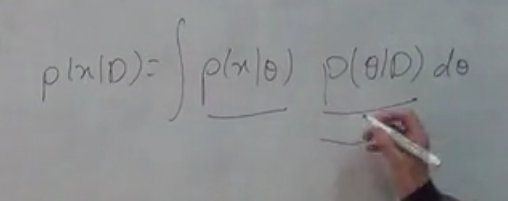
جلسه سوم:

در دیدگاه ML به این سمت میرفتیم که فقط به آزمایش ها نگاه میکردیم در دیدگاه MAP پارامتر ها را به صورت متغیر تصادفی میدیدم. در شروع یک توزیعی روی داده ها در نظر میگرفتیم بر اساس دانش پیشین داده ها مثلا میدانستیم میانگین چه قدر است تا بتوانیم تخمین خوبی برای پارامتر یا تتا بدست بیاوریم. بعد از آن بر اساس همین توزیع را ضربدر likelihood میکردیم و میتوانستیم برای کلاس فعلی احتمال داده ها را محاسبه بکنیم. و میبینیم که هر چه قدر داده ها بیشتر میشود واریانس هم کم میشود، دیدیم که دانش پیشین که شکل گوسی دارد در تک به تک likelihood ها ضرب شود حاصل هم گوسی خواهد بود با میانگین داده شده. در واقع بعد از دیدن داده ها ما تدریجی اعتقادمون به یک پارامتر یا یک کلاس بیشتر میشود و یعنی دیگر واریانسی نداریم و واریانس کمتر میشود و انگار روی مقدار پارامتر یا کلاس دارد peak میزند و واریانس به شدت کم میشود، در 2 دیدگاه قبلی یکی به نمونه های آزمایش نگاه میکرد و likelihood را حداکثر میکردیم و دیدگاه دوم به دنبال مقداری بودیم از پارامتر که بیز را حداکثر بکند یعنی بعد از دیدن داده ها به کدوم کلاس و پارامتر بیشتر از همه معتقد هستیم همان را برمیگردانیم . در دیدگاه سوم که fully baisien خواهد بود، تا از اطلاعات کامل بیز استفاده بکنیم نه صرفا دوباره بیشینه پارامتر بگردیم، در این دیدگاه ما مقدار پارامتر یا کلاس را فیکس در نظر نمیگیریم و میگیم بعد از دیدن داده ها تمام پارامتر ها را در خروجی دخیل میکند برای هر کدام یک توزیع گوسی میکشیم و به هر کدام چه مقدار اهمیت میدهیم و به اون پارامتر بعد از دیدن گوسی ها چه قدر اعتقاد داریم؟ اون فرمول انتگرال. یعنی مثلا میبینیم چون مقدار پارامتر یک توزیع گوسی کم هست پس کم بهش اعتقاد داریم و به اون توزیع گوسی مقدار کمتری خواهیم داد. و اگر توزیع گوسی دقیقا زیر MAP باشد وزن بیشتری بهش میدهیم:

 راستی مقدار وزن هست که بعد از دیدن داده ها به این مقدار پارامتر چه قدر معتقد هستیم و سمت چپی توزیع با اون مقدار پارامتر هست و به جای اینکه ما فقط یک توزیع در بیاوریم با یک پارامتر مشخص، همه را در کنار هم میگذاریم و یک توزیع کامل بدست میاوریم در واقع از همه پارامتر ها استفاده میکنیم به جای استفاده از یک پارامتر و دانش پیشین ما.

چه موقع جواب دیدگاه سوم با دوم یکی است؟ واریانس هر دو خیلی کم باشد و به شکل مثلث شود یعنی MAP فقط یک جا مقدار دارد برای اون توزیع و بقیه جاها صفر است. انگار در فرمول انتگرال ضریب همه توزیع ها صفر میشود جز اونکه MAP هست ضریب آن صفر نیست و MAP ما در دیدگاه دوم انگار peak زده است.

اگر تعداد داده ها زیاد شود چون واریانس posterior کم میشود جواب این 3 دیدگاه یکسان میشود. بنابراین چون تعداد داده ها کم است ما مشکل بیش پردازش داریم که سعی میکنیم از روش دوم یا MAP استفاده بکنی با دانش پیشین و مشاهده داده ها جلوی بیش پردازش را بگیریم. کلا در این درس با ML و MAP سروکار داریم.

Regression: برای وقتی است که داده پیوسته است. دوست داشتیم که f(x) ما تخمین خوبی برای y واقعی باشد. با استفاده از الگوریتم یادگیری میخواستیم فرضیه مناسبی را از فضای فرضیه برداریم بر اساس داده ها آموزشی جدید خروجی را حدس بزنیم. برای رگرشن خطی همه توابع خطی را در نظر میگیریم و الگوریتم یادگیری باید از بین توابع خطی یکی را پیدا کند و در خروجی به ما بدهد که تا حد ممکن با اون y که ما شناخت نداریم و صرفا از رو داده ها میشناسیم را تخمین بزند.

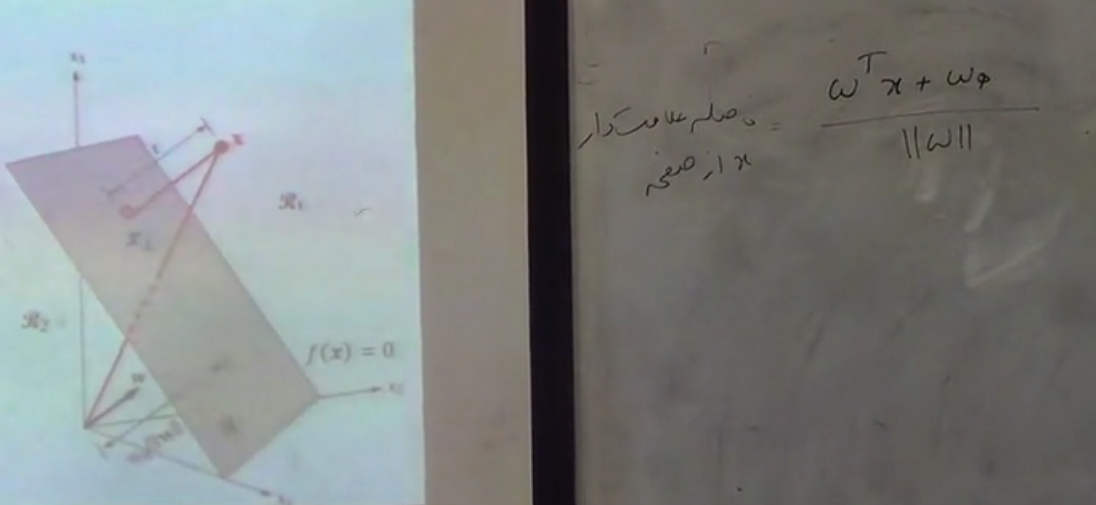
در مسئله بهینه سازی میخواهیم پارامتری را انتخاب بکنیم که فاصله تخمین ما را با هدف ناشناخته نزدیک بشود. فضای فرضیه یک سری مقادیر مختلف پارامتر هستند برای w0 تا wd که تا تقریب خوبی برای هدف بدست بیاوریم هدف یا target.

اول یک معیار خطا باید مشخص کنیم که خروجی ما را با هدف مقایسه کند ببیند چه قدر فاصله دارد.

کلاس بند خطی:

میخواهیم برای هر کلاس یک تابع بگیریم مثلا با f(x) جاهایی که بزرگتر باشد کلاس مثبت و جاهایی که منفی باشد کلاس دیگری باشد. در واقع دنبال این هستیم که مرز خطی پیدا کنیم.

مرز تصمیم دو کلاس جایی میشود که این صفر هست و یک ابر صفحه میشود و داده های 2 کلاس جدا میشود. و w بردار نرمال اون ابر صفحه خواهد بود.



فرمول پایین این صفحه را دوباره بخوان .

هر چه فاصله نقطه تا صفحه زیاد تر شد یعنی اون نقطه دور تر از صفحه است. تا دقیقه 40.

جلسه دهم-SVM:

چرا margin زیاد بهتر است؟ بخاطر تعمیم پذیری چون اگر واریانس زیاد باشد میتوانیم با حاشیه زیاد تعمیم پذیری ایجاد کنیم.

اگر معادله ابر صفحه را در یک ضریب ثابت ضرب کنیم اتفاقی واسه ابر صفحه نمیفتد و نقاط باز هم روی ابر صفحه خواهند بود.

X(n) نزدیک ترین نقطه به ابر صفحه است. w جوری این را scale میکند که قدر مطلق فاصله اون 1 خواهد بشود حتی اگر 30 باشد باز جوری scale میشود که به 1 برسیم.

چرا 2 برابر کردیم؟ چون خط از وسط margin رد شده است پس 2 برابر آن را باید max کنیم. اگر نزدیک ترین نقطه را در معادله ابر صفحه بگذاریم مقدارش 1 میشود. چون min را بهینه سازی آن را بلد نیستیم حل بکنیم باید min را حذف کنیم.

چرا قدر مطلق برابر میشود با خروجی ضربدر معادله ابر صفحه؟ چون اگر مثبت باشد داده ما هم اون برچسب مثبت هست هم معادله ما در قسمت مثبت اگر منفی باشد که منفی و حاصل مثبت میشود پس انگار کار قدر مطلق را دارد انجام میدهد فرض این است داده با خط جدا میشوند.

دقت کن اگر یک تابع بخواهی حداکثر کنی اگر مثبت باشد میتوانیم توان 2 را حداکثر کنیم پس اون توان 2 روی norm به این شکل بدست آمده است. برای min کردن هم به جای اینکه خود تابع را حداکثر کنیم میتوانیم 1/f را حداقل کنیم.

دقت کن label داده که مثلا مثبت است مستطر است در فرمول ابر صحفه چون فرض کردیم درست داده ها دسته بندی شده اند و وزن کلاس مثبت واقعا مثبت خواهد بود.

2 تا هدف داریم هم داده ها همه درست دسته بندی شوند هم margin را حداکثر کنیم. اینکه همه داده ها درست دسته بندی شوند یعنی فرمول ابر صفحه با علامت label واقعی داده باید یکی شوند.

شکل تابع SVM محدب است. و میتوانیم بهینه سازی آن را حل کنیم.

بهینه سازی quadratic درجه 2 برحسب x هست و یک درجه 1 دارد و محدودیت ها شکل خطی دارد چه تساوی چه نا مساوی.

مسئله بهینه سازی را چطوری حل میکنیم با قیود؟ اگر بخواهیم اون لاگرانژ رو بیشینه کنیم هم باید علامت لامبدا با h رو یکی کنیم هم تا حد ممکن h را زیاد کنیم تا به بینهایت برسیم.

اما اگر شرط برقرار باشد h که صفر است هیچی ولی مقدار gها که منفی است الفا باید صفر باشد تا منفی ها از بین بروند و مقدار بیشینه شود. دقت کن اونجایی که شرط برقرار هست که میشود f(x) اونجایی که برقرار نیست میشود بینهایت.

Max, min نسبت به هم خاصیت جابجایی ندارند.

**درخت:**

ما مسائلی داریم که دیگر به دنبال پارامتر ها و بهینه سازی آنها نیستیم و بر اساس دسته بندی نمیکنیم و به دنبال مثلا تقسیم بندی فضا هست که در هر فضا چه برچسب ی را به آن ناحیه بدهد.

از ترکیب درخت و bagging یادگیری تجمعی که مدل قوی تری هست را میسازیم.

درخت ها: داده ها به صورت پیش فرض در واقع ویژگی های ما گسسته هستند مثلا سن بالای 40 سال هست یا نیست. اینکه ویژگی های ما گسسته باشند یا categorical باشند مناسب تر است.

Node ها همان attribute ها یا feature های ما هستند و یال هایی که ازش خارج میشود به تعداد value هایی که میگیرد میتواند یال داشته باشد. در برگ ها دیگر attributeیی نداریم و صرفا برچسب اختصاص میدهیم. انگار فضا را دارد تقسیم بندی میکند و برچسب میدهد تقسیم بندی هم یا بر اساس یک ویژگی هست یا بر اساس and یک سری ویژگی که هر مسیر از ریشه تا برگ هست حالا یا خودش یا not اون attribute ظاهر شده است. چند تا ناحیه که کنار همدیگر قرار میدهیم مثلا میشوند برچسب مثبت.

درخت تصمیم رویکرد حریصانه دارد و ویژگی که بهترین جدا سازی را میکند صرفا انتخاب میکند.

مسائل که مقادیر گسسته دارند یا فضای گسسته داریم و دنبال درخت بهینه هستیم از جنس np-complete هستند. یعنی از بین درخت های مختلف با رویکرد حریصانه سعی میکند درخت را بسازد.

ID3: چجوری ویژگی بهتر را انتخاب بکنیم؟ معیار باید تابعی از چه چیزی باشد؟ معیار information gain روی یک مجموعه نمونه با یک attribute اعمال میشود.

کی تموم میکند کار را و بازگشتی صدا نمیکند؟ همه نمونه هایی که در node فعلی داریم یک ویژگی داشته باشیم یعنی مثلا همه + باشند. یا همه attribute ها را استفاده کنیم ولی همه داده ها برچسب یکسان پیدا نکنند خاتمه پیدا میکنیم. در این شرایط که اگر ویژگی ها را استفاده کردیم و همه یک برچسب نگرفتند باید رای گیری کنیم در اون برگ و به بیشترین برچسب را اختصاص میدهیم.

خود بازگشتی: تو مجموعه میگردیم بهترین attribute را برمیداریم بعد تفکیک میکنیم بعد مسئله را حالا برای اون یال ها حل میکند.

معیار چی هست؟ میخواهیم attribute خاصیتی داشته باشد که وقتی بر اساس اون درخت را تشکیل میدهیم دوست نداریم خلوص هر زیر درخت پایین باشد یعنی دوست داریم همه از یک برچسب باشند که برای این خلوص از معیار entropy استفاده میکنیم، به نوعی expectation منهای log p(x) را حساب میکند. وقتی مقدار نیم باشد entropy حداکثر مقدار خودش را دارد و وقتی خروجی h x صفر یا یک باشد کمترین مقدار خودش را دارد. یعنی هر چی که در مورد احتمال برچسب ها اطلاعات کمتر داشته باشیم و نظم کمتری وجود داشته باشد entropy بیشترین مقدار و اگر مطمئن باشیم همه از یک برچسب هستیم entropy کمترین مقدار را دارد.

یعنی دنبال attribute هایی هستیم که اگر بر اساس اون بشکنیم بی نظمی کمینه میشود.

تعداد node کمتر به تعمیم پذیری مرتبط است یعنی داریم به فرضیه های کوچک تر است اولویت میدهیم در نتیجه واریانس آنها کم میشود.

معیار information gain: از آنتروپی استفاده میکند. محور افقی پارامتر توزیع برنولی است و محور عمودی مقدار entropy را نشان میدهد.

در جمع وزن دار میگوید نسبت sample هایی که به اون نقطه میرسند ضربدر entropy آن نقطه یعنی میخواهد ببیند بعد که این attribute را در نود گذاشت مقدار entropy هر کدام از فرزندان چه قدر است، و بعد جمع وزن دار آنها با تعداد آنها را در نظر بگیