دانشگاه آزاد اسلامی واحد تبریز

نام درس: یادگیری ماشین

بخش: یادگیری مبتنی بر نمونه

نام استاد: دکتر مسعود کارگر

مقدمه

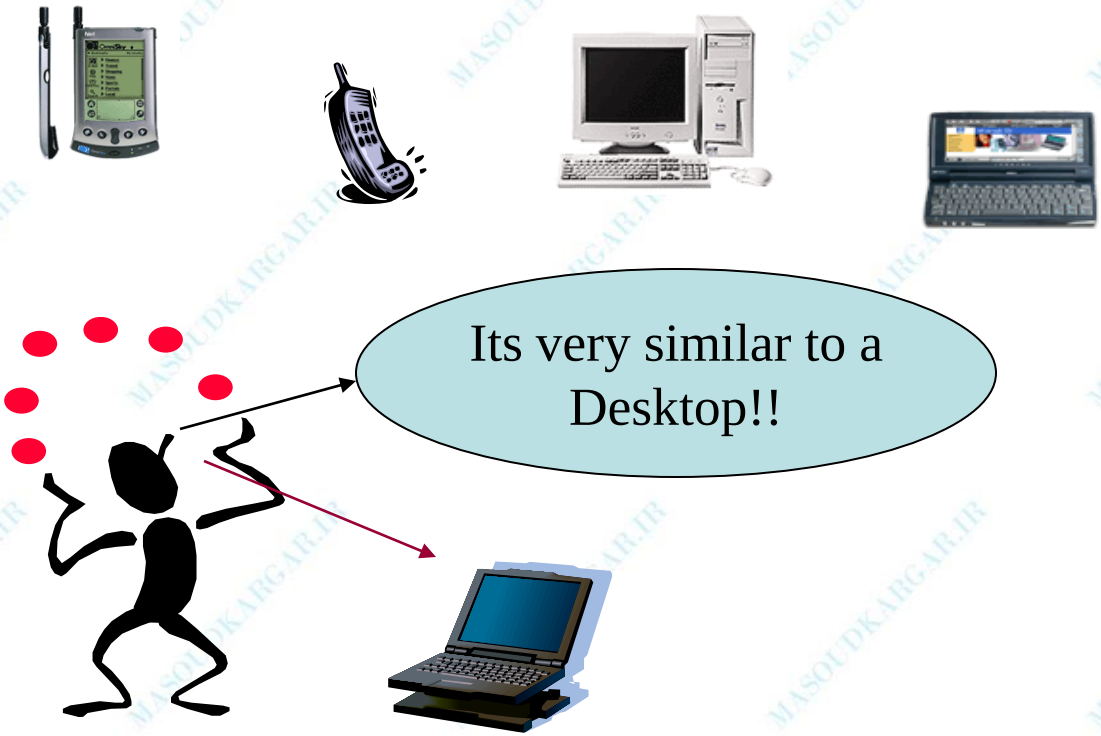
- در روشهائی که تاکنون بررسی کردیم، سعی بر این بود که با استفاده از مثالهای آموزشی تابعی پیدا کنیم که بتواند توصیف‌کننده داده‌ها باشد.

- در روش یادگیری IBL بسادگی فقط مثالها را ذخیره می‌کنیم و هرگونه تعمیم تا مشاهده مثال جدید به تعویق می‌افتد .به همین دلیل این روش گاهی روش تنبل یا lazy هم نامیده می‌شود.

- با مشاهده مثالهای جدید رابطه آن با نمونه‌های ذخیره شده بررسی شده و یک مقدار برای تابع هدف آن نسبت داده می‌شود.

در روش IBL یک فرضیه عمومی مشخص برای داده‌ها بدست نخواهد آمد بلکه دسته‌بندی هر نمونه جدید هنگام مشاهده آن و بر اساس نزدیکترین مثالهای ذخیره شده، انجام خواهد شد.

یادگیری مبتنی بر نمونه
Instance based learning IBL



یک تفاوت اساسی

- روش IBL برای هر نمونه جدید، تقریب جداگانه‌ای از تابع هدف را ایجاد می‌کند. این تقریب فقط به همسایگی نمونه جدید قابل اعمال بوده و هرگز نمی‌تواند بر روی فضای تمام نمونه‌ها عمل کند.

- کاربرد این روش هنگامی موثر است که تابع هدف خیلی پیچیده بوده ولی در عین حال قابل نمایش توسط توابع ساده‌تر محلی باشد.

مشخصه‌ها

- این روش دارای 3 مشخصه اصلی است:

1. تابع شباهت: مشخص می‌کند که دو نمونه چقدر نزدیک به هم هستند.انتخاب این تابع می‌تواند بسیار مشکل باشد .مثلا چگونه می‌توان شباهت رنگ موی 2 نفر را بیان نمود؟

2. انتخاب نمونه‌ها برای ذخیره: در این الگوریتم سعی می‌شود نمونه‌هائی ذخیره شوند که عمومی‌تر باشند. تشخیص اینکه آیا یک نمونه عمومیت دارد یا خیر، می‌تواند کار مشکلی باشد.

3. تابع دسته‌بندی‌کننده: تابعی است که بامشاهده یک مثال دسته‌بندی آنرا تعیین می‌کند.

مشکلات

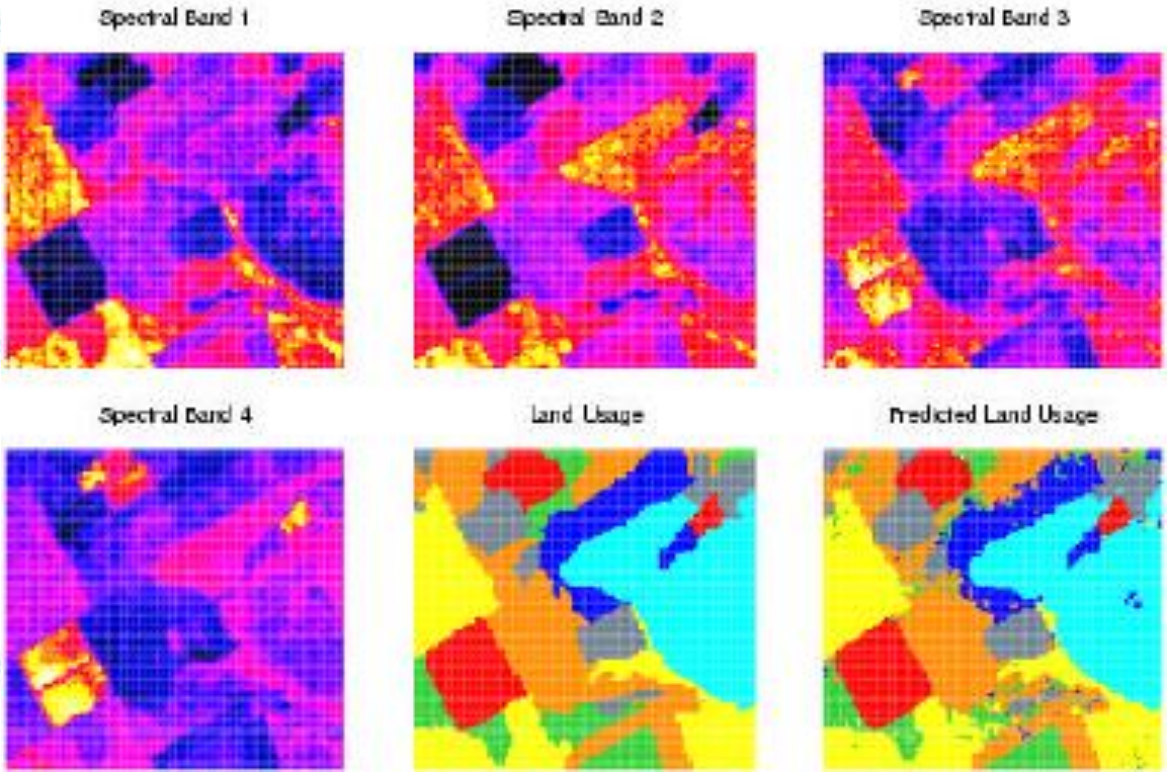
- دسته‌بندی داده جدید می‌تواند بسیار پرهزینه باشد. زیرا در مرحله آموزش عملی صورت نمی‌پذیرد و تمامی محاسبات در هنگام دسته‌بندی انجام می‌گردند.

- از این رو برای کاهش زمان دسته‌بندی از تکنیک‌های ایندکس استفاده می‌شود.

- در اغلب روشهای *IBL* برای بازیابی مثالهای مشابه از حافظه از تمامی ویژگی‌های موجود استفاده می‌شود. بنابراین اگر تابع هدف فقط به برخی از ویژگی‌ها بستگی داشته باشد، مثالهائی که واقعا مشابه هستند ممکن است بسیار از یکدیگر دور شوند.

مثالی از کاربردها

Image Scene Classification



برای هر تصویر با استفاده از مقادیر پیکسلهای آن یک *signature* محاسبه شده و از آن برای مقایسه تصویر ورودی با تصاویر موجود در پایگاه داده استفاده می‌شود.

مثالی از کاربردها

- *image size: 82x100 pixels*
- *each pixel is associated with 36(=(1+8)x4) features*
- *5NN is used for prediction*
- *error rate is about 9.5%*
- *5NN performs best among LVQ, CART, NN, …*

.

روشهای مختلف

- K-Nearest neighbor (*k-NN*)
  - Discrete Target Functions
  - Continuous Target Functions
  - Distance Weighted
- Locally weighted regression
- Radial basis function networks
- Case-based reasoning
- General Regression Neural Networks

K-Nearest Neighbor Learning (*k-NN*)

- *k-NN* ساده‌ترین و متداولترین روش مبتنی بر یادگیری نمونه است.

- در این روش فرض می‌شود که تمام نمونه‌ها نقاطی در فضای *n*بعدی حقیقی هستند و همسایه‌ها بر مبنای فواصل اقلیدسی استاندارد تعیین می‌شوند.

- مراد از k تعداد همسایه‌های در نظر گرفته شده است.

فاصله اقلیدسی

- اگر یک مثال دلخواه را به صورت یک بردار ویژگی نمایش دهیم:

$$< a_1(x), a_2(x), ... a_n(x) >$$

- فاصله بین دو مثال x_i و x_j به صورت زیر تعریف می‌شود:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

الگوریتم k-NN برای تابع هدف گسسته

برای یک تابع هدف گسسته به صورت $f : \Re^n \rightarrow V$, where V is the finite set $\{v_1,...v_s\}$

الگوریتم k-NN به صورت زیر است:

الگوریتم یادگیری

- هر مثال آموزشی $< x , f(x) >$ را به لیست *training_examples* اضافه کنید.

الگوریتم دسته‌بندی :

- *برای نمونه مورد بررسی* x_q :
  - *نزدیک‌ترین نمونه‌هائی از training_examples به آنرا با* $x_1 \ldots x_k$ *نمایش دهید.*
  - *مقدار زیر را محاسبه نموده و برگردانید .*

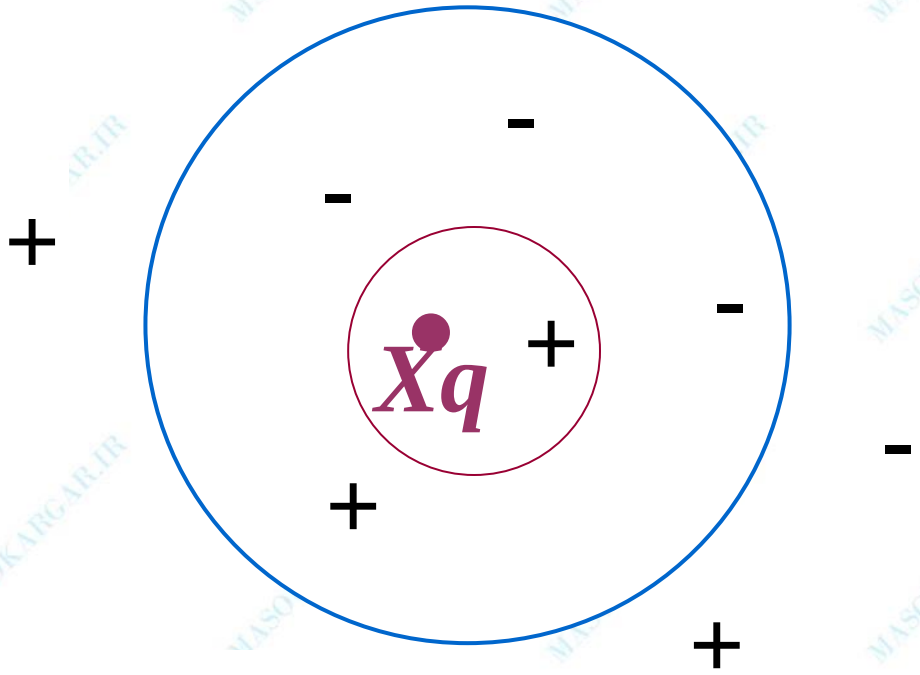
$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

where $\delta(a,b) = 1$ if $a = b$

$= 0$ otherwise

مثال

- اگر $k=1$ انتخاب شود الگوریتم $1\text{-}NN$ مقدار نزدیکترین نمونه به x_q را انتخاب خواهد نمود .برای مقادیر بزرگتر k متداولترین مقدار بین k همسایه نزدیک انتخاب خواهد شد.



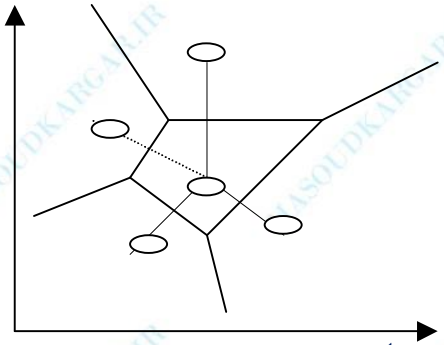
در این مثال x_q در حالت $1\text{-}NN$ مثبت و برای $5\text{-}NN$منفی خواهد بود.

فضای فرضیه

ماهیت فضای فرضیه ضمنی در نظر گرفته شده توسط الگوریتم *k-NN* چیست؟

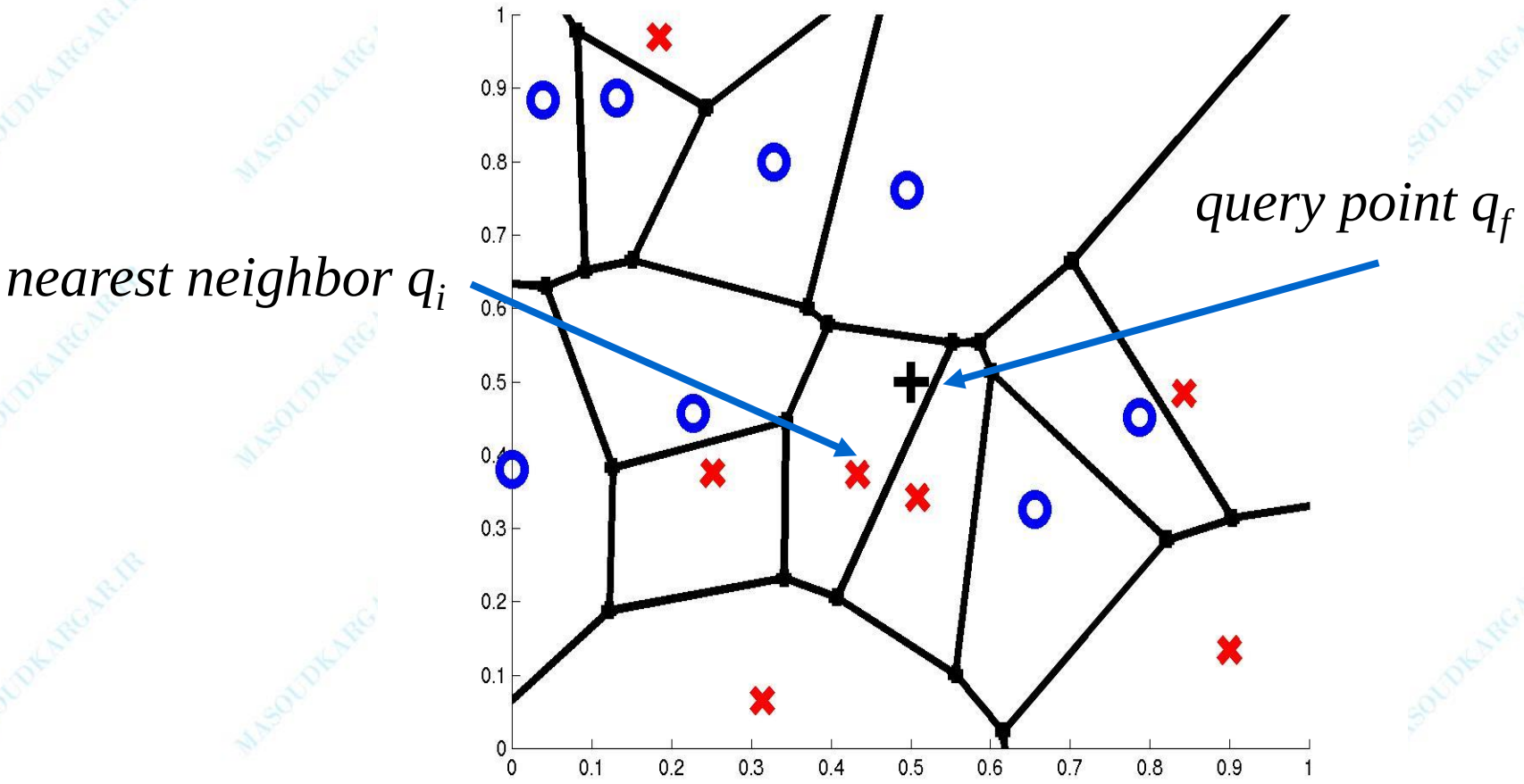
- اگرچه این الگوریتم هرگز فرضیه عمومی مشخصی ایجاد نمی‌کند، با این وجود ممکن است سطح تصمیم القا شده توسط الگوریتم برای یک فضای دو بعدی را به صورت ترکیبی از چندوجهی‌ها نشان داد که هر چند وجهی مجموعه‌ای از نقاطی را که توسط آن دسته‌بندی خواهند شد را مشخص می‌نماید.

- نقاط خارج چندوجهی نقاطی خواهند بود که توسط سایر چندوجهی‌ها دسته‌بندی خواهند شد.



- این نوع نمودار *Voronoi diagram* خوانده می‌شود.

Voronoi diagram



بایاس استقرا

- بایاس استقرا الگوریتم k-NN را می‌توان به صورت زیر در نظر گرفت:

دسته‌بندی یک نمونه مشابه دسته‌بندی نمونه‌های دیگری خواهد بود که در نزدیکی آن قرار دارند.

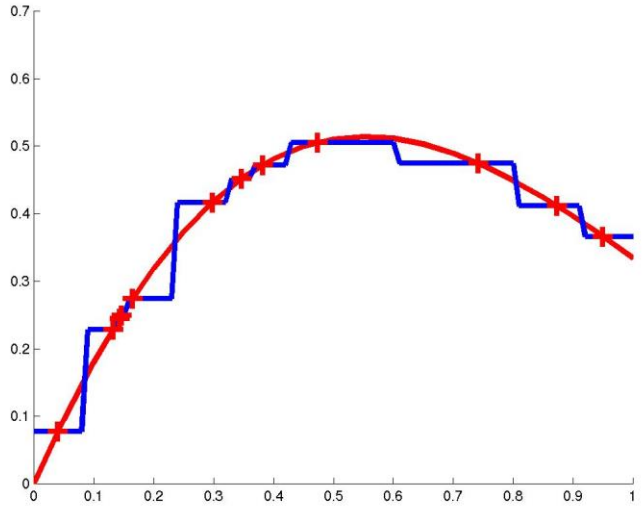
الگوریتم k-NN برای تابع هدف پیوسته

- الگوریتم k-NN را می‌تواند بسادگی برای توابع هدف پیوسته نیز استفاده نمود. در این حالت بجای انتخاب متداولترین مقدار موجود در همسایگی مقدار میانگین k مثال همسایه محاسبه می‌شود.
- در نتیجه در خط آخر الگوریتم از رابطه زیر استفاده می‌شود:

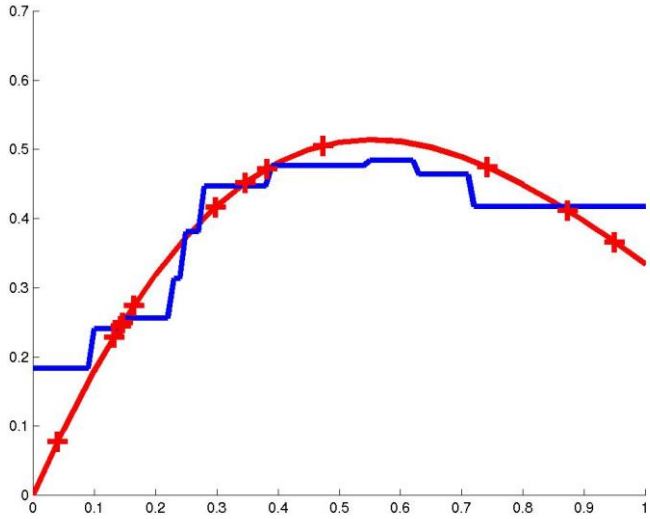
$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

k-NN برای تابع هدف پیوسته

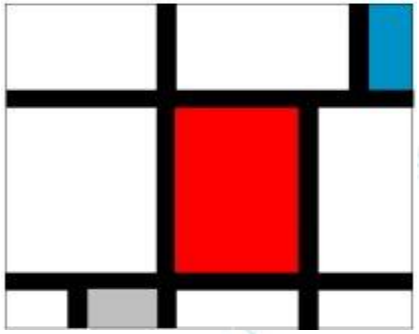
1-nearest neighbor



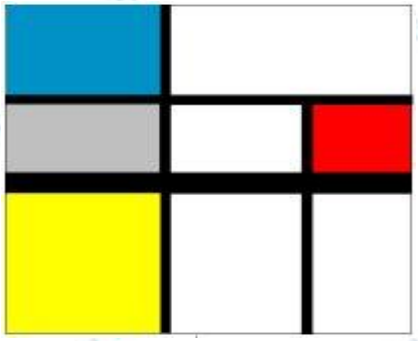
3-nearest neighbor



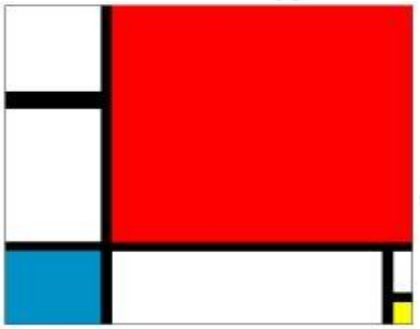
مثال



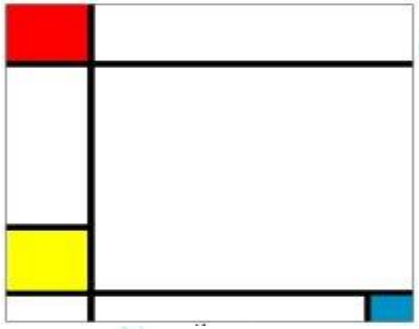
one



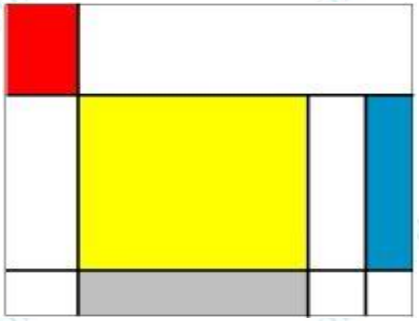
two



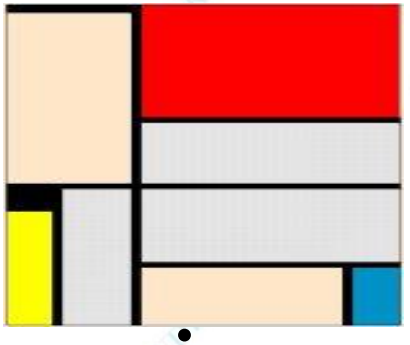
three



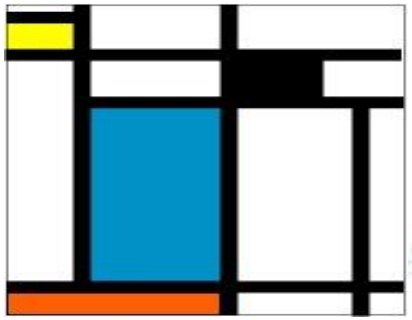
four



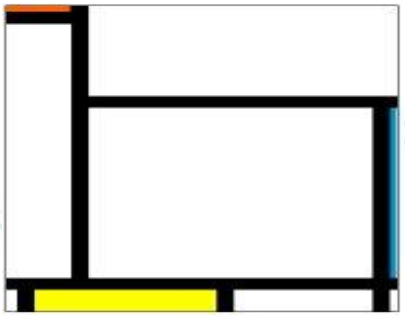
five



six



seven



Eight ?

Training data

Number	Lines	Line types	Rectangles	Colours	Mondrian?
1	6	1	10	4	*No*
2	4	2	8	5	*No*
3	5	2	7	4	Yes
4	5	1	8	4	Yes
5	5	1	10	5	*No*
6	6	1	8	6	Yes
7	7	1	14	5	*No*

Test instance

Number	Lines	Line types	Rectangles	Colours	Mondrian?
8	7	2	9	4	

نرمالیزه کردن داده‌های آموزشی

یک راه نرمالیزه کردن داده آموزشی $a_r(x)$ به $a'_r(x)$ عبارت است از

$$x_t' \equiv \frac{x_t - \bar{x}_t}{\sigma_t}$$

$$\bar{x}_t \equiv mean\ of\ t^{th}\ attributes$$

$$\sigma_t \equiv standard\ deviation\ of\ t^{th}\ attributes$$

Normalised training data

Number	Lines	Line types	Rectangles	Colours	Mondrian?
1	0.632	-0.632	0.327	-1.021	*No*
2	-1.581	1.581	-0.588	0.408	*No*
3	-0.474	1.581	-1.046	-1.021	Yes
4	-0.474	-0.632	-0.588	-1.021	Yes
5	-0.474	-0.632	0.327	0.408	*No*
6	0.632	-0.632	-0.588	1.837	Yes
7	1.739	-0.632	2.157	0.408	*No*

Test instance

Number	Lines	Line types	Rectangles	Colours	Mondrian?
8	1.739	1.581	-0.131	-1.021	

Distances of test instance from training data

Example	Distance of test from example	Mondrian?
1	2.517	No
2	3.644	No
3	2.395	Yes
4	3.164	Yes
5	3.472	No
6	3.808	Yes
7	3.490	No

Classification	
1-NN	Yes
3-NN	Yes
5-NN	No
7-NN	No

Distance-weighted k-NN

می‌توان عملکرد این الگوریتم را با در نظر گرفتن وزنی برای هر یک از k مثال همسایگی بهتر نمـود. این وزن بر اساس فاصله نمونه‌ها تا نمونه مورد بررسی اعمال می‌شود و معمولا با فاصله نمونه‌هـا رابطع معکوس دارد.

- در حالت گسسته:

$$\hat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(x_i)) \qquad \text{where } w_i = \frac{1}{d(x_q, x_i)^2}$$

- در حالت پیوسته:

$$\hat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i \, f(x_i)}{\sum_{i=1}^{k} w_i} \qquad \text{where } w_i = \frac{1}{d(x_q, x_i)^2}$$

در صورت اعمال وزن این امکان وجود خواهد داشت که به جای k نمونه همسایه از تمامی نمونـه‌هـا برای دسته‌بندی استفاده کنیم. اما این انتخاب باعث کند شدن عمل دسته‌بندی خواهد شد.

نکاتی در مورد الگوریتم *k-NN*

- الگـوریتم *Distance-weighted k-NN* بطـور مـوثری در مسائل عملی مختلفی برای استنتاج استقرائی بکار رفته است.

- این روش نسبت بـه نـویز مقـاوم بـوده و در مـواردی کـه داده آموزشی زیادی موجود باشد بسیار کاراست.

The curse of dimensionality

- از آنجائیکه برای محاسبه فاصله از تمامی ویژگی‌ها استفاده می‌شود این امکان وجود دارد که حتی ویژگی‌های نامرتبط در امر دسته‌بندی مورد استفاده قرار گیرند .این امر بر خلاف روشهائی مثل درخت تصمیم است که در آن سعی می‌شد تا فقط از ویژگی‌های مرتبط استفاده شود.
- برای مثال فرض کنید که هر نمونه با 20 ویژگی مشخص شوند که از میان آنان فقط 2 ویژگی برای دسته‌بندی کافی باشند در این صورت ممکن است نمونه‌های ذخیره شده‌ای که در این دو ویژگی مشابه هستند بسیار از هم فاصله داشته باشند. در این صورت معیار فاصله مورد استفاده در k-NN می‌تواند بسیار گمراه‌کننده باشد.
- این مسئله *curse of dimensionality* نامیده می‌شود.

Cross-validation

- یک راه حل این مشکل استفاده از وزن بیشتر برای ویژگی‌های مرتبط است. این امر مشابه تغییر مقیاس محورهاست: محور ویژگی‌های مرتبط کوتاهتر و محور ویژگی‌های نامرتبط طولانی‌تر می‌شوند.
- برای تعیین وزن ویژگی‌ها می‌توان از روش *cross-validation* استفاده نمود:
  - مجموعه‌ای از داده‌ها به عنوان داده‌های آموزشی انتخاب می‌شوند.
  - مقادیر $z_1,\ldots,z_n$ بعنوان ضرایبی که باید در هر محور ضرب شوند انتخاب می‌گردند. این انتخاب به نحوی است که خطای دسته‌بندی در باقیمانده مثالها کاهش یابد.
  - می‌توان با قرار دادن $z_j=0$ اثر یک ویژگی را بکلی حذف نمود.

Indexing

از آنجائیکه در روش *K-NN* دسته‌بندی مثالها تا زمان برخورد با آن مثال به تعویق می‌افتد استفاده از *Indexing* برای مرتب کردن مثالهای آموزشی می‌تواند بطور چشمگیری کارائی الگوریتم را افزایش دهد.

روش *kd-tree* یک روش برای ایندکس کردن است که در آن نمونه‌ها در سطح یک درخت ذخیره شده و نمونه‌های نزدیک به هم در همان گره و یا گره‌های نزدیک به هم ذخیره می‌شوند.

ویژگیهای یادگیری نمونه

- مزایا:
  - می‌تواند توابع پیچیده را مدل کند.
  - اطلاعات موجود در مثالهای آموزشی از بین نمی‌رود.
  - می‌تواند از نمایش سمبلیک نمونه‌ها استفاده کند.
- معایب:
  - بازده الگوریتم هنگام انجام دسته‌بندی کم است.
  - تعیین یک تابع فاصله مناسب مشکل است.
  - ویژگیهای نامرتبط تاثیر منفی در معیار فاصله دارند.
  - ممکن است به حافظه بسیار زیادی نیاز داشته باشد.

واژگان

- *Regression*: عبارت است از تقریب یک تابع با مقدار حقیقی.

- *Residual*: عبارت است از مقدار خطای حاصل از تقریب تابع.

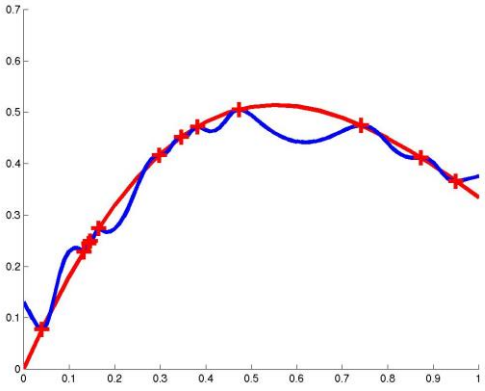
- *Kernel Function*: عبارت است از تابعی که با استفاده از فاصله، مقدار وزنهای مثالهای آموزشی را معین می‌کند.

توابع Kernel

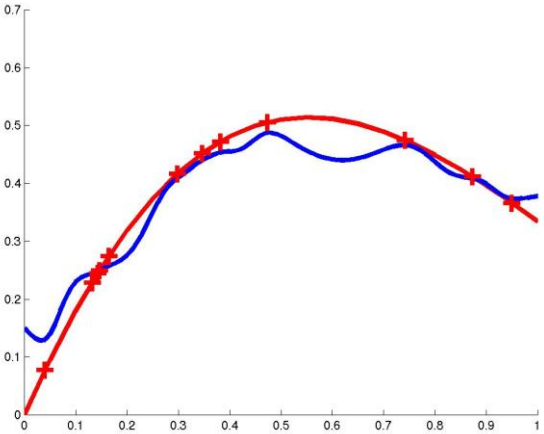
- معمولا با فاصله رابطه معکوس دارند تا نقاط نزدیکتر وزن بیشتری بگیرند.
- $K(d(x_i,xq))$
  - $1/d^2$ –
  - e^{-d} –
  - $1/(1+d)$ –

توابع Kernel

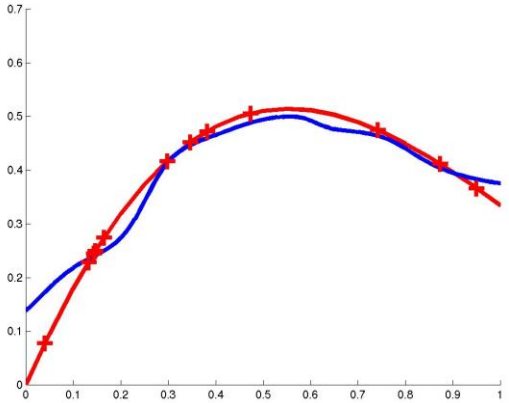
$K(d(x_q,x_i)) = 1/ d(x_q,x_i)^2$



$K(d(x_q,x_i)) = 1/(d_0+d(x_q,x_i))^2$



$K(d(x_q,x_i)) = exp(-(d(x_q,x_i)/\sigma_0)^2)$

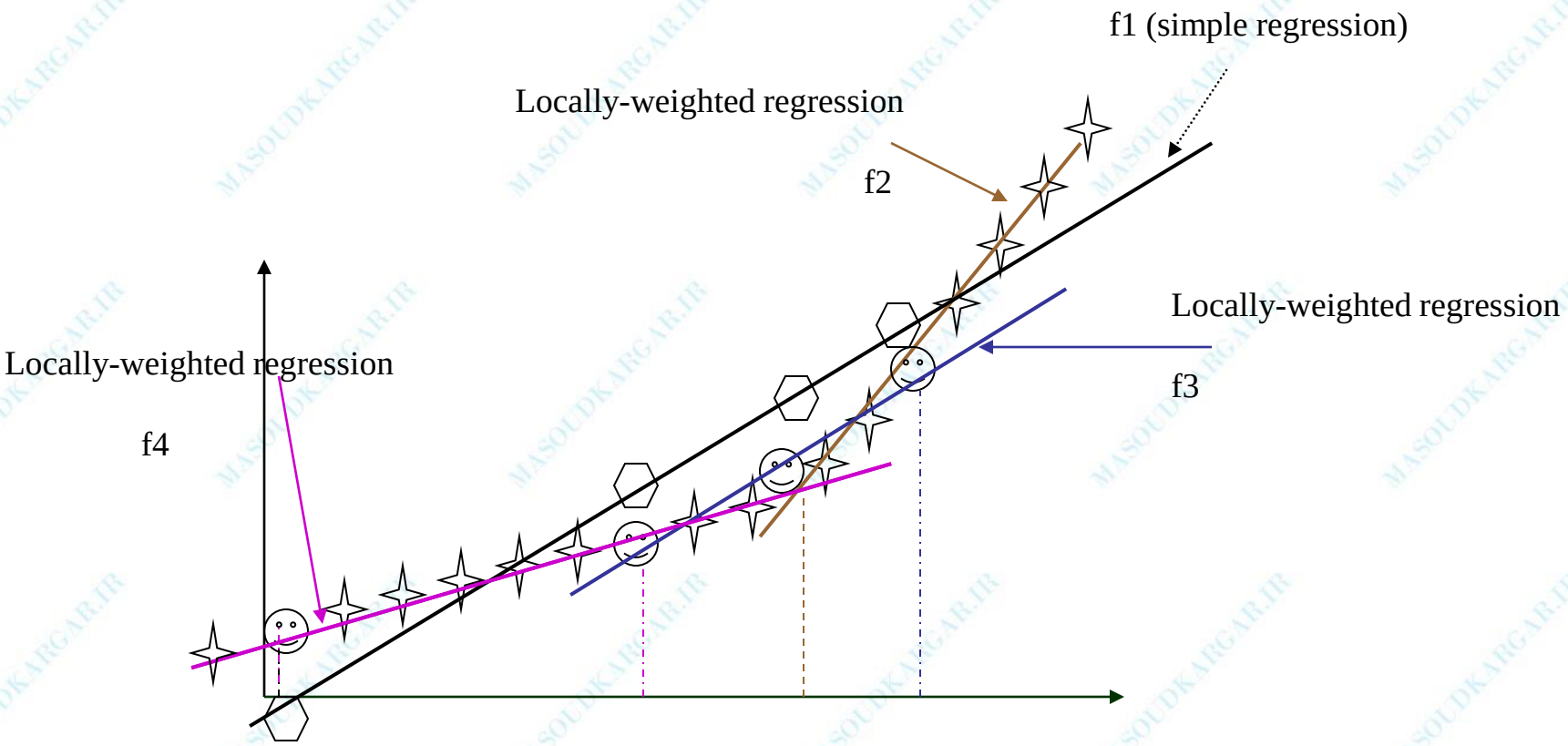


Locally Weighted Regression

- الگوریتم LWR تعمیمی بر الگوریتم $K\text{-}NN$ است که تقریب صریحی از تابع f حول ناحیه محلی در برگیرنده نمونه مورد بررسی x_q بدست می‌دهد.
- این تقریب محلی با استفاده از مثالهای نزدیک هم و یـا مثالهـای *distance weighted* انجام می‌شود.
- این تابع ممکن است یک تابع خطی، درجه دو و یا یک شبکه عصبی باشد.

دلیل نامگذاری:

- *local*: از مثالهای نزدیک نمونه مورد بررسی استفاده می‌کند.
- *Weighted*: اثر هر مثال آموزشی با در نظر گرفتن فاصله آن منظور می‌شود.
- *Regression*: برای تقریب یک تابع با مقدار حقیقی بکار می‌رود.



Training data

Predicted value using simple regression

Predicted value using locally weighted (piece-wise) regression

Locally Weighted Linear Regression

- این روش از یک تابع خطی برای تقریب تابع هدف در نزدیکی مثال مورد بررسی استفاده می‌کند:

$$\hat{f}(x) = w_0 + w_1 a_1(x) + \ldots + w_n a_n(x)$$

- این تابع مشابه تابع مورد استفاده در فصل 4 برای محاسبه وزنهای یک شبکه عصبی است که در آن وزنها طوری انتخاب می‌شدند که مقدار خطای زیر حداقل گردد:

$$E(\vec{w}) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2$$

- که برای اینکار از قانون آموزش *gradient descent* استفاده می‌شد.

$$\Delta w_i = \eta \sum_{x \in D} (f(x) - \hat{f}(x)) a_i(x)$$

رابطه محلی؟

- قانون دلتا یک رویه تقریب کلی است در حالیکه در روش *nearest neighbor* به دنبال یک رابطه برای تقریب محلی هستیم.

- سوال: چگونه می‌توان با استفاده از رابطه کلی قانون دلتا رابطه محلی مورد نظر را بدست آوریم؟

استفاده از خطای محلی

- به نظر می‌رسد که ساده‌ترین راه، تعریف مجدد رابطه خطاست به نحویکه با مثالهای محلی آموزشی تطبیق نماید.
- اینکار را به سه روش می‌توان انجام داد:

1-استفاده از k مثال همسایگی

$$E_1(x_q) = \frac{1}{2} \sum_{x \in k-NN} (f(x) - \hat{f}(x))^2$$

2-استفاده از تمامی مثالها با تخصیص یک مقدار وزنی به آنها

$$E_2(x_q) = \frac{1}{2} \sum_{x \in D} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

3-ترکیبی از روشهای 1 و2

$$E_3(x_q) = \frac{1}{2} \sum_{x \in k-NN} (f(x) - \hat{f}(x))^2 K(d(x_q, x))$$

قانون تغییر وزنها

- در انتخابهای فوق:
- E_1 فاصله را در نظر نمی‌گیرد.
- E_2 جالبتر از همه بوده اما محاسبه آن پرهزینه است.
- E_3 یک انتخاب بینابین است.
- با انتخاب E_3 می‌توان قانون دلتا را برای یادگیری وزنها به صورت زیر نوشت:

$$\Delta w_i = \eta \sum_{x \in k-NN} K(d(x_q, x))(f(x) - \hat{f}(x)) a_i(x)$$

انتخاب مقدار k

- اگر k خیلی کوچک باشد، نسبت به نویز حساس خواهد بود.
- اگر k خیلی بزرگ باشد ممکن است یک همسایگی نقاطی از سایر کلاسها را نیز در برگیرد.