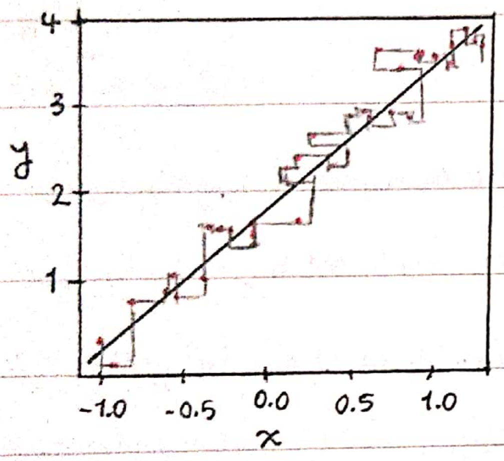
/ /

ML.7

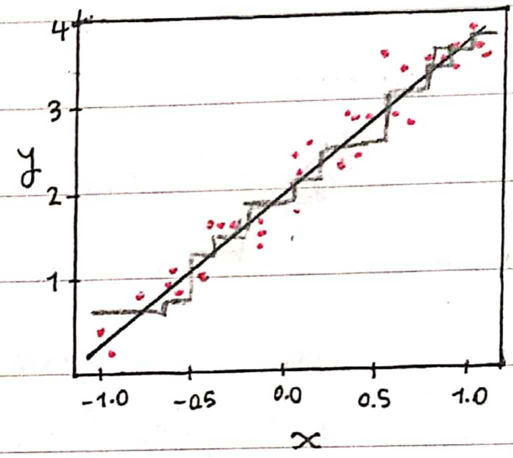
در این جلسه درباره‌ی logistic regression ، classification توضیح داده می‌شود.

در ابتدا نکاتی که از کتاب KNN و linear regresion با مقایسه است گفته خواهد شد.

K-nearest neigbor regression



k=1



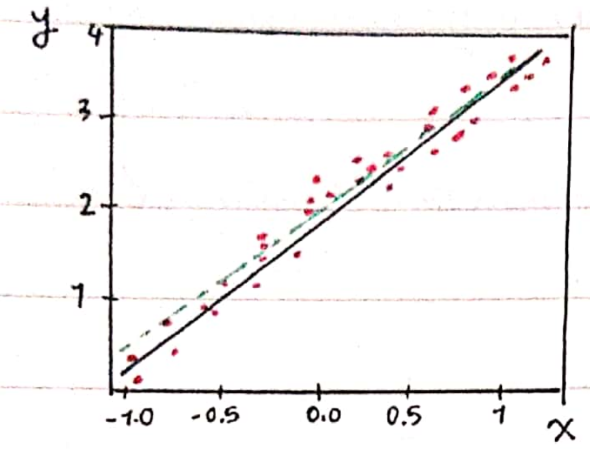
k=9

Ref: ISL book , Fig 3.17

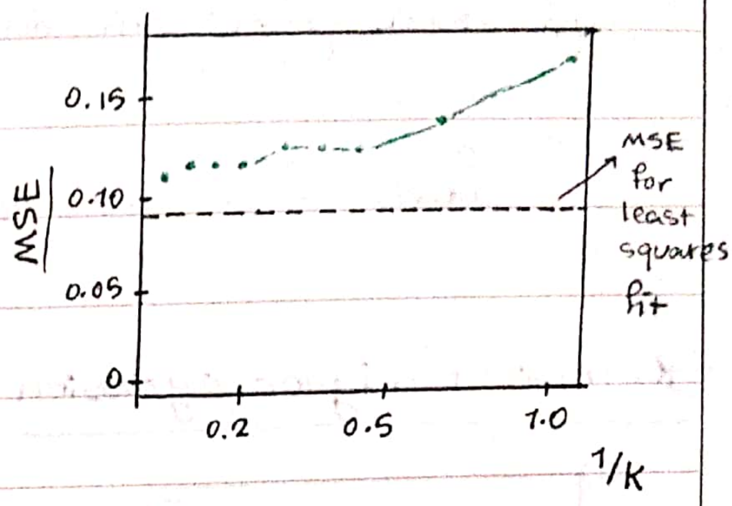
همانطور که قبلاً هم گفته شد در KNN وقتی k کوچک باشد، KNN مدل پیچیده‌ای است و هرچه k بزرگتر شود مدل ساده‌تر می‌شود. در کتاب ISL یک سری مقایسه بین KNN و LR صورت گرفته است.

در شکل 3.18 کتاب ISL ، با فرض اینکه مدل مشخص است (خط سیاه رنگ)، یک سری داده simulate شده است که با نقاط قرمز رنگ مشخص شده است و یکبار linear regression را برای داده‌ها به کار برده و least squares fit را محاسبه کرده و یکبار هم از KNN استفاده با k های مختلف استفاده کرده.

least squares fit



The MSE



ISL book- Fig 3.18

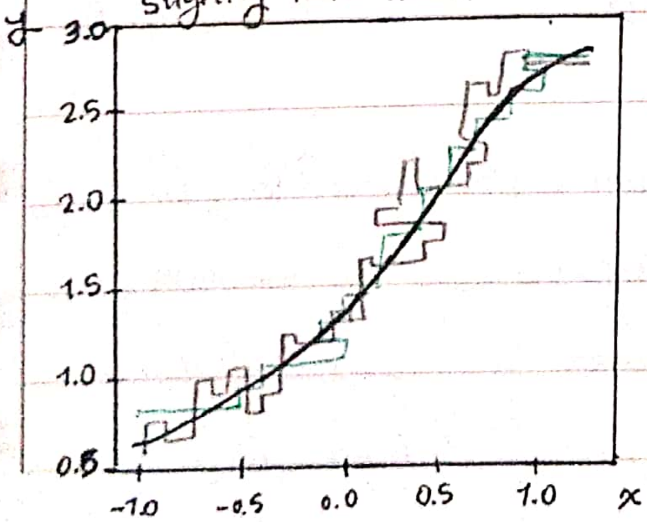
در شکل سمت چپ linear regression model را با خط سبز رنگ مشخص کرده است. در شکل سمت راست تستِ MSE را برای LR و KNN با K های مختلف حساب کرده است. خط چین مشکی خطای مدل LR است و خط سبز خطای KNN.

