به نام خدا

الگوريتم k نزديكترين همسايه

فاطمه دلدار- مليحه رضايي

الگوريتم k نزديكترين همسايه

الگوريتم k نزديكترين همسايه يک الگوريتم تعليم با سرپرستي است. در حالت كلي از اين الگوريتم به دو منظور استفاده مي‌شود: براي تخمين تابع چگالي توزيع داده‌هاي تعليم و براي طبقه‌بندي داده‌هاي تست بر اساس الگوهاي تعليم.

تخمين چگالي توزيع داده‌ها با استفاده از الگوريتم K_n نزديک‌ترين همسايه

براي تخمين p(x) از روي n نمونه‌ي تعليم توسط الگوريتم k نزديک‌ترين همسايه مي‌توانيم يک سلول به مركزيت x ايجاد كرده و اجازه دهيم اين شعاع اين سلول تا حدي گسترش پيدا كند كه k_n نمونه‌ي تعليم را در بر گيرد. اين نمونه‌ها k_n نزديكترين همسايه‌هاي x هستند.

- در حالت كلي k را به صورت k_n در نظر مي‌گيريم كه k_n تابعي تعريف شده از n است.

اگر چگالي نقاط تعليم اطراف x زياد باشد سلول كوچک مي‌شود و بنابراين نتيجه‌ي به دست آمده نتيجه‌ي بهتري است و در صورتي كه چگالي نقاط تعليم اطراف x كم باشد سلول بزرگ مي‌شود. در حالت كلي چگالي توزيع به ازاي هر نقطه‌ي x توسط رابطه‌ي زير محاسبه مي‌شود:

$$p_n(\mathbf{x}) = \frac{k_n / n}{V_n}$$

اگر با رشد n، k_n نيز افزايش پيدا كند به طوري كه با رفتن n به سمت بي-نهايت k_n نيز به بي‌نهايت ميل كند، آنگاه مي‌توان مطمئن بود كه k_n/n يک تخمين خوب از اين احتمال است كه يک نقطه در يک سلول به حجم V_n قرار بگيرد. بنابراين دو شرط زير شروط لازم و كافي براي اين است كه $p_n(x)$ در نهايت به p(x) همگرا شود:

$$\lim_{n\to\infty} k_n = \infty, \quad \lim_{n\to\infty} k_n / n = 0$$

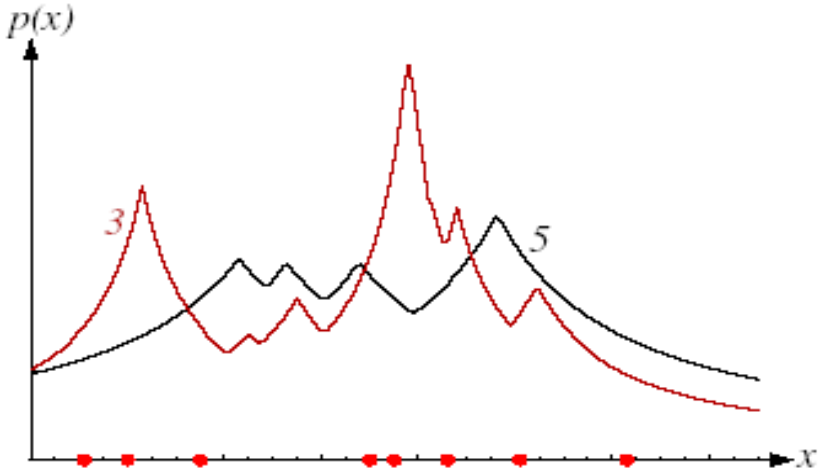
به عنوان مثال اگر k_n را $\sqrt{n}$ در نظر بگيريم داريم:

