim [n]$, e colorando $x\ con\ c_x \in \{\pm 1\}$ uniformemente. Per ogni i sia B_i l'evento da cui i è stato estratto, e sia A l'evento per cui il colore estratto risulta $+1$. Formalmente $B_i =$

$\{i\} \times \{\pm 1\}$ *e* $A = [n] \times \{\pm 1\}$. E' dimostrato che la dimensione di $\mathcal{B} = \{B_i : i \leq n\}$ e $\mathcal{A} = \{A\}$ risulta uguale al massimo 1. Questa affermazione fallisce in questo caso: si potrebbe verificare che se si estraggono n samples da questo spazio con probabilità costante ci saranno alcuni j tali che:

(i) j ottiene sempre lo stesso colore (ad esempio {+1}) e

(ii) j è campionato almeno $\Omega(\log n / \log\log n)$ volte. (come visto in *M. Raab and A. Steger. Balls into bins - a simple and tight analysis. In Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings, pages 159–170,1998*).

Quindi con probabilità costante si ottiene che

$$P_n(A|B_i) = 1 \; e \; P(A|B_i) = 1/2$$

E quindi la differenza tra gli errori risulta ovviamente $1 - \frac{1}{2} = 1/2$, che chiaramente non è limitato superiormente da $O(\sqrt{\log \frac{\log n}{\log n}})$.

Viene dimostrata la seguente variante, più debole:

Teorema 5 (restatement): *Sia P la distribuzione di probabilità su X, e siano $\mathcal{A}, \mathcal{B}$ due famiglie di subsets misurabili di X tali che $VC(\mathcal{A})$, $VC(\mathcal{B}) \leq d_0$. Sia $n \in \mathbb{N}$ e siano $x_1 \dots x_n$ n $i.i.d.$ da P. Quindi il seguente evento accade con probabilità almeno $1 - \delta$:*

$$(\forall A \in \mathcal{A})(B \in \mathcal{B}) : |P(A|B) - P_n(A|B)| \leq \sqrt{\frac{k_0}{\#_n(B)}}$$

Dove $k_0 = 1000(d_0 \log 8n + \log(\frac{4}{\delta}))$ *e* $\#_n(B) = \sum_{i=1}^{n} 1[x_i \in B]$.

Si definiscono per prima cosa i seguenti lemmi, utilizzati poi per la discussione della variante del Teorema 5:

Lemma 6. *Esiste una costante universale c_0 tale che sia $\mathcal{B}$ una qualsiasi classe di un subset misurabile di X di dimensione x_0. Si prenda un qualsiasi $0 < \delta < 1$. Quindi con*

probabilità di almeno $1 - \delta^2/2$ *sulla scelta di* $(x_1, y_1) \dots (x_n, y_n)$*, per ogni* $B \in \mathcal{B}$ *e per ogni intero* k*, si ha:*

$$\mu(B) \geq \frac{k}{n} + \frac{c_0}{n} \max(k, d_0, \log n/\delta) \implies \mu_n(B) \geq k/n$$

Ci si aspetterebbe che il bias empirico $\eta_n(S)$ *di un set* $S \subset X$*, come definito nella (5), sia vicino al vero bias* $\eta(S)$*, laddove* $\eta(S) > 0$*.*

Lemma 7. *Esiste una costante universale* c_0 *tale che sia* $\mathcal{C}$ *una qualsiasi classe di un subset misurabile di* X *di dimensione* x_0*. Si prenda un qualsiasi* $0 < \delta < 1$*. Quindi con probabilità di almeno* $1 - \delta^2/2$ *sulla scelta di* $(x_1, y_1) \dots (x_n, y_n)$*, per ogni* $C \in \mathcal{C}$ *si ha:*

$$|\eta_n(C) - \eta(C)| \leq \Delta(n, \#_n(C), \delta)$$

Dove $\#_n(C) = |\{i : x_i \in B\}|$ *è il numero di punti in* C *e*

$$\Delta(n, k, \delta) = c_1 \sqrt{\frac{d_0 \log n + \log(\frac{1}{\delta})}{k}} \qquad (12)$$

Il Lemma 7 è un caso speciale di limite di convergenza uniforme per probabilità condizionate.

Discussione variante del Teorema 5. Il Teorema 5 può essere combinato con il Lemma 6 per ottenere un limite sulla minima n per la quale $P_n(A|B)$ risulta una approssimazione non banale di $P(A|B)$. Sicuramente il Lemma 6 implica che se n è grande abbastanza in modo tale che $P(B) = \Omega(\frac{d_0 \log n}{n})$, allora la stima empirica $P_n(A|B)$ risulta una buona approssimazione. Nel contesto del classificatore *AKNN* questo significa che i bias empirici forniscono stime importanti dei veri bias per circonferenze di misura $\Omega(\frac{d_0}{n})$. Questo ricorda il learning rate in contesti realizzabili.

Dato che la dimensione di $\{B \cap A : i \in \mathcal{I}\}$ risulta al massimo d_0, allora

$$P_n(A|B) \approx \frac{P(A \cap B) \pm \sqrt{d_0/n}}{P(B) \pm \sqrt{d_0/n}}$$

Questo limite garantisce stime non banali solo quando $P(B)$ è circa $\sqrt{d_0/n}$. Questo è simile al learning rate in contesti non realizzabili.