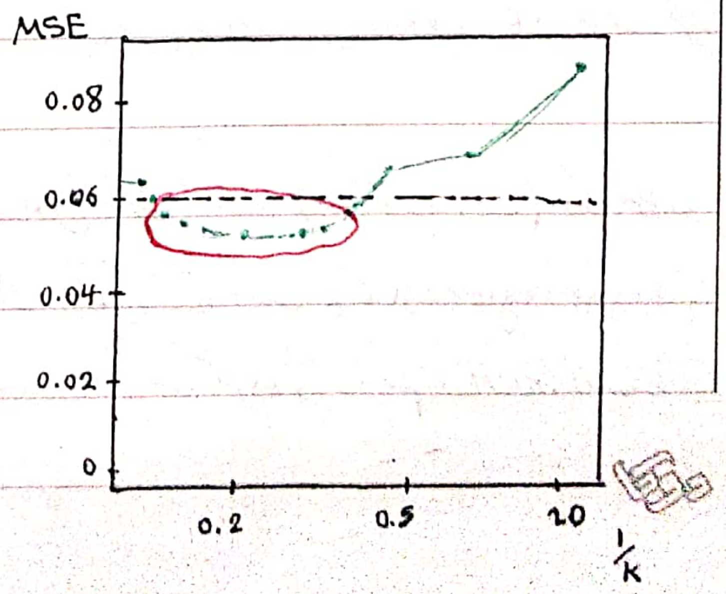
این مقایسه نشان می دهد که اگر فرضیات LR را داشته باشیم بهتر و با خطای کمتری جواب می دهد و برای K های مختلف این امر صادق است.

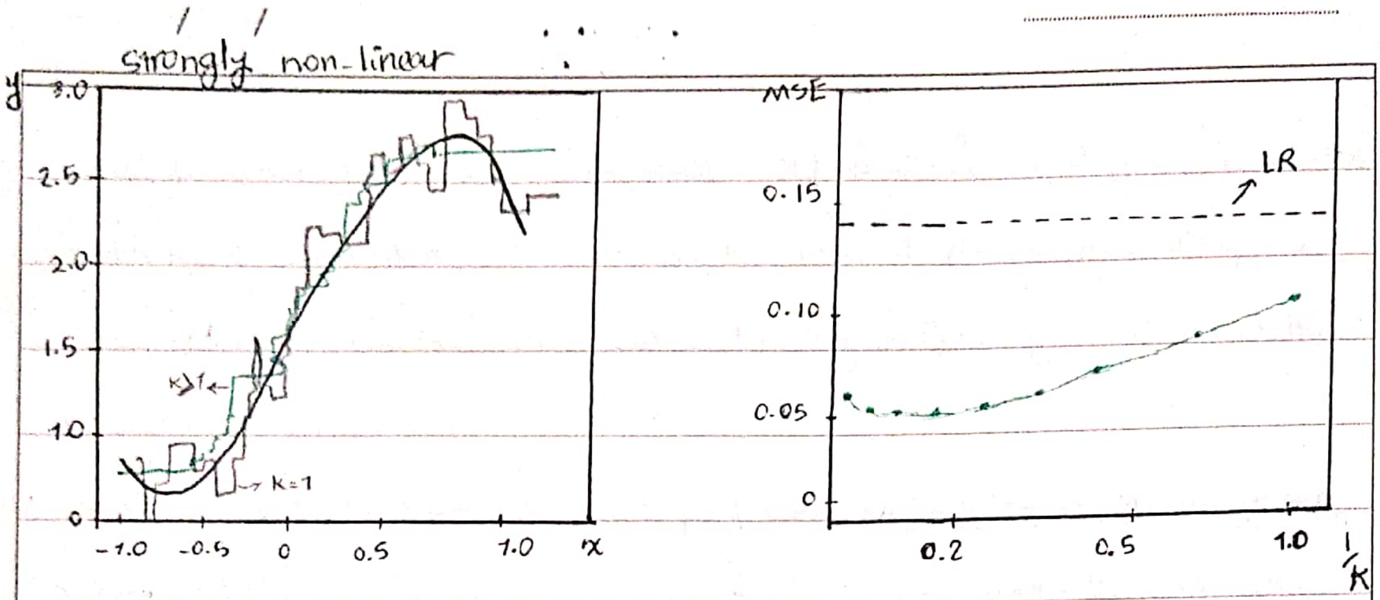
اما در مدل KNN هرچه K بزرگتر باشد (یعنی $\frac{1}{K}$ کوچکتر)، مدل پیچیده‌تر می شود و خطای آن تستِ کمتر می شود.

Non-linear relationship between X and Y

slightly non-linear







ISL book - Fig 3.19

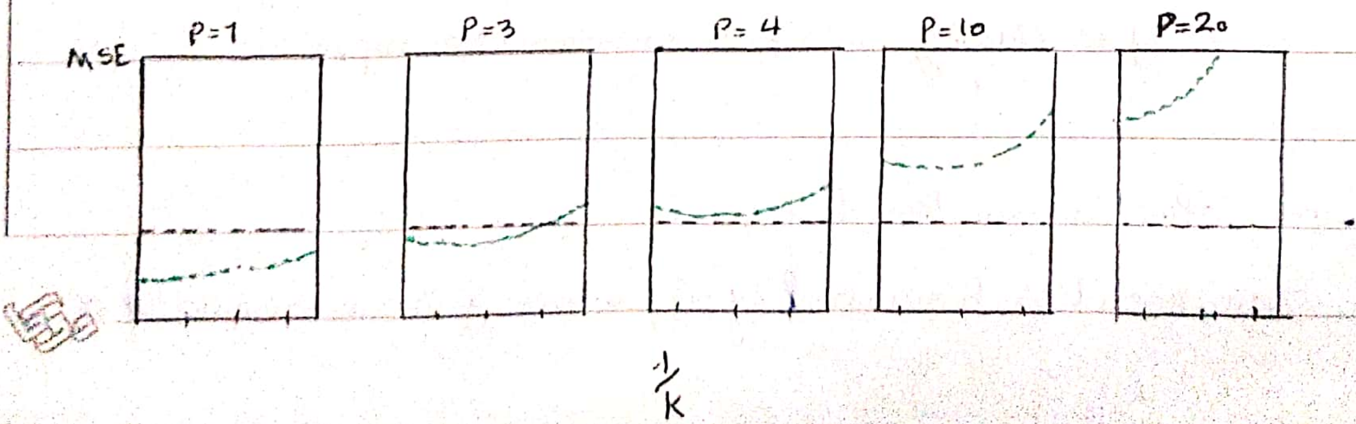
درحالت اول شکل ۴ که slightly non-linear است، در رنج خاصی از K ها، مدل KNN مدل بهتری است که علامت گذاری شده است.

اما برای حالتی که strongly non-linear است، مدل KNN با همه‌ی حالات مختلف K مدل خوب و با خطای کمتری محسوب می‌شود.

البته این نتایج فقط

Curse of dimensionality

- K observations that are close to x_0 may be very far away in p-dimensional space when p is large.
- By our definition, Y only depends on the first predictor. The additional predictors are noise.



ISL book - Fig 3.20

در شکل قبل تر simulation انجام شده تا KNN با LR مقایسه شود. فرض بر این است که مدل واقعی صرفاً به یکی از predictor ها وابسته است. یعنی اگر مثلاً 20 تا متغیر هم داشته باشیم، مدل واقعی صرفاً به اولی وابسته است و بقیه Random اند و هیچ ارتباطی بین y و سایر predictor ها وجود ندارد.

در نمودار اول p=1 است و صرفاً یک متغیر داریم، و KNN عملکرد بهتری نسبت به LR دارد و هر تعداد predictor ها افزایش می‌یابد مدل LR تغییر چندانی نکرده است ولی مدل KNN خطای زیادی پیدا کرده است. البته خطای LR هم کمی افزایش یافته است و آن هم به خاطر predictor هایی است که noise هستند.

و علت افزایش خطای KNN در ابعاد بزرگ هم این است که در ابعاد بزرگ فاصله بین نقطه‌ی مورد نظر تا نقاط همسایه بسیار مهم است. در ابعاد بزرگ باید تعداد خیلی زیادی data point داشته باشیم که فاصله ها به همدیگر نزدیک شوند.

Classification

- The response variable is qualitative/categorial.

$$D=\{(x[1],y[1]),\dots,(x[n],y[n])\}$$

example : 1) Input: gene expression data, response var: origion of cancer

2) to classify an email to spam/non-spam

- we define a new loss function:

Zero-one loss function: $\ell(y\neq\hat{y})$ where $\hat{y}$ is an estimate for x.

در اینجا از تعریف جدیدی از loss function استفاده می‌کنیم ← Zero-one loss function

در این تعریف باید تعداد $\hat{y}$ هایی که با y واقعی یکسان نشدند را بشماریم. در واقع تعداد $\hat{y}$ هایی که misclassify شده است را به دست می‌آوریم.

Training error rate:

$$\boxed{\frac{1}{n}\sum_{i=1}^{n} I(y[i] \neq \hat{y}[i])}$$

Test error: $\boxed{E(I(y \neq \hat{y}))}$ → expectation روی داده‌ی unseen

$I \rightarrow$ Indicator تابع $\Longrightarrow I(z) = \begin{cases} 1 & z = \text{True} \\ 0 & z = \text{False} \end{cases}$

The Bayes classifier:

- Assign each data point to the most likely class, given its predictor values.

- Bayes classifier حالت تئوری دارد و برای تحلیل می‌توان از آن استفاده کرد ولی در عمل نمی‌توان آن را ساخت.

- Formally, the Bayes classifier is defined.

$$\boxed{\hat{y} = \arg\max_{c} P(Y=c \mid x)}$$

به ازای هر x که داریم، $P(Y=c)$ که c تک تک کلاس‌ها است را حساب می‌کنیم، هرکدام بیشتر بود به عنوان $\hat{y}$ در نظر می‌گیریم.

- اما چرا فرمول مذکور را نمی‌توان در عمل حساب کرد؟ چون احتمال را نداریم. چون P همان توزیع واقعی data است که حتماً آن را نمی‌دانیم. (و این P واقعی فقط در مورد simulation داریم اما در بقیه‌ی مواقع نه)

اما اگر P را داشته باشیم، Bayes classifier، بهترین classifier ممکن است.

- The Bayes error rate: → معادل خطای غیرقابل کاهش در LR $(f(x) = \hat{f}(x) + \varepsilon)$

* The Bayes classifier produces the lowest possible test error rate. (Bayes classifier، min خطای ممکن را می‌دهد)

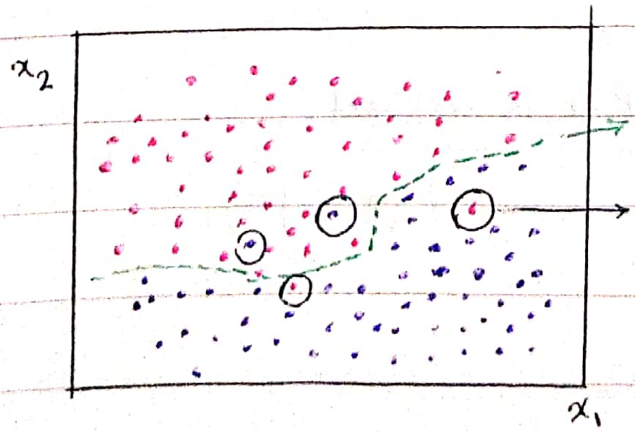
$$\boxed{1 - E\left(\max_c P(Y=c|x)\right)}$$

expectation over all possible values for x

مثلاً فرض کنید برای یک x خاص در ۹۰٪ کلاس c_1 و در ۱۰٪ موارد کلاس c_2 می‌شود.
پس حتی اگر بهترین حالت هم باشد، ۱۰٪ مواقع error دارد → bayes error
این min خطای ممکن است که معادل خطای غیرقابل کاهش در LR است.

The Bayes oprimal decision boundry

- Two class problem:



این موارد خطای bayes هستند که هرچه هم مدل را بهتر کنیم ($\uparrow P$) کاهش نمی‌یابند.

/ /

- Bayes decision boundry را فقط در داده‌های simulation می‌توانیم استفاده کنیم و در داده‌های واقعی نمی‌توان از آن استفاده کرد.

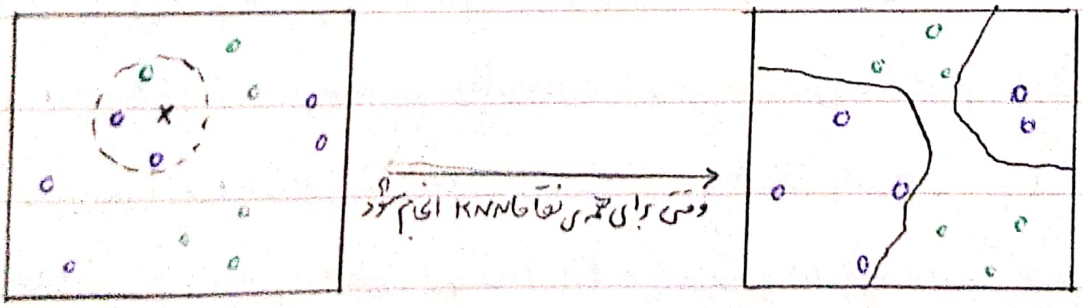
* Given the generating density for each class, we can calculate the boundry exactly.

K-nearest neighbor classifier

* $P(Y|x)$ is not known for real data, so computing the Bayes classifier is impossible!
* Many classifiers attempt to estimate $P(Y|x)$.
* K-nearest neighbor classifier is such a method:

$$\boxed{P(Y=c|x) = \frac{1}{k} \sum_{i \in N_0} I(y[i]=c)}$$

ازآنجایی که در مدلهای واقعی P را نداریم، روش‌های مختلف ML سعی در تخمین $P(Y|X)$ دارند که KNN classifier یکی از آنهاست و با فرمول majority voting عمل می‌کند.



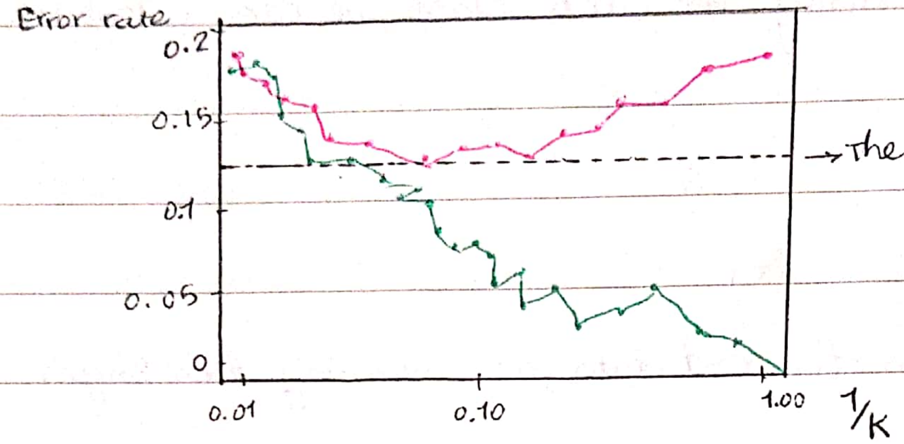
k=3

اگر k کوچک باشد مدل پیچیده‌تر و اگر k بزرگ باشد مدل ساده‌تر می‌شود.

/ /

Training and test error rates

- 200 training obs , 5000 test obs.



logistic regression

* Why not linear regresion for classification problem ?

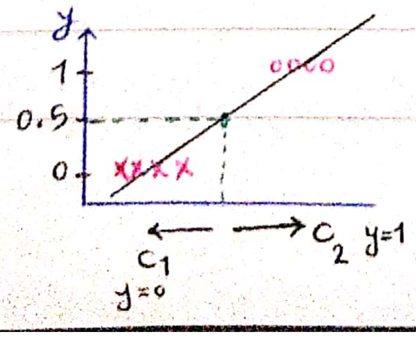
- Approach :
  - performing linear regression
  - And classifying according to $\hat{Y} > 0.5$

- logistic regression یک مدل خطی است ، اولین سؤالی که پیش می‌آید این است که اگر بخواهیم یک مسئله‌ی classification را حل کنیم چرا از linear regression استفاده نمی‌کنیم.

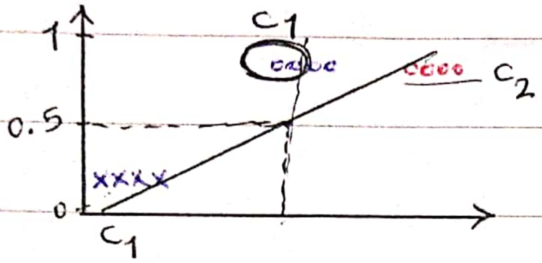
ایده : یک LR را انجام بدهیم و classify را بر این اساس که $\hat{Y} > 0.5$ باشد انجام بدهیم.

مثلاً فرض کنید ، اینجا سؤال روی binary classification است و $y = \{\overset{C_1}{0}, \overset{C_2}{1}\}$ است.

حت. حالا اگر LR بزنیم ، خطی بین نقاط fit می‌کند و اگر $y > 0.5$ بود ، می‌گوییم C_1 و اگر $y \leq 0.5$ جزء کلاس C_2 است.



حالا اگر به شکل قبل چند نقطه اضافه کنیم ← تعدادی از نقاط کلاس 2 در کلاس 1 قرار می‌گیرد. پس حتی در چنین حالت ساده‌ای از classification ، LR نمی‌تواند به خوبی عمل کند.



یک اشکال دیگر هم این است که بعضی از estimate ها بزرگتر از یک است و خارج از محدوده‌ی صفر و یک در نظر گرفته شده است.

* some estimates are outside [0,1]

- logistic regression

- logistic Regression یک extension هست که بر روی مدلهای linear دارند تا بتوانند در classification از آن استفاده کنند. (در binary classification)

ایده کلی آن هم این است که خروجی ها ($\hat{y}$) را force کند تا بین صفر و یک باشد.

Idea: enforce the responses to be in [0,1]

و تفسیری هم که از خروجی می‌شود به صورت احتمال است. مثلاً اگر $y=0.2$ باشد، با احتمال 20% به کلاس C_1 متعلق است و مثلاً اگر $y=0.9$ باشد یعنی با احتمال 90% به کلاس 1 تعلق دارد، چون که y برای کلاس C_1 را 1 در نظر گرفته ایم.

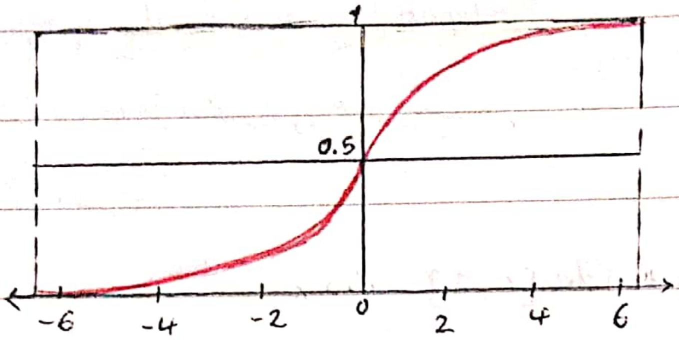
$$\boxed{Z = \beta^T x = \underline{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3}}$$

مقدار این عبارت می‌تواند بین $+\infty$ تا $-\infty$ باشد. بنابراین Z را با یک تابع سیگموئید، (logistic function or sigmoid function) ، به فضای $g(z)$ در بازه‌ی صفر تا یک می‌برند ($0 < g(z) < 1$)

$$g(t) = \frac{1}{1+e^{-z}} \rightarrow \begin{cases} g(z)=0 & z=-\infty \\ g(z)=0.5 & z=0 \\ g(z)=1 & z=+\infty \end{cases}$$

اگر Z منفی باشد کلاس 1 انتخاب می‌شود.

اگر Z + باشد، کلاس 2



classifier که در مدل logistic regression ساخته شده است به این صورت است:

$$\boxed{f_\beta(x) = g(\beta^T x) = \frac{1}{1+e^{-\beta^T x}}}$$

این یک مدل logistic regression است که غیرخطی است و هر X به آن بدهیم، عددی بین صفر و یک می‌دهد و این مدل یک سری پارامترهای مخصوص به خود را دارد.

- How to estimate the parameters?

Given $D=\{(x[1],y[1]),\dots,(x[n],y[n])\}$

$x[n] \in R^d$

$y[n] \in \{0,1\}$

فرضیات logistic reg است.

(ضمناً در SVM به جای $\{0,1\}$، $y[n]\in\{-1,1\}$ در نظر می‌گیریم.)

[در اینجا چون در فرضیات logistic، بحث احتمالاتی داریم $y\in\{0,1\}$ تا محاسبات راحت‌تر باشد.]

Scanned by CamScanner

حالا اگر بخواهیم یک مدل احتمالاتی بسازیم:

$$\boxed{Y = \text{Bernoulli}\left(g(\beta^T x)\right)}$$

حالا اگر بخواهیم این مدل را تخمین بزنیم:
می توانیم از MLE استفاده کنیم.

$$\text{MLE}: \quad \hat{\beta}_{ML} = \arg\max_{\beta} L(\beta)$$

چون y ها random است، y را به شرط β و x ها می نویسیم.

fixed

likelihood

$$L(\beta) = P\left(y[1], \dots, y[n] \mid \beta, x[1], \dots, x[n]\right)$$

می توانیم به صورت ضرب احتمالات بنویسیم چون obs ها مستقل اند.

$$= \prod_{i=1}^{n} P(y[i] \mid \beta, x[i])$$

$$\alpha[i] = g(\beta^T x[i]) = P(y[i]=1 \mid x[i], \beta)$$

$$= \prod_{i=1}^{n} \alpha[i]^{y[i]} (1-\alpha[i])^{1-y[i]}$$

همان توزیع برنولی

$$\begin{cases} P(y[i]=1 \mid \beta, x[i]) = \alpha[i] \\ P(y[i]=0 \mid \beta, x[i]) = 1-\alpha[i] \end{cases}$$

$$\boxed{\ell(\beta) = \sum_{i=1}^{n} \underbrace{y[i]\log\alpha[i] + (1-y[i])\log(1-\alpha[i])}_{E[i]}}$$

maximize $\ell(\beta)$! $\Rightarrow$ + no analytical solution

چون ترم های غیرخطی دارد.

* gradient ascent algorithm

$$\boxed{\beta := \beta + \lambda \nabla_{\beta}\ell(\beta)}$$

در gradient descent خلاف شیب حرکت می‌کردیم ولی در gradient ascent در جهت آن.

$$\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^{n} \frac{\partial E[i]}{\partial \beta_j} = \boxed{\frac{\partial E[i]}{\partial \alpha[i]} \times \frac{\partial \alpha[i]}{\partial z[i]} \times \frac{\partial z[i]}{\partial \beta_j}}$$

← قاعده‌ی chain rule

$$= \frac{y[i]}{\alpha[i]} - \frac{1-y[i]}{1-\alpha[i]} \times \alpha[i](1-\alpha[i]) \times x_j[i]$$

توضیح بیشتر

$$\frac{\partial \alpha[i]}{\partial z[i]} = \frac{\partial g(z)}{\partial z} = \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) = \left[\frac{0 \times (1+e^{-z}) + (e^{-z})(1)}{(1+e^{-z})^2}\right.$$

HW.

$$= \left(\frac{e^{-z}}{1+e^{-z}}\right)\underbrace{\left(\frac{1}{1+e^{-z}}\right)}_{g(z)} = \left(\frac{e^{-z}}{1+e^{-z}}\right)(g(z)) \xrightarrow[\text{+1 −1 می‌کنیم}]{\text{عبارت اول را}} \left(\frac{1+e^{-z}-1}{1+e^{-z}}\right)(g(z)) = \Big(1 - \underbrace{\frac{1}{1+e^{-z}}}_{g(z)}\Big)(g(z))$$

$$= (1-g(z))\, g(z)$$

HW:

$$\frac{\partial l(\beta)}{\partial \beta_j} = \left[\frac{y[i] - \cancel{y[i]\alpha[i]} - \alpha[i] + \cancel{\alpha[i]y[i]}}{\cancel{\alpha[i](1-\alpha[i])}} \times \cancel{\alpha[i](1-\alpha[i])} \times x_j[i]\right]$$

$$\boxed{= (y[i] - \alpha[i])\, x_j[i]}$$

$$\Rightarrow \boxed{\frac{\partial l(\beta)}{\partial \beta_j} = \sum_{i=1}^{n} (y[i] - \alpha[i])\, x_j[i]}$$

$\hat{y}[i] = \beta^T x[i]$

در فرمول Linear R مشابه همین فرمول بود فقط به جای $\alpha[i]$، $\hat{y}[i]$ بود.

میانگین توزیع برنولی

Vector form:

$$y=\begin{bmatrix} y[1] \\ \vdots \\ y[n] \end{bmatrix} \qquad \alpha=\begin{bmatrix} \alpha[1] \\ \vdots \\ \alpha[n] \end{bmatrix} \qquad x=\begin{bmatrix} x_1[1] & \cdots & x_d[1] \\ & \vdots & \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial \ell(\beta)}{\partial \beta_1} \\ \vdots \\ \frac{\partial \ell(\beta)}{\partial \beta_d} \end{bmatrix} = \nabla \ell(\beta) = x^T (y-\alpha)$$

gradient ascent algorithm:

$$\boxed{\beta \mathrel{+}= \beta + \lambda x^T (y-\alpha)}$$

Hessian

$$\nabla^2 \ell(\beta) = \underbrace{-\underbrace{x^T W x}_{\text{SPD}}}_{}$$

$$W=\begin{bmatrix} w_{11} & & 0 \\ & \ddots & \\ 0 & & w_{nn} \end{bmatrix}$$

$$\boxed{w_{ii}=\alpha[i](1-\alpha[i])}$$

$\downarrow$

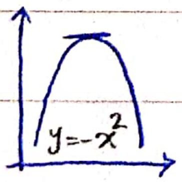
semi negative definite (SND)

$\Downarrow$

concave

$\Downarrow$

unique max



* گرادیان روش درجه 1 است یعنی فقط مشتق درجه اول را حساب می‌کند. جلسه‌ی بعد روش نیوتن برای logistic گفته می‌شود که روش درجه 2 است.

/ /

ML-8

Newton's method (iteratively reweighted least squares)

این جلسه در مورد روش نیوتن و کاربرد آن به عنوان یک روش optimization گفته خواهد شد.
روش نیوتن مشابه gradient descent/ascent، روشی برای optimization است، اما
second order است، یعنی از مشتق اول و دوم هر دو را استفاده می‌کند، درصورتی که در روش
gradient descent/ascent، صرفاً با گرادیان و مشتق اول سروکار داشتیم.
استفاده از مشتق دوم در روش نیوتن سبب می‌شود که سرعت همگرایی بیشتر شود و در خیلی از مسائل
بهینه‌سازی یادگیری ماشین استفاده می‌شود از جمله logistic regression که در اینجا بحث خواهد شد.

Newton's method

$\boxed{f: R \rightarrow R}$ ← مثلاً تابعی مثل f را فرض کنید

The goal: to find x such that $f(x)=0$.

ابتدا به دنبال این هستیم که $f(x)$ کجا صفر می‌شود، بعد با ایجاد تغییراتی می‌توانیم maximization،
یا minimization را حل کنیم.

newton's method
1) iterative
2) second order

updating formula : $\boxed{x := x - \dfrac{f(x)}{f'(x)}}$

ex: find the zero of x^3-8 → $\boxed{x=2}$ analytical

↘ by newton's method → $x := x - \dfrac{x^3-8}{3x^2}$

$x^{(0)}=4$
$x^{(1)}=2.83$
$x^{(2)}=2.22$
$x^{(5)}=2.001$

اما فرمول نیوتن از کجا به دست آمده است (درواقع از تعریف مشتق استفاده شده است):

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

حالا فرض می‌کنیم، x به اندازه‌ی کافی نزدیک x_0 هست و بنابراین lim را حذف می‌کنیم.

$$\Rightarrow f'(x_0) \approx \frac{\overset{0}{\overbrace{f(x)}} - f(x_0)}{x - x_0}$$

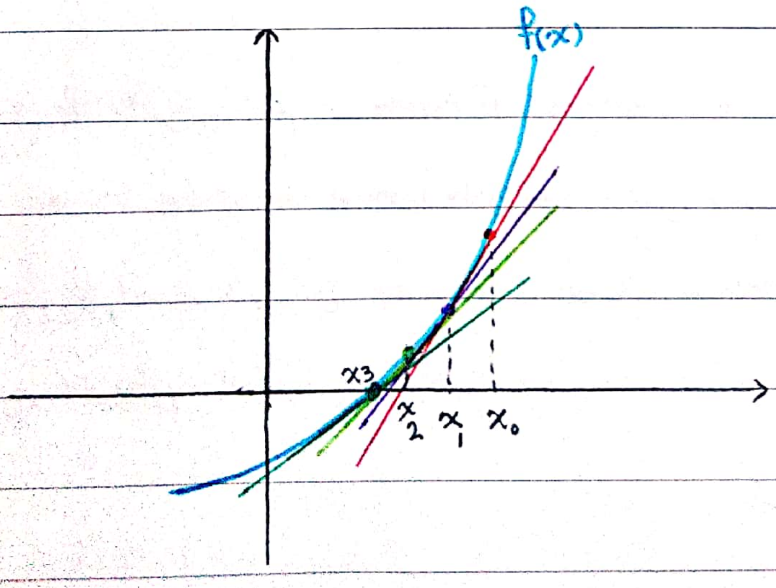
$\rightarrow$ خب حالا دنبال x هستیم که $f(x)=0$ باشد.

$$\Rightarrow f'(x_0) \approx \frac{-f(x_0)}{x - x_0} \rightarrow \boxed{x \approx x_0 - \frac{f(x_0)}{f'(x_0)}}$$

$\leftarrow$ این رابطه همان رابطه‌ی نیوتن است.

$$\boxed{f(x) \approx f(x_0) + f'(x)(x - x_0)}$$

$\rightarrow$ درواقع نقاطی که نزدیک x_0 هستند به عنوان خط مماس هستند.



شکل الگوریتم نیوتن:

How to maximize $\ell(\beta)$ using Newton's method?

اکنون هدف این است که با روش نیوتن likelihood را maximize کنیم.

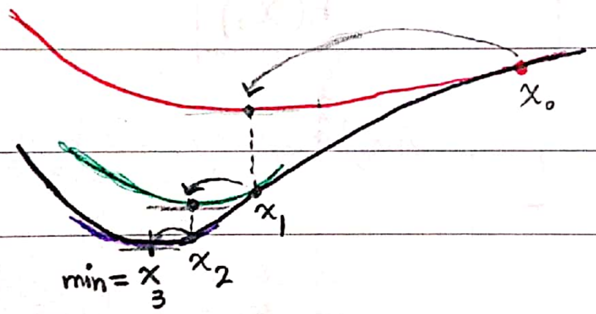
solve: $\ell'(\beta) = 0$

خب در بعضی مواقع مثل linear regression با روش analytical می‌توانستیم حل کنیم ولی مثلاً در logistic regression باید با روش عددی حساب کرد.

$l'(\beta)=0 \qquad (f(x)=l'(\beta))$ (دقتاً همه‌ی بحث روی univariate است)

$$\boxed{\beta := \beta - \frac{l'(\beta)}{l''(\beta)}}$$

در هر نقطه یک تقریب درجه 2 می‌زنیم به جایی که نقطه‌ی x و تقریبی که زده‌ایم به تابع f(x) مماس شوند که این یک تقریب محلی (local) است.



دقت f(x) را تقریب زدیم، min در آن تابع تقریب زده شده را به دست می‌آوریم، min به دست آمده، x را estimation بعدی ما می‌شود و به همین ترتیب ادامه می‌دهیم.

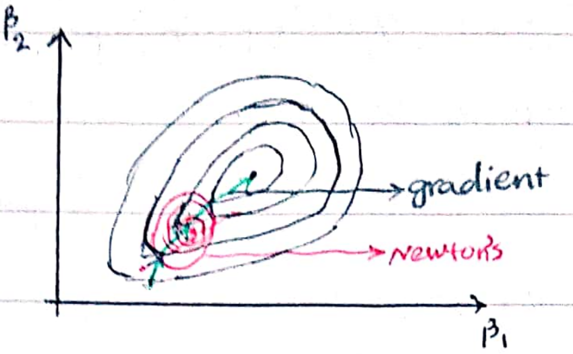
این روش هم شبیه روش‌های gradient حرکت می‌کند تا به نقطه‌ی min برسد، فقط حرکت آن سریع‌تر از روش‌های gradient است.

Multidimensional case:

$$\boxed{\beta := \beta - \underbrace{H^{-1}}_{\text{Hessian}} \nabla l(\beta)}$$

/ /

فرض کنید contour plot برای optimization به شکل زیر است :



Taylor's theorem:

همان x_0

$$\boxed{f(x) \approx \boxed{f(a) + \nabla f(a)(x-a) + \frac{1}{2}(x-a)^T H (x-a)}} \rightarrow q(x)$$

⇓ تقریب درجه 2 برای $f(x)$ که همان بسط تیلور است.

$q(x)$: an approximation of $f(x)$ [around $\underline{a}$]

$$\boxed{q(x) = \frac{1}{2} x^T H x + b^T x + c} \longrightarrow b = \nabla f - Ha$$

c = constant

⇓

$$\nabla q(x) = 0 \longrightarrow \nabla q(x) = Hx + b = 0$$

$$\Rightarrow x = -H^{-1} b = -H^{-1}(\nabla f - Ha) \Rightarrow \boxed{x = a - H^{-1} \nabla f}$$

فرمول به دست آورده، هر دو maximization و minimization را انجام می‌دهد (در دو حالت علامت منفی در فرمول یکسان است)

در روش نیوتن باید چک کنیم که مشتق دوم q ، SND باشد.

$\nabla^2 q \to$ SND (semi negative definite)

چون در تابع‌هایی که هم convex و هم concave است، دنبال هم min و هم max (در دو... مثلاً به دنبال max هستیم، ممکن در min گیر بیفتیم، و اگر به چنین حالتی برسیم و $\nabla^2 q$ از نوع SND نبود آن موقع با gradient مسئله را حل می‌کنیم.

Multidimensional case - 2 :

in statistics : Fisher's scoring algorithm

در آمار به مشتق اول likelihood ، score گویند.

1) حالتی که SND نباشد می‌توان از gradient descent / ascent استفاده کرد.

2) rather calculation H^{-1}, solve $Hy = \nabla f \Rightarrow \boxed{x_{t+1} = x_t - y}$

$$\boxed{x_{t+1} = x_t - H^{-1}\nabla f}$$

از لحاظ ریاضی این دو یکی هستند.

مشکلی که در فرمول نیوتن هست این است که Inverse یک ماتریس ممکن است مشکلات عددی داشته باشد یا زمانبرتر باشد.

مزیت محاسبه‌ی y این است که به لحاظ عددی robust تر است.

Newton's method for logistic regression:

جلسه‌ی قبل logistic regression را با استفاده از MLE حل کردیم و الگوریتم gradient ascent آن را به دست آوردیم. اما مشکلی که این گرادیان دارد، خیلی کند است و همگرایی آن سریع نیست و روش نیوتن خیلی سریع‌تر است.

اگر بخواهیم به روش نیوتن عمل کنیم می‌توانیم Hessian و گرادیان likelihood را بنویسیم.

$$H = -(x^T W x) \qquad W = \begin{bmatrix} w_{11} & & 0 \\ & \ddots & \\ 0 & & w_{nn} \end{bmatrix} \qquad w_{ii} = \alpha[i]\,(1-\alpha[i])$$

$$\alpha[i] = \frac{1}{1+e^{-\beta^T x[i]}}$$

$$\nabla \ell(\beta) = x^T (\underset{\text{response}}{y} - \alpha)$$

$$\Rightarrow \beta^{new} := \beta - H^{-1} \nabla \ell(\beta)$$

$$= \beta + (x^T W x)^{-1} x^T (y-\alpha) =$$

$$= [(x^T W x)^{-1} x^T W x]\beta + (x^T W x)^{-1} x^T W W^{-1} (y-\alpha)$$

$$= (x^T W x)^{-1} x^T W \overbrace{(x\beta + W^{-1}(y-\alpha))}^{Z}$$

$$\beta_{new} = \boxed{(x^T W x)^{-1} x^T W Z} \longrightarrow Z = x\beta + W^{-1}(y-\alpha)$$

adjusted response

این فرمول در واقع یک weighted linear R از $x \to Z$ (نه $x \to y$) است.

درواقع فرمول پیشین برای β_{new} می‌گوید که اگر بخواهیم نیوتن را در R پیاده سازی کنیم می‌توانیم از weighted Lm استفاده کنیم، یعنی برای حل logistic R با روش نیوتن کافیست در هر iteration، یک WLR حل کنیم. پس برای حل تعدادی logistic R وزن دار حل می‌کنیم.

at each step: solve weighted least sqveres.

به روش نیوتن برای logistic R در برخی روش‌های دیگر، ↓ گوییم:

Iteratively reweighted least squares

$$\boxed{\beta^{new} = \arg\min (Z - X\beta)^T W (Z - X\beta)} \rightarrow WLR$$

Regularized logistic Regression:

چون می‌خواهیم min، منفی likelihood می‌نویسیم.

likelihood for logistic R $\longrightarrow$

$$\hat{\beta} = \arg\min_{\beta} - \sum \left(y[i] \log \alpha[i] + (1 - y[i]) \log (1 - \alpha[i]) \right) + \lambda \sum_{j=1}^{p} |\beta_j|$$

* repeated application of weighted LASSO.

Exponential families :

تا اینجا برای حل مسائل مختلف از توزیع های متفاوتی استفاده کردیم مثلاً از توزیع گوسین (نرمال) ، توزیع برنولی و ...

حالا سؤال اینجاست که اگر y ، توزیع دیگری باشد و از دو توزیع نام برده پیروی نکند ، مثلاً count data گسسته است و مثلاً توزیع پواسن مناسب است ، یا مثلاً توزیع بین صفر و یک است ، پس توزیع β مناسب است و یا زمان که همیشه مثبت است و توزیع گاما مناسب است و سایر موارد.

اگر مثلاً y توزیع binomial دارد ← binary classification

اگر " " " گوسین باشد ← Linear Regression

" " " " پواسن " ← puasson Regression

* حال اگر y از نوع دیگری بود ، چگونه مسئله را حل کنیم؟ چگونه Regression را به دست آوریم؟ چگونه پارامترهای آن را تخمین بزنیم؟

پس ابتدا باید با خانواده ی Exponential distribution آشنا شویم.

اگر توزیع های گاما ، بتا ، گواسین ، برنولی و ... جزء این خانواده هستند و اگر توزیعی جزء این خانواده باشد به راحتی می توان parameter estimation انجام داد.

Def: An exponential family is a set of distributions with 4 components.

$$\boxed{P_\theta(x) = \frac{1}{Z(\theta)} h(x) \exp\left(t(\theta)^T s(\theta)\right)}$$

اگر توزیعی را بتوانیم به این صورت بنویسیم جزء خانواده ی exponential محسوب می شود.

/ /

$\theta \in R^P$

$\theta = (\theta_1, \dots, \theta_p)$

$x \in R^d$ x چیز متغیر است.

$Z(\theta) : \theta \to [0,\infty]$ → partion function

$h : R^d \to [0,\infty]$

چون توزیع احتمال باید مثبت باشد.

این دوتا وکتور هستند.

$s_i : R^d \to R$ sufficient statistic

$t_i : \theta \to R$

ex1: Exponential distribution

$x \sim Exp(\lambda) \quad \lambda > 0$

$$P_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & x \geqslant 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{or} \quad P_\lambda(x) = \lambda e^{-\lambda x} I(x \geqslant 0)$$

$\Downarrow$

$$Z(\lambda) = \frac{1}{\lambda}, \; h(x) = I(x \geqslant 0), \; t(\lambda) = +\lambda, \; s(x) = -x$$

ex2: Bernoulli distribution

$$P_\theta(y) = \theta^y (1-\theta)^{1-y} \qquad y \in \{0,1\}$$

$\downarrow$

solve1)

$$= \boxed{\exp\left(y \log\theta + (1-y)\log(1-\theta)\right)}$$

$$h(y) = I(y \in \{0,1\}), \; Z(\theta) = 1, \; s(y) = [y \quad 1-y]$$

$$t(\theta) = [\log\theta \quad \log(1-\theta)]$$

از این فاکتور می‌گیریم در نهان

solve 2) $P_\theta(y) = \exp\left(y\left(\log\theta - \log(1-\theta)\right) + \log(1-\theta)\right)$

$$= \exp\left(\log(1-\theta)\right)\exp\left(y\log\frac{\theta}{1-\theta}\right) = \boxed{(1-\theta)\exp\left(y\log\frac{\theta}{1-\theta}\right)}$$

$$Z(\theta) = \frac{1}{1-\theta}, \quad \boxed{S(y) = y}, \quad t(\theta) = \log\frac{\theta}{1-\theta}$$

فرم‌هایی که در آن $S(y)=y$ است، natural نامیده می‌شود و در کاربرد بهتر است
چون محاسبه‌ی آنها راحت‌تر است.

---

ex 3 : Gussian distribution

$\boxed{\sigma^2 = 1}$

$$P(y) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}(y-\mu)^2\right)$$

$$= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}y^2 + \frac{2\mu}{2}y - \frac{\mu^2}{2}\right)$$

* قسمت‌هایی که فقط به پارامتر وابسته است را بیرون می‌آوریم.

$$= \boxed{\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\mu^2}{2}\right)\exp\left(-\frac{1}{2}y^2\right)\exp(\mu y)}$$

$$\Rightarrow Z(\mu) = \sqrt{2\pi}\exp\left(\frac{\mu^2}{2}\right), \quad h(y) = \exp\left(-\frac{1}{2}y^2\right), \quad t(\mu) = \mu, \quad S(y) = y$$