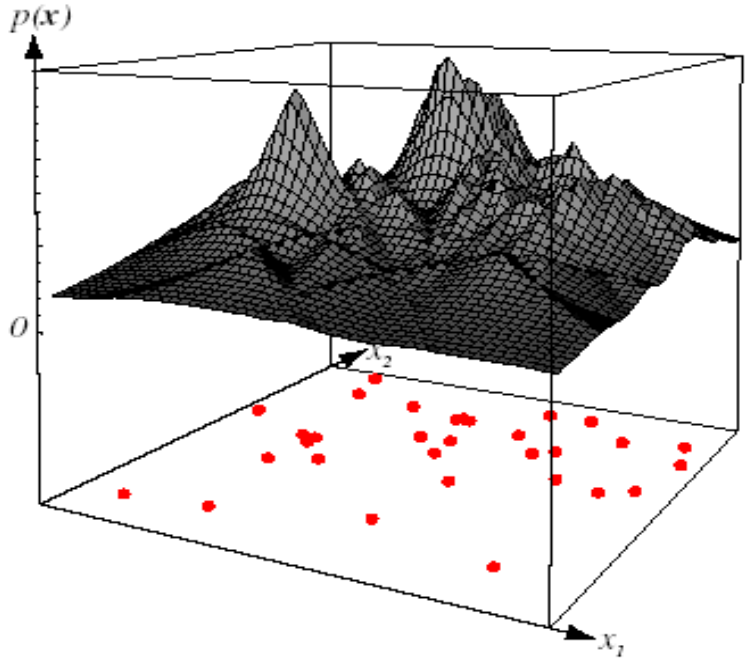
$$k_n = \sqrt{n}$$

$$V_n \approx \frac{1}{\sqrt{n}\,p(x)}, \quad V_n \approx \frac{V_1}{\sqrt{n}} \to 0 \text{ as } n \to \infty$$

و بنابراين شروط فوق براي k_n داده شده برقرار بوده و با افزايش n به سمت بي‌نهايت $p_n(x)$ به p(x) ميل مي‌كند. شكل‌هاي زير تخميني از تابع چگالي

توزيع نمونه‌هاي داده شده را به ترتيب براي نمونه‌هاي يک‌بعدي و دوبعدي نشان مي‌دهند.





تخمين احتمال‌هاي posteriori

تکنيک بيان شده در قسمت قبل مي‌تواند براي تخمين احتمال‌هاي posteriori، P(wi|x) از يک مجموعه از n نمونه‌ي برچسب‌گذاري شده استفاده شود. فرض مي‌کنيم که يک سلول به حجم V حول x رسم کرده‌ايم که k نمونه‌ي

ورودي را در بر گرفته است و ki تا از اين نمونه‌ها در كلاس wi قرار دارند. بنابراين يک تخمين براي احتمال p(x,wi) به صورت

$$p_n(\mathbf{x},\omega_i)=\frac{k_i/n}{V}$$

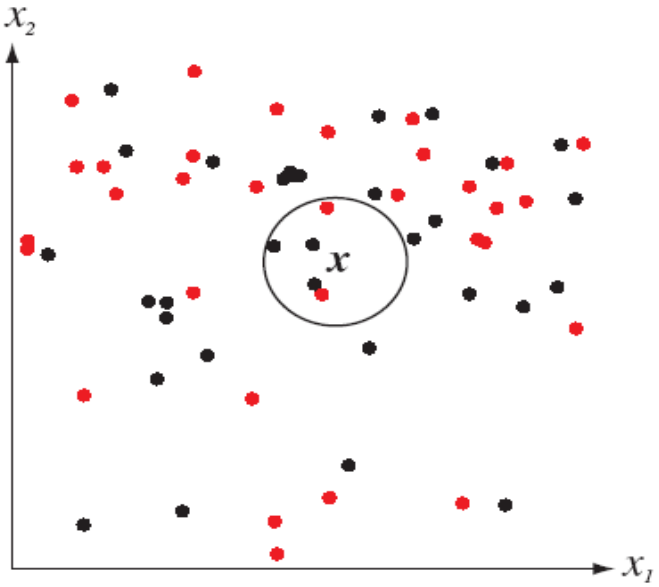
است و بنابراين داريم:

$$p_n(\omega_i\mid\mathbf{x})=\frac{p_n(\mathbf{x},\omega_i)}{\sum_{j=1}^{c}p_n(\mathbf{x},\omega_j)}=\frac{\frac{k_i/n}{V}}{\sum_{j=1}^{c}\frac{k_j/n}{V}}=\frac{k_i}{k}$$

يعني براي اينكه نرخ خطا حداقل باشد، نمونه‌ي تست را در دسته‌اي قرار مي‌دهيم كه بيشترين تكرار را در سلول داشته باشد. اگر تعداد نمونه‌ها به اندازه‌ي كافي زياد باشد و سلول به اندازه‌ي كافي كوچک باشد، اين انتخاب از لحاظ كارايي بهترين انتخاب است.

قانون نزديک‌ترين همسايه

در تعريف قانون نزديک‌ترين همسايه k را مساوي يک در نظر گرفته شده است. فرض مي‌كنيم $D^n=\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}$ مجموعه‌اي از n الگوي ورودي باشد و $\mathbf{x}'\in D^n$ نزديک‌ترين الگوي ورودي به نقطه‌ي تست x باشد. قانون نزديک‌ترين همسايه براي طبقه‌بندي x آن را در كلاسي مشابه با كلاس x′ قرار مي‌دهد.
قانون نزديک‌ترين همسايه يک روال sub-optimal است يعني نرخ خطاي آن معمولاً بيشتر از حداقل نرخ خطاي ممكن يعني نرخ خطاي الگوريتم بيز است. ولي ثابت مي‌شود كه در صورت استفاده از تعداد نامحدودي از الگوهاي ورودي نرخ خطا در بدترين حالت بيشتر از دو برابر نرخ خطاي الگوريتم بيز نخواهد شد.
قانون k نزديک‌ترين همسايه گسترشي از قانون نزديک‌ترين همسايه است و همان‌طور كه واضح است اين قانون x را در دسته‌اي طبقه‌بندي مي‌كند كه بيشترين تكرار را در بين k نزديک‌ترين همسايه‌ي x دارد. به عنوان مثال در شكل زير x توسط اين قانون در كلاس سياه قرار مي‌گيرد.



پيچيدگي محاسباتي در الگوريتم نزديک‌ترين همسايه

براي محاسبه‌ي پيچيدگي الگوريتم k نزديک‌ترين همسايه فرض مي‌کنيم که n نمونه‌ي تعليم d بعدي داريم و مي‌خواهيم نزديک‌ترين آنها به نقطه‌ي تست x (k=1) را به دست آوريم. در ساده‌ترين روش براي هر نقطه‌ي تعليم فاصله‌ي آن تا نقطه‌ي تست را محاسبه کرده و نزديک‌ترين آنها به نقطه تست را برمي‌گردانيم. محاسبه‌ي فاصله‌ي اقليدس براي هر نقطه در O(d) انجام مي‌شود و چون n نقطه‌ي تعليم ذخيره شده داريم اين محاسبه O(nd) زمان مي‌برد و در نهايت انتخاب کوتاهترين فاصله نيز در زمان O(n) انجام مي‌شود. بنابراين پيچيدگي زماني الگوريتم در حالت کلي O(nd) مي‌باشد.

سه تکنيک الگوريتمي براي کاهش پيچيدگي محاسباتي الگوريتم نزديک‌ترين همسايه وجود دارد:

- محاسبه‌ي فاصله‌هاي جزئي
- Prestructuring
- ويرايش الگوهاي ذخيره شده

محاسبه‌ي فاصله‌هاي جزئي: در اين روش فاصله‌ي بين دو نقطه را ابتدا به صورت کامل محاسبه نمي‌کنيم بلکه آن را به صورت تدريجي و با افزودن ابعاد محاسبه مي‌کنيم. اگر فضا را d بعدي در نظر بگيريم، فاصله‌ي جزئي بر مبناي r بعد انتخاب شده از اين d بعد به صورت زير محاسبه مي‌شود:

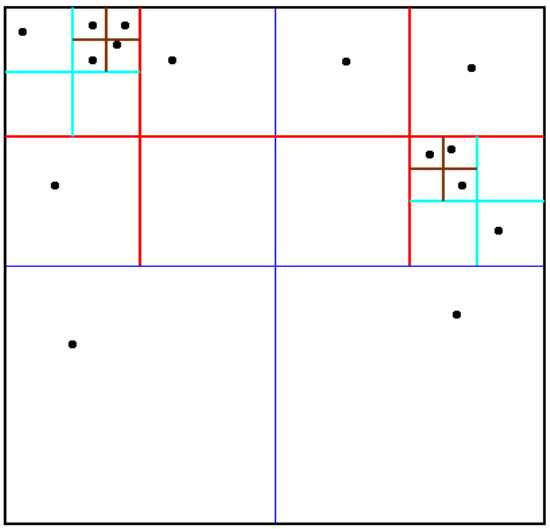
$$D_r(\mathbf{a},\mathbf{b}) = \left(\sum_{k=1}^{r} (a_k - b_k)^2 \right)^{1/2}, \quad r < d$$

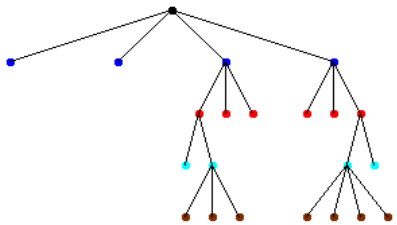
مي‌توانيم محاسبه‌ي فاصله‌ي جزئي را تا زماني ادامه دهيم که بزرگتر از فاصله‌ي فعلي نقطه‌ي تست از نزديک‌ترين الگوي فعلي شود. در اين حالت

مطمئن مي‌شويم كه اين الگو نزديكترين الگو به نقطه‌ي تست مورد نظر نيست و محاسبه‌ي فاصله را متوقف مي‌كنيم.

Prestructuring: در اين روش بر روي الگوهاي تعليم مورد جستجو شكلي از درخت جستجو را ايجاد مي‌كنيم. در حين فرآيند طبقه‌بندي كردن فقط فاصله-ي نقطه‌ي تست را تا يک يا تعداد كمي از الگوهاي ذخيره شده در درخت محاسبه مي‌كنيم. به اين معني كه يک الگو كه نزديک‌تر از بقيه‌ي الگوها به نقطه‌ي تست به نظر مي‌رسد را انتخاب كرده و به صورت بازگشتي فقط الگوهاي متصل به آن را در درخت جستجو مي‌كنيم. اگر درخت به درستي ساخته شده باشد تعداد كل الگوهاي موردنياز براي جستجو كاهش پيدا مي‌كند.

يكي از اين ساختارهاي درختي QuadTree است. در اين درخت هر نود مياني چهار فرزند دارد. اين درخت اغلب به اين منظور استفاده مي‌شود كه فضاهاي دوبعدي را به صورت بازگشتي به چهار ناحيه تقسيم كند. اين نواحي مي‌توانند مربع، مستطيل يا هر شكل دلخواه ديگري داشته باشد. شكل زير ساخته شدن يک QuadTree را نشان مي‌دهد.





اين روش به هيچ عنوان تضمين نمي‌كند كه نزديكترين الگو را پيدا كند. در جستجوي الگوهاي تعليم از طريق اين درخت در هر مرحله ¾ فضاي جستجو حذف مي‌شود و فقط ¼ اي باقي مي‌ماند كه فاصله‌ي نقطه‌ي تست تا مركز آن ناحيه كمتر از سه ناحيه‌ي ديگر است. بنابراين ممكن است نزديكترن الگو به نقطه‌ي تست در يكي از نواحي ديگر قرار داشته باشد (اين مشكل بيشتر در مواقعي پيش مي‌آيد كه نقطه‌ي تست نزديک به مرز ناحيه قرار دارد) و بنابراين با حذف آن ناحيه نزديک‌ترين الگو به نقطه‌ي تست را نيز از دست

بدهيم. با وجود اينكه اين روش روش تضمين‌شده‌اي نيست ولي براي كاهش فضاي جستجو روش مناسبي است.

ويرايش الگوهاي ذخيره‌شده: روش سوم براي كاهش پيچيدگي جستجوي نزديكترين همسايه حذف الگوهاي غيرمفيد در حين تعليم است. يک روش براي كاهش پيچيدگي فضايي O(n) اين است كه الگوهايي كه همه‌ي نقاط تعليم اطراف آنها در كلاسي مشابه با خود الگو قرار دارند را حذف كنيم. با اين كار مرزهاي تعليم تغيير نمي‌كند در حالي كه فضاي جستجو كاهش پيدا مي‌كند. يک الگوريتم ساده براي اين كار به صورت زير است:

Algorithm. (Nearest-Neighbor Editing)

1. **begin** **initialize** $j \leftarrow 0$, $D \leftarrow$ data set, $n \leftarrow$ # prototypes
2. construct the full Voronoi diagram of D
3. **do** $j \leftarrow j + 1$; for each prototype **x**′j
4. find the Voronoi neighbors of **x**′j
5. if any neighbor is not from the same class as , **then** mark **x**′j
6. **until** $j = n$
7. discard all points that are not marked
8. construct the Voronoi diagram of the remaining (marked) prototypes
9. **end**

از آنجايي كه الگوهايي كه در مرز تصميم قرار دارند حداقل يكي از همسايه-هايشان از دسته‌اي متفاوت هستند، بنابراين با اجراي الگوريتم بالا حذف نمي‌شوند و مرز تصميم تغييري پيدا نمي‌كند. اين الگوريتم تضمين نمي‌دهد كه حداقل مجموعه الگوهاي تعليم مفيد باقي بمانند. يكي از مشكلات چنين سيستم نزديک‌ترين همسايه‌اي اين است كه نمي‌توان بعداً الگوي تعليمي به اين الگوها اضافه كرد چون براي اجراي اين الگوريتم نياز است كه همه‌ي الگوهاي تعليم وجود داشته باشند.
مي‌توان اين سه روش كاهش پيچيدگي را با هم ادغام كرد، يعني ابتدا الگوها را ويرايش كرد، بعد بر روي الگوهاي باقيمانده يک درخت جستجو ايجاد كرد و در نهايت در هنگام انجام عمل طبقه‌بندي فاصله‌هاي جزئي را محاسبه كرد.

پياده‌سازي الگوريتم k نزديكترين همسايه

در پياده‌سازي الگوريتم k نزديک‌ترين همسايه در نرم‌افزار مطلب كلاسهاي زير به كار رفته است:

کلاس CalDistance.m

این کلاس فاصله‌ي بين همه‌ي نقاط تعليم از همه‌ي نقاط تست را بر اساس يکي از معيارهاي فاصله محاسبه کرده و ماتريس فاصله‌ها را برمي‌گرداند. ورودي اين کلاس train(نقاط تعليم)، test(نقاط تست) و distance(معيار فاصله-ي موردنظر) است که اين معيار فاصله مي‌تواند يکي از موارد زير باشد:

- euclidean: فاصله‌ي اقليدس
- cityblock: مجموع قدرمطلق فاصله‌ها
- cosine: يک منهاي کسينوس زاويه‌ي محصور بين دو نقطه (نقاط را به عنوان بردار در نظر مي‌گيريم)
- correlation: يک منهاي کورروليشن بين دو نقطه
- hamming: اين فاصله فقط براي داده‌هاي باينري مناسب است و درصدي از بيت‌ها را که با هم متفاوت هستند نشان مي‌دهد.

کلاس KNNClassification.m

اين کلاس کلاس اصلي برنامه مي‌باشد و k نزديکترين همسايه به هر يک از نقاط تست را محاسبه کرده و بر اساس آنها مشخص مي‌کند که هر يک از نقاط تست در چه دسته يا کلاسي قرار مي‌گيرند.

ورودي‌هاي اين کلاس train(نقاط تعليم)، test(نقاط تست)، group(ماتريسي ستوني که تعداد سطرهاي آن مساوي با تعداد سطرهاي train است و مشخص مي‌کند که هر يک از الگوهاي تعليم در چه کلاسي قرار مي‌گيرند)، k که تعداد نزديکترين همسايه مورد نظر براي الگوريتم KNN را مشخص مي-کند، dist که معيار فاصله‌ي مورد نظر براي محاسبه‌ي فاصله‌ي بين نقاط تعليم و نقاط تست را بيان مي‌کند و rule که مي‌تواند يکي از موارد زير باشد:

- consensus: وقتي اين گزينه به عنوان rule انتخاب شود به اين معني است فقط در صورتي که همه‌ي k نزديکترين همسايه‌ي يک نقطه‌ي تست به يک کلاس مشابه تعلق داشته باشند آن نقطه‌ي تست نيز در همان کلاس طبقه‌بندي مي‌شود ولي اگر حتي يکي از نقاط k نزديکترين همسايه‌ي يک نقطه‌ي تست در کلاسي متفاوت با بقيه قرار داشته باشد، آن نقطه‌ي تست قابل طبقه‌بندي با اين الگوريتم نيست. در ساير موارد به غير از مواقعي که اين گزينه به عنوان rule قرار مي‌گيرد از قانون اکثريت استفاده مي‌شود يعني يک نقطه‌ي تست در کلاسي قرار مي‌گيرد که تعداد نقاطي از k نزديکترين همسايه‌ي آن که در آن کلاس قرار دارند نسبت به ساير کلاسها بيشتر باشد. در مواقعي که گره ايجاد مي‌شود يعني بيشتر نقاط در يک کلاس قرار نمي‌گيرند بلکه نقاط قرار گرفته در بيش از يک کلاس با هم برابر هستند از يکي از سه قانون زير براي شکستن اين گره استفاده مي‌شود:

- nearest: يعني از بين گروه‌هاي به هم گره خورده كلاس نزديكترين همسايه به عنوان كلاس خروجي انتخاب مي‌شود.
- farthest: يعني از بين گروه‌هاي به هم گره خورده كلاس دورترين همسايه به عنوان كلاس خروجي انتخاب مي‌شود.
- random: يعني براي شكستن گره به صورت تصادفي يكي از همسايه-هايي كه به هم گره خورده‌اند انتخاب شده و كلاس آن همسايه به عنوان كلاس خروجي برگردانده مي‌شود.

در كلاس KNNClassification، ابتدا توسط تابع CalDistance فاصله‌ي بين نقاط تعليم و تست محاسبه شده و در ماتريس d قرار مي‌گيرد. سپس اين ماتريس را مرتب مي‌كنيم يعني براي هر نقطه‌ي تست نقاط تعليم را به صورت صعودي بر حسب فاصله از آن نقطه‌ي تست در سطر مربوط به آن قرار داده و سپس k نقطه‌ي تعليم اول هر يک از اين سطرها را به عنوان k نزديكترين همسايه‌ي هر يک از نقاط تست جدا مي‌كنيم. پس از آن در يک حلقه به ازاي هر يک از نقاط تست محاسبه مي‌كنيم كه هر يک از k نزديكترين همسايه‌ي آن در چه كلاسي قرار دارند و كلاسي كه ماكزيمم تعداد همسايه‌ها در آن قرار دارند را به عنوان كلاس خروجي انتخاب مي-كنيم. در صورت به وجود آمدن گره نيز با توجه به نوع rule گره را باز مي‌كنيم.

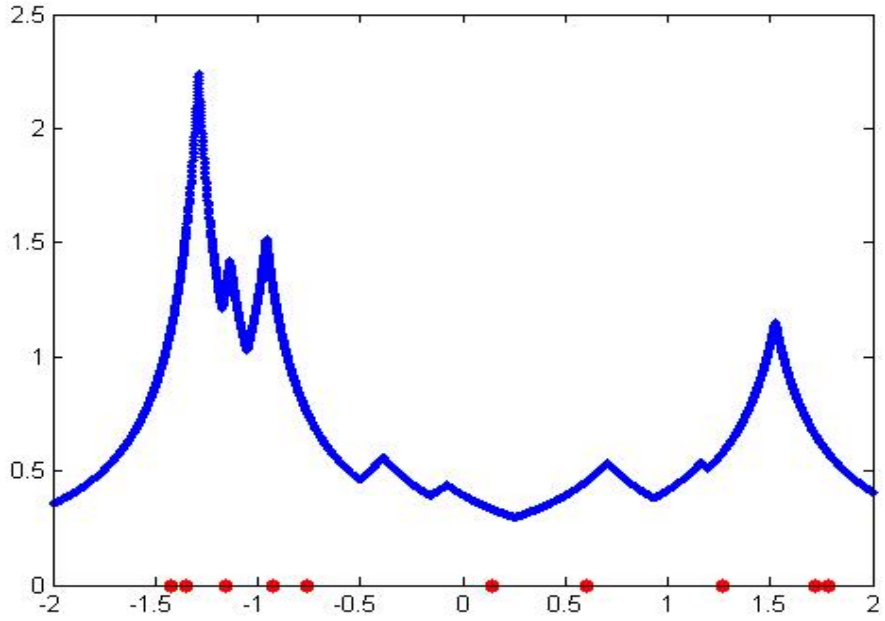
كلاس classification.m

اين كلاس كلاس اجرايي برنامه است كه در آن الگوهاي تعليم و گروه‌هاي آنها، الگوهاي تست، مقدار k، معيار فاصله و rule را انتخاب كرده و با توجه به آنها كلاس KNNClassification را فراخواني مي‌كنيم.

كلاس KNN1.m

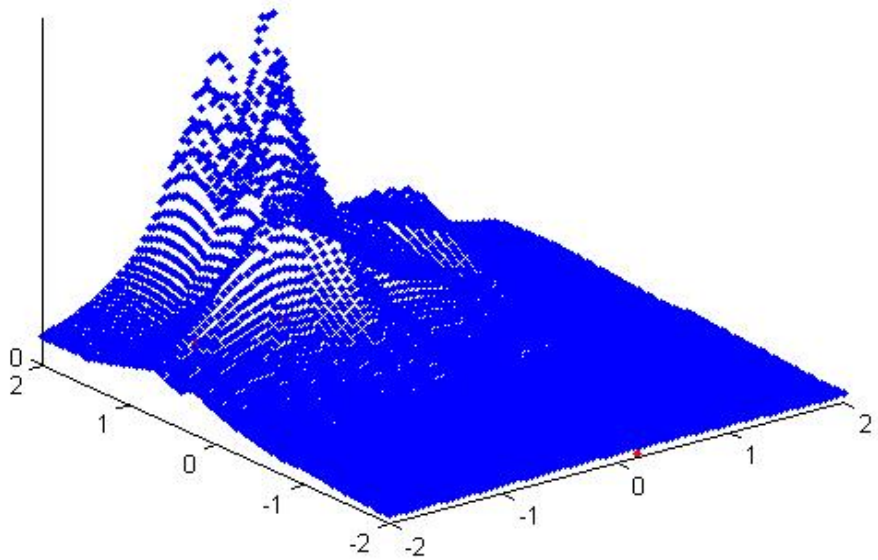
اين كلاس براي تخمين تابع چگالي توزيع نقاط تعليم با توجه به الگوريتم k نزديكترين همسايه استفاده مي‌شود. به اين منظور نقاط تست را از ابتدا تا انتهاي بازه‌اي كه نقاط تعليم در آن قرار دارند با فاصله‌ي كم از هم در نظر گرفته و براي هر كدام از آنها تابع CalDistance را فراخواني مي‌كنيم تا فاصله‌ي آن از تمام نقاط تعليم را به دست آوريم. سپس اين فاصله‌ها را مرتب مي-كنيم. از آنجايي كه در استفاده از الگوريتم k نزديک‌ترين همسايه براي تخمين چگالي توزيع الگوهاي ورودي در نقطه‌ي x به مركزيت آن نقطه سلولي ايجاد مي‌كنيم و اجازه مي‌دهيم شعاع اين سلول تا حدي گسترش پيدا كند كه k نزديک‌ترين همسايه‌ي نقطه‌ي x را در بر گيرد، بنابراين در ليست مرتب شده kامين عدد را به عنوان شعاع در نظر مي‌گيريم و در نهايت با استفاده از رابطه‌ي p=k/n.v مقدار تابع چگالي در نقطه‌ي مورد نظر را به دست مي‌آوريم. در كلاس KNN1 داده‌ها به صورت يک بعدي در نظر گرفته شده‌اند و بنابراين به جاي v طول خط يا همان k را در نظر مي‌گيريم. در شكل زير نمونه‌اي از

خروجي اين كلاس نشان داده شده است. (الگوهاي ورودي با نقاط قرمز مشخص شده‌اند)



كلاس KNN2.m

اين كلاس كاملاً مشابه كلاس KNN1 است با اين تفاوت كه در اين كلاس داده‌ها به صورت دو بعدي در نظر گرفته شده‌اند و به همين دليل به جاي v مساحت دايره‌ي به مركزيت x و شعاع k را محاسبه مي‌كنيم. در شكل زير نمونه‌اي از خروجي اين كلاس آمده است.


0
2
1
0
-1
-2
-2
-1
0
1
2