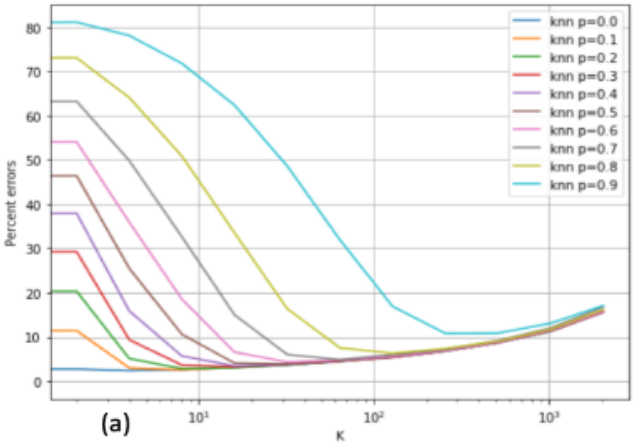
Un altro grande vantaggio del limite di convergenza uniforme del Teorema 5 è che è dipendente dai dati: se molti punti di un sample appartengono a $B \in \mathcal{B}$, ovvero se $\#_n(B)$ è grande, allora si ottengono migliori garanzie sull'approssimazione di $P(A|B)$ da $P_n(A|B)\ per\ ogni\ A \in \mathcal{A}$.

4. Esperimenti

4.1 Esperimenti degli autori

Gli autori hanno testato l'algoritmo AKNN su diversi dataset reali.

Il primo dataset che viene utilizzato è *MNIST*. Esso contiene circa 70000 immagini di numeri scritti a mano. Di questo numero, 60000 sono specifiche per il training set mentre le restanti 10000 per il test set. Questo set è un subset dell'originale NIST, con la differenza che le immagini sono state già preprocessate (ovvero hanno subito un resize ed i numeri sono stati acccentrati).



(a)

noise level	optimal k	error of optimal k	error of adaptive knn
0.0	4	2.39	3.39
0.1	8	2.59	3.61
0.2	8	2.86	3.79
0.3	16	3.27	4.26
0.4	16	3.46	4.53
0.5	32	3.91	5.12
0.6	32	4.35	6.15
0.7	64	4.87	7.71
0.8	128	6.32	9.50
0.9	256	10.84	12.22

(b)

In questa figura gli autori mostrano le prestazioni dell'algoritmo *AKNN* applicato a *MNIST*. In particolare si mostra l'effetto del rumore di classe sia sul classico *KNN* sia sulla sua versione adattiva. Per diversi valori e livelli di rumore di classe randomico p e per diversi valori di k. Ogni linea nella figura **(a)** rappresenta le prestazioni del *kNN* in funzione di k per un dato livello di rumore. Chiaramente più aumenta il rumore, più la scelta ottimale di k diventa un numero maggiore. Le prestazioni diminuiscono di molto per un valore piccolo di k.

La figura **(b)** mostra che *AKNN*, per un fissato valore di A, performa quasi ugualmente a *KNN* con la scelta ottimale di k.

Un più importante esperimento viene effettuato sul dataset *notMNIST*, il quale consiste nella raccolta di immagini di lettere di diversi font, alla *A* alla *J,* come mostrato in un esempio in figura sottostante.

Viene riportato di seguito il codice dell'algoritmo *AKNN* in Python 3.6, adattato al caso *notMNIST*. Sono necessari l'utilizzo di famose librerie esterne, quali:

- **Scikit-learn** (ex scikits.learn) è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy.
- **NumPy** è una libreria open source per il linguaggio di programmazione Python, che aggiunge supporto a grandi matrici e array multidimensionali insieme a una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati.

```
import numpy as np, sklearn, umap
import sklearn.metrics
from sklearn import preprocessing
```

```
def aknn(nbrs_arr, labels, thresholds, distinct_labels=['A', 'B', 'C',
'D', 'E', 'F', 'G', 'H', 'I', 'J']):
    """
    Apply AKNN rule for a query point, given its list of nearest
neighbors.

    Parameters
    ----------
    nbrs_arr: array di dimensione (n_neighbors)
        Indici dei `n_neighbors` nearest neighbors neldataset.
    labels: array di dimensione (n_samples)
        Labels del dataset.

    thresholds: array di dimensione (n_neighbors)
        Bias thresholds per divverenti dimensioni di neighborhood.
    Returns
    -------
    pred_label: string
        Predizione della label di AKNN.
    first_admissible_ndx: int
        n-1, dove AKNN sceglie la dimensione del neighborhood pari a
n.

    fracs_labels: array di dimensione (n_labels, n_neighbors)
        Frazione di ogni label in circonferenze di diversa dimensione
per ogni neighborhood.
    """
    query_nbrs = labels[nbrs_arr]
    mtr = np.stack([query_nbrs == i for i in distinct_labels])
    rngarr = np.arange(len(nbrs_arr)) + 1
```

```
    fracs_labels = np.cumsum(mtr, axis=1) / rngarr
    biases = fracs_labels - 1.0 / len(distinct_labels)
    numlabels_predicted = np.sum(biases > thresholds, axis=0)

admissible_ndces = np.where(numlabels_predicted > 0)[0]
    first_admissible_ndx = admissible_ndces[0] if
len(admissible_ndces) > 0 else len(nbrs_arr)
    pred_label = '?' if first_admissible_ndx == len(nbrs_arr) else
distinct_labels[np.argmax(biases[:,

first_admissible_ndx])]  # Break any ties between labels at stopping
radius, by taking the most biased label
    return (pred_label, first_admissible_ndx, fracs_labels)



def predict_nn_rule(nbr_list_sorted, labels, log_complexity=1.0,
                    distinct_labels=['A', 'B', 'C', 'D', 'E', 'F',
'G', 'H', 'I', 'J']):
    """
    Data una matrice di nearest neighbors ordinati per ogni punto,
ritorna la predizione della classe fatta da AKNN and e le dimensioni
adattive del neighborhood.

    Parameters
    ----------
    nbr_list_sorted: array di dimensione (n_samples, n_neighbors)
        Indici dei `n_neighbors` nearest neighbors nel dataset, per
ogni punto.
    labels: array di dimensione (n_samples)
        Labels del dataset.
```

```
    log_complexity: float
        Il parametro di confidenza "A".
    Returns
    -------
    pred_labels: array di dimensione (n_samples)
        Predizioni delle label di AKNN sul dataset.
    adaptive_ks: array di dimensione (n_samples)
        Dimensioni dei AKNN neighborhood sul dataset.
    """
    pred_labels = []
    adaptive_ks = []
    thresholds = log_complexity /
np.sqrt(np.arange(nbr_list_sorted.shape[1]) + 1)
    distinct_labels = np.unique(labels)
    for i in range(nbr_list_sorted.shape[0]):
        (pred_label, adaptive_k_ndx, _) = aknn(nbr_list_sorted[i, :],
labels, thresholds)
        pred_labels.append(pred_label)
        adaptive_ks.append(adaptive_k_ndx + 1)
    return np.array(pred_labels), np.array(adaptive_ks)



def calc_nbrs_exact(raw_data, k=1000):
    """
    Calcola la lista dei `k` esatti nearest neighbors Euclidei per
ogni punto.

    Parameters
    ----------
    raw_data: array di dimensione (n_samples, n_features)
        Dataset di input.
```

```
    Returns
    -------
    nbr_list_sorted: array di dimensionee (n_samples, n_neighbors)
        Indici dei `n_neighbors` nearest neighbors nel dataset, per
ogni punto.
    """

 a = sklearn.metrics.pairwise_distances(raw_data)
    nbr_list_sorted = np.argsort(a, axis=1)[:, 1:]
    return nbr_list_sorted[:, :k]


def knn_rule(nbr_list_sorted, labels, k=10):
    toret = []
    for i in range(nbr_list_sorted.shape[0]):
        uq = np.unique(labels[nbr_list_sorted[i, :k]],
return_counts=True)
        toret.append(uq[0][np.argmax(uq[1])])
    return np.array(toret)
```

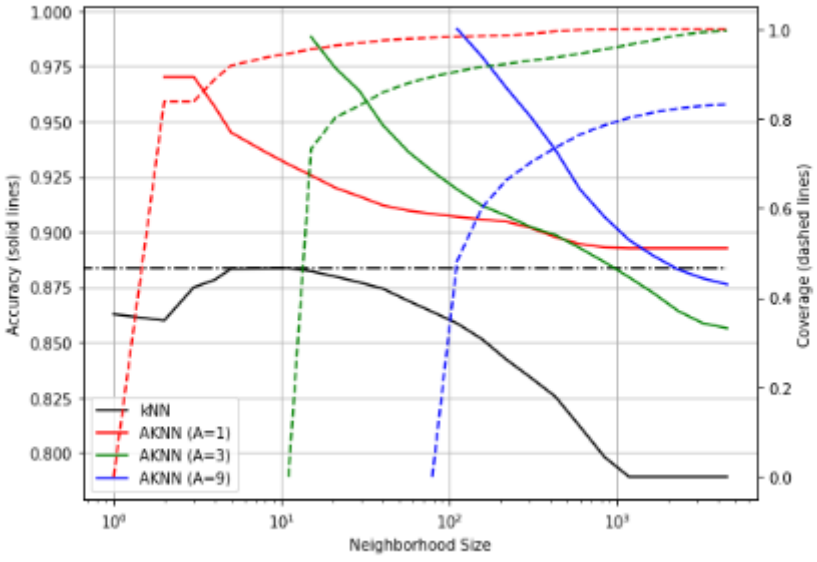
Di seguito viene proposto invece il codice per l'applicazione sul dataset *notMNIST.* Il dataset originale viene proposto in un file .mat di Matlab, viene adattato per Python seguendo le indicazioni del creatore del dataset (http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html).

```
import numpy as np, scipy as sp, time, scipy.io
import aknn_alg

if __name__ == '__main__':
```

```
    '''
    Importazione del dataset notMNIST e conversione da file .mat a .py
seguendo le istruzioni date dal creatore del dataset.
    '''
    print("Importo dataset...")
    notMNIST_small =
scipy.io.loadmat("notMNIST_small.mat")['images'].reshape(784, 18724)
    print("Dataset importato correttamente.")
    nmn = (notMNIST_small.T - 255.0 / 2) / 255.0   #resize
    labels =
scipy.io.loadmat("notMNIST_small.mat")['labels'].astype(int)
    labels_to_symbols = {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5:
'F', 6: 'G', 7: 'H', 8: 'I', 9: 'J'}
    labels = np.array([labels_to_symbols[x] for x in labels])

    # Calcola la lista dei k esatti nearest neighbors Euclidei per
ogni punto usando l'apposita funzione in aknn_alg.py.
    itime = time.time()
    nbrs_list = aknn_alg.calc_nbrs_exact(nmn, k=1000)
    print('Neighbor indices computed. Time:\t {}'.format(time.time() -
itime))

    itime = time.time()
    aknn_predictions = aknn_alg.predict_nn_rule(nbrs_list, labels)
    print('AKNN predictions made. Time:\t {}'.format(time.time() -
itime))

    #Confronto tra kNN e AkNN.
    kvals = [3, 5, 7, 8, 10, 30, 100]
    for i in range(len(kvals)):
        knn_predictions = aknn_alg.knn_rule(nbrs_list, labels,
```

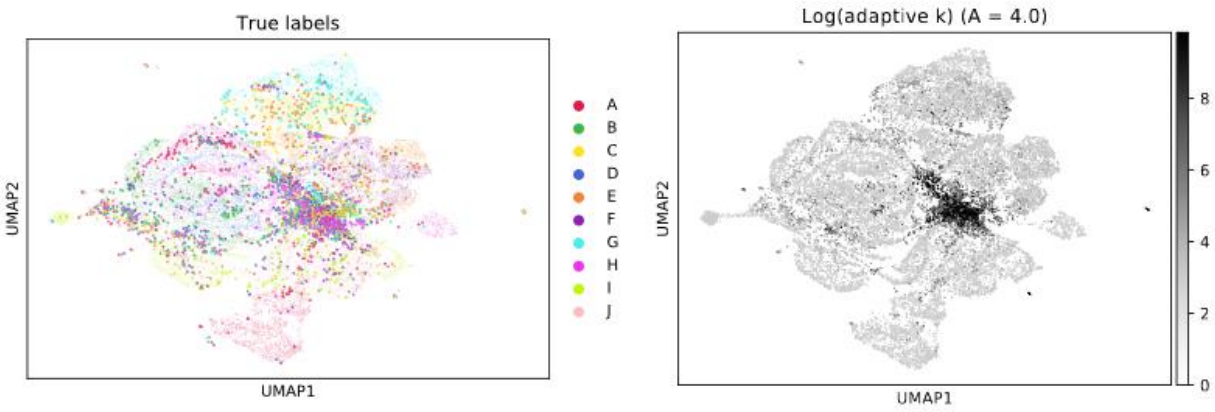
```
k=kvals[i])
        aknn_cov_ndces = aknn_predictions[1] <= kvals[i]
        aknn_cov = np.mean(aknn_cov_ndces)
        aknn_condacc = np.mean((aknn_predictions[0] ==
labels)[aknn_cov_ndces])
        print('{}-NN accuracy: \t\t{}'.format(kvals[i],
np.mean(knn_predictions == labels)))


print('AKNN accuracy (k <= {}): \t{} \t\t Coverage: \t{}\n'.format(
            kvals[i], aknn_condacc, aknn_cov))
    print('Overall AKNN accuracy:
{}'.format(np.mean(aknn_predictions[0] == labels)))



Neighbor indices computed. Time:   24.704349040985107
AKNN predictions made. Time:      3.232913017272949
3-NN accuracy:            0.8749198889126255
AKNN accuracy (k <= 3):   0.9701739850869926
 Coverage:        0.838015381328776
5-NN accuracy:            0.8833048493911557
AKNN accuracy (k <= 5):   0.9450261780104712
 Coverage:        0.9180730613116855
7-NN accuracy:            0.8836252937406537
AKNN accuracy (k <= 7):   0.9407625310030571
 Coverage:        0.9259239478743858
```

```
8-NN accuracy:            0.8834650715659047

AKNN accuracy (k <= 8):   0.9361872146118722

 Coverage:        0.935697500534074

10-NN accuracy:           0.8822901089510788

AKNN accuracy (k <= 10):  0.9321813031161473

 Coverage:        0.9426404614398632

30-NN accuracy:           0.8767891476180303

AKNN accuracy (k <= 30):  0.9158672400485169

 Coverage:        0.9687032685323649

100-NN accuracy:          0.858577227088229

AKNN accuracy (k <= 100):         0.9071918180829072

 Coverage:        0.9817346720786156

Overall AKNN accuracy: 0.8925977355265969
```

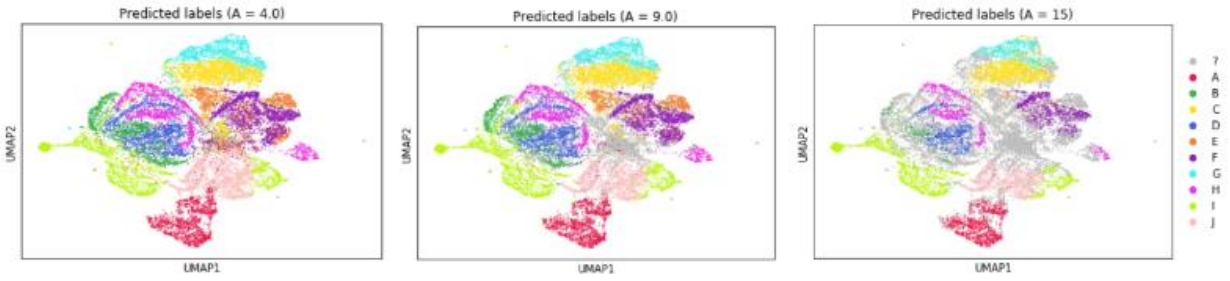


Nel grafico soprastante si mostrano le prestazioni di *AKNN* utilizzando diversi valori di A, in particolare $A = \{1,3,9\}$ in funzione della dimensione dei neighborhood.

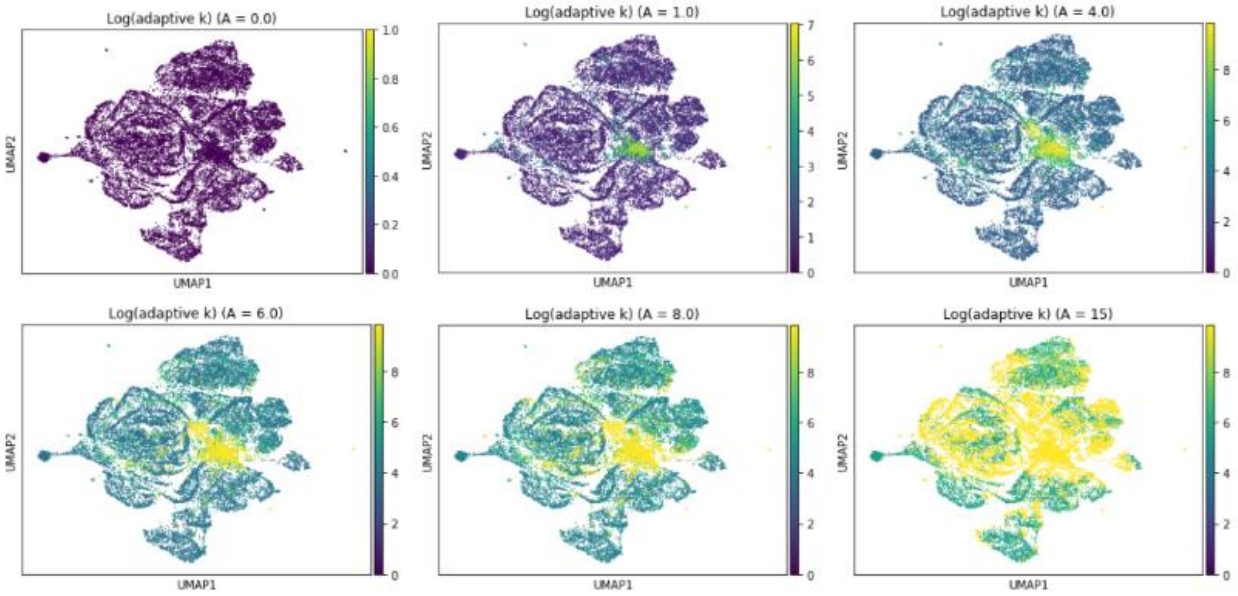
Per ogni livello di confidenza gli autori mostrano due grafici: uno per l'*accuracy* (linea solida) e uno di *coverage* (linea tratteggiata). Per ogni valore di k vengono plottate *accuracy* e *coverage* di *AKNN* usando un neighborhood di dimensione massima k. Aumentando A si nota un aumento dell'*accuracy* ma una diminuzione della *coverage*. Per alti valori di A e valori troppo bassi di k la *coverage* tende a zero. Sono plottate inoltre le prestazioni del *KNN* per confrontarle con quelle di *AKNN* in funzione di k. L'*accuracy* maggiore viene ottenuta per $k = 10$ (linea tratteggiata orizzontale) ed è sorpassata da *AKNN* con alta *coverage* (100% con $A = 1$).



Qui sopra gli autori propongono una visualizzazione delle prestazioni di *AKNN* sul dataset *notMNIST*. Sono evidenziate nella prima figura le labels corrette, con gli errori di predizione di *AKNN* per $A = 4$. Nella seconda figura viene mostrato il valore che assume k quando vengono fatte predizioni per ogni punto. Per diversi valori di A i grafici delle predizioni sono i seguenti:



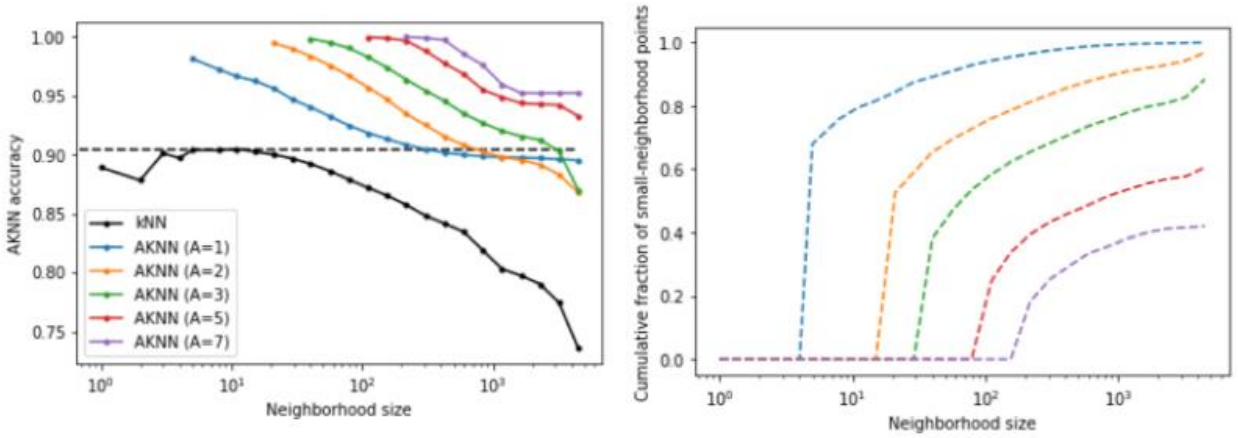
In scala logaritmica:



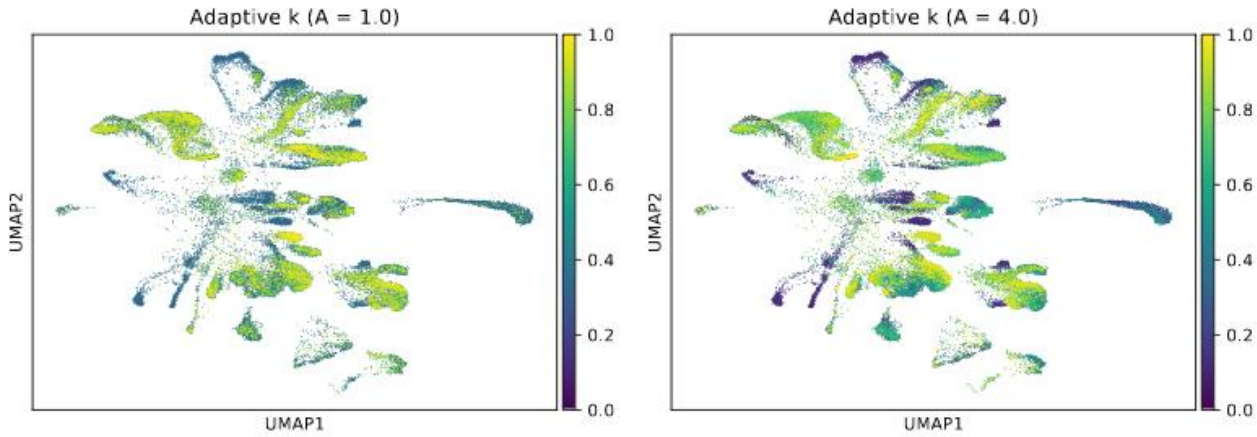
Dove in alto a sinistra abbiamo il grafico di *AKNN* con $A = 1$, in basso a destra con $A = 15$.

Gli autori utilizzano inoltre *AKNN* su un impegnativo compito di classificazione binaria di interesse continuo, che coinvolge i dati di espressione genica su una popolazione di singole cellule di diversi organi di topo raccolti dal consorzio Tabula Muris. Il dataset è costituito da 45291 cellule (traning set). Ogni cellula ha i suoi dati raccolti usando uno dei due seguenti approcci. Il compito è classificarli tra loro. I dati sono raccolti usando protocolli rappresentativi di due approcci attualmente dominanti per isolare e misurare singole cellule: un approccio basato su piastra che ordina le celle in micropozzetti e un

approccio basato su goccioline che manipola le cellule usando tecnologie microfluidiche. Ogni approccio ha una propria serie di pregiudizi tecnici, sui quali rimane ancora molto da capire. Identificare e caratterizzare questi pregiudizi per discriminare tali approcci è attualmente di grande interesse. Entrambi gli approcci misurano efficacemente le stesse cellule per i nostri scopi, quindi esiste un grande limite decisionale nel problema della classificazione binaria.



Questa figura mostra le prestazioni di *AKNN* su una singola cellula. Come per *notMNIST*, *AKNN* è accurato per punti di piccoli neighborhoods a *coverage* moderata, le prestazioni diminuiscono sostanzialmente per alti k, con il parametro A che controlla la frontiera.



4.2. Esperimento 10 Monkey Species

Similmente all'esperimento fatto con il semplice dataset notMNIST, ho provato ad utilizzare l'algoritmo implementato con un dataset più complesso, presente su *Kaggle* a questo link: https://www.kaggle.com/slothkong/10-monkey-species

In particolare il dataset è formato da 1098 immagini di scimmie (generalmente in qualità molto alta, in seguito discuterò di questo dettaglio) di 10 specie differenti, le quali formeranno poi le classi, o *labels*.

Di seguito riporto il codice, molto simile a quello utilizzato sul dataset notMNIST, chiaramente con metodologie differenti di importazione del dataset e di gestione delle immagini, fatta mediante la libreria *openCV*, utilizzando *Numpy* per la creazione degli array che verranno poi utilizzati dall'algoritmo.

```
import numpy as np, sklearn, umap
import sklearn.metrics
from sklearn import preprocessing

def aknn(nbrs_arr, labels, thresholds, distinct_labels=['n0', 'n1',
'n2', 'n3', 'n4', 'n5', 'n6', 'n7', 'n8', 'n9']):
    """
    Apply AKNN rule for a query point, given its list of nearest
neighbors.

    Parameters
    ----------
    nbrs_arr: array di dimensione (n_neighbors)
```

```
        Indici dei `n_neighbors` nearest neighbors neldataset.
    labels: array di dimensione (n_samples)
        Labels del dataset.

    thresholds: array di dimensione (n_neighbors)
        Bias thresholds per divverenti dimensioni di neighborhood.
    Returns
    -------
    pred_label: string
        Predizione della label di AKNN.
    first_admissible_ndx: int
        n-1, dove AKNN sceglie la dimensione del neighborhood pari a
n.

    fracs_labels: array di dimensione (n_labels, n_neighbors)
        Frazione di ogni label in circonferenze di diversa dimensione
per ogni neighborhood.
    """
    query_nbrs = labels[nbrs_arr]
    mtr = np.stack([query_nbrs == i for i in distinct_labels])
    rngarr = np.arange(len(nbrs_arr)) + 1
    fracs_labels = np.cumsum(mtr, axis=1) / rngarr
    biases = fracs_labels - 1.0 / len(distinct_labels)
    numlabels_predicted = np.sum(biases > thresholds, axis=0)
    admissible_ndces = np.where(numlabels_predicted > 0)[0]
    first_admissible_ndx = admissible_ndces[0] if
len(admissible_ndces) > 0 else len(nbrs_arr)
    pred_label = '?' if first_admissible_ndx == len(nbrs_arr) else
distinct_labels[np.argmax(biases[:,

first_admissible_ndx])]  # Break any ties between labels at stopping
```

```
radius, by taking the most biased label
    return (pred_label, first_admissible_ndx, fracs_labels)



def predict_nn_rule(nbr_list_sorted, labels, log_complexity=1.0,
                    distinct_labels=['n0', 'n1', 'n2', 'n3', 'n4',
'n5', 'n6', 'n7', 'n8', 'n9']):
    """
    Data una matrice di nearest neighbors ordinati per ogni punto,
ritorna la predizione della classe fatta da AKNN and e le dimensioni
adattive del neighborhood.

    Parameters
    ----------
    nbr_list_sorted: array di dimensione (n_samples, n_neighbors)
        Indici dei `n_neighbors` nearest neighbors nel dataset, per
ogni punto.
    labels: array di dimensione (n_samples)
        Labels del dataset.

    log_complexity: float
        Il parametro di confidenza "A".
    Returns
    -------
    pred_labels: array di dimensione (n_samples)
        Predizioni delle label di AKNN sul dataset.
    adaptive_ks: array di dimensione (n_samples)
        Dimensioni dei AKNN neighborhood sul dataset.
    """
    pred_labels = []
    adaptive_ks = []
```

```
    thresholds = log_complexity /
np.sqrt(np.arange(nbr_list_sorted.shape[1]) + 1)
    distinct_labels = np.unique(labels)
    for i in range(nbr_list_sorted.shape[0]):
        (pred_label, adaptive_k_ndx, _) = aknn(nbr_list_sorted[i, :],
labels, thresholds)
        pred_labels.append(pred_label)
        adaptive_ks.append(adaptive_k_ndx + 1)
    return np.array(pred_labels), np.array(adaptive_ks)



def calc_nbrs_exact(raw_data, k=1000):
    """
    Calcola la lista dei `k` esatti nearest neighbors Euclidei per
ogni punto.

    Parameters
    ----------
    raw_data: array di dimensione (n_samples, n_features)
        Dataset di input.
    Returns
    -------
    nbr_list_sorted: array di dimensionee (n_samples, n_neighbors)
        Indici dei `n_neighbors` nearest neighbors nel dataset, per
ogni punto.
    """
    a = sklearn.metrics.pairwise_distances(raw_data)
    nbr_list_sorted = np.argsort(a, axis=1)[:, 1:]
    return nbr_list_sorted[:, :k]
```

```
def knn_rule(nbr_list_sorted, labels, k=10):
    toret = []
    for i in range(nbr_list_sorted.shape[0]):
        uq = np.unique(labels[nbr_list_sorted[i, :k]],
return_counts=True)
        toret.append(uq[0][np.argmax(uq[1])])
    return np.array(toret)
```

```
import numpy as np, scipy as sp, time, scipy.io
from PIL import Image
from cv2 import *
import os
import aknnminecopy



def load_images_from_folder(folder, images):

    for filename in os.listdir(folder):
        img = cv2.imread(os.path.join(folder, filename), 0) #greyscale
        img=cv2.resize(img,(100,100), interpolation=INTER_AREA)

        '''#Per vedere la correttezza delle immagini ridotte
        cv2.imshow('img', img)
        cv2.waitKey()'''

        if img is not None:
            images.append(img)
    return images
```

```
def load_labels(folder, labels_db):
    for filename in os.listdir(folder):
        if filename is not None:
            label=filename[:2]
            labels_db.append(label)
    return labels_db
if __name__ == '__main__':
    '''
    Importazione dataset con immagini che sono numpy arrays
    '''
    print("Importo dataset...")
    images = []
    labels_db = []

    images=load_images_from_folder('./training/training/', images)

    labels_db=load_labels('./training/training/', labels_db)
    print("Dataset importato correttamente.")
    np_images=np.asarray(images).reshape(10000, 1098)

    np_labels=np.asarray(labels_db)
    print(type(np_images),type(np_labels))
    print(np_images.shape, np_labels.shape)
    print('Conversione effettuata')


    nmn = (np_images.T - 255.0 / 2) / 255.0   #resize
    #nmn = np_images.T

    # Calcola la lista dei k esatti nearest neighbors Euclidei per
ogni punto usando l'apposita funzione in aknn_alg.py.
```

```
    itime = time.time()
    print('Inizio calcolo...')
    nbrs_list = aknnminecopy.calc_nbrs_exact(nmn, k=1000)
    print('nbrs shape', nbrs_list.shape)
    print(nbrs_list)
    print('Neighbor indices computed. Time:\t {}'.format(time.time() -
itime))

    itime = time.time()
    aknn_predictions = aknnminecopy.predict_nn_rule(nbrs_list,
np_labels)
    print('AKNN predictions made. Time:\t {}'.format(time.time() -
itime))

    #Confronto tra kNN e AkNN.
    kvals = [3, 5, 7, 8, 10, 30, 100]
    for i in range(len(kvals)):
        knn_predictions = aknnminecopy.knn_rule(nbrs_list, np_labels,
k=kvals[i])
        aknn_cov_ndces = aknn_predictions[1] <= kvals[i]
        aknn_cov = np.mean(aknn_cov_ndces)
        aknn_condacc = np.mean((aknn_predictions[0] ==
np_labels)[aknn_cov_ndces])
        print('{}-NN accuracy: \t\t{}'.format(kvals[i],
np.mean(knn_predictions == np_labels)))
        print('AKNN accuracy (k <= {}): \t{} \n Coverage:
\t{}\n'.format(
            kvals[i], aknn_condacc, aknn_cov))
    print('Overall AKNN accuracy:
{}'.format(np.mean(aknn_predictions[0] == np_labels)))
```

```
AKNN predictions made. Time:      0.1926558017730713

3-NN accuracy:            0.3469945355191257

AKNN accuracy (k <= 3):   0.9375

 Coverage:        0.04371584699453552

5-NN accuracy:            0.5163934426229508

AKNN accuracy (k <= 5):   0.4935400516795866

 Coverage:        0.3524590163934426

7-NN accuracy:            0.2522768670309654

AKNN accuracy (k <= 7):   0.4935400516795866

 Coverage:        0.3524590163934426

8-NN accuracy:            0.33515482695810567

AKNN accuracy (k <= 8):   0.4935400516795866

 Coverage:        0.3524590163934426

10-NN accuracy:           0.3123861566484517

AKNN accuracy (k <= 10):  0.40380549682875266

 Coverage:        0.4307832422586521

30-NN accuracy:           0.43624772313296906

AKNN accuracy (k <= 30):  0.35344827586206895

 Coverage:        0.5282331511839709

100-NN accuracy:          0.22586520947176686
```

```
AKNN accuracy (k <= 100):        0.3032544378698225

 Coverage:        0.6156648451730419

Overall AKNN accuracy: 0.19034608378870674
```

Purtroppo l'algoritmo non funziona come mi sarei aspettato, o meglio, si nota bene come all'aumentare dall'*accuracy* per bassi valori di k la *coverage* sia pressochè nulla, al contrario aumentando k aumenta la *coverage*, ma con prestazioni decisamente peggiori rispetto a quelle viste per notMNIST. La ragione di ciò *potrebbe* essere un uso errato della libreria Numpy, anche se analogo a quanto fatto per notMNIST, in quanto quest'ultimo proviene da un file matLab e quindi gestito in modo differente per quanto riguarda l'importazione.

Una seconda motivazione riguardante le basse prestazioni potrebbe risiedere nella complessità delle immagini presenti in *10 Monkey Species*, il quale possiede solo 1000 immagini contro le quasi 19000 di notMNIST, ma ben più complesse a livello di dimensioni. Probabilmente il resize con successivo reshape dell'array hanno compromesso la qualità delle informazioni che portano poi a fare le predizioni.

5. Conclusioni

Gli autori hanno dimostrato teoricamente come *AKNN* sia comparabile alla migliore regola per *KNN*. In particolare hanno dimostrato che con un k fissato le prestazioni di *AKNN* tendono a quelle di $AKNN$, in particolar modo riguardo alla convergenza degli algoritmi. Gli autori dimostrano inoltre che con un rumore simulato indipendente dagli input del dataset MNIST, un piccolo valore di k risulta ottimale per dati senza rumore, ma performa molto male per dati pieni di rumore. Per questo motivo l'algoritmo implementato potrebbe essere buono per dataset semplici, senza rumore e con bassa varianza di valori dei pixel, nonostante $AKNN$ adatti automaticamente il livello del rumore localmente, come si nota per il dataset notMNIST.

Il particolare utilizzo del parametro A dimostra come l'algoritmo sia conservativo nel decidere se astenersi da una scelta piuttosto che scegliere un errore in predizione. $A \rightarrow 0$ è risultato essere l'approccio più aggressive, l'algoritmo infatti non si astiene mai, predicendo come *1NN.* Aumentando il valore di A, l'algoritmo tende ad astenersi dalle scelte più frequentemente in alcuni dei punti predetti, per i quali non esiste un neighborhood sufficientemente piccolo con un significativo *label bias*.

La scelta adattiva della dimensione del neighborhood riflette confidenza locale, in quanto il numero scelto da *AKNN* è una quantità locale che fornisce una misura punto per punto della confidenza associata alle predizioni delle label. Piccoli neighborhoods sono scelti quando una label viene misurata come statisticamente più vicina e possibile. Ciò risulta non essere vero quando sono necessari neighborhood più grandi. Negli

esperimenti si nota che le prestazioni su punti con un valore di neighbors alto calano drasticamente.

Questo algoritmo potrebbe essere utilizzato, come per il *KNN*, per la classificazione multiclasse in ambito machine learning. La variante degli autori però potrebbe essere inaccurata qualora si scelga di usare un dataset troppo complesso, oppure quando si deve operare con una grande mole di dati, in quanto i tempi sarebbero decisamente troppo lunghi, a meno di sporcare il dataset, procedendo con un reshape brutale che comprometterebbe però la qualità delle prestazioni, come visto nell'esperimento *10 Monkey Species*.

6. Bibliografia

- The Distance-Weighted k-Nearest-Neighbor Rule, *IEEE Transactions on Systems, Man, and Cybernetics (Volume: SMC-6 , Issue: 4 , April 1976)*
- *A k-nearest neighbor based algorithm for multi-label classification, 2005 IEEE International Conference on Granular Computing, Min Ling Zhang, Zhi-Hua Zhou*
- *Li Baoli, Lu Qin, Yu Shiwen. An adaptive k-nearest neighbor text categorization strategy. ACM Transactions on Asian Language Information Processing. 2004*
- *J Wang, P Neskovic, LN Cooper. Improving nearest neghbor rule with a simple adaptive distance measure. Pattern Recognition Letters, 2007 – Elsevier*
- *Stefanos Ougiaroglou, Alexandros Nanopoulos, Apostolos N. Papadopoulos, Yannis Manolopoulos, Tatjana Welzer-Druzovec. Adaptive k-Nearest-Neighbor Classification Using a Dynamic Number of Nearest Neighbors. ADBIS 2007: Advances in Databases and Information Systems pp 66-82. 2007*
- http://scholarpedia.org/article/K-nearest_neighbor
- http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
- https://scikit-learn.org/stable/
- https://www.kaggle.com/slothkong/10-monkey-species
- https://lorenzogovoni.com/knn/
- http://yaroslavvb.blogspot.com/2011/09/notmnist-dataset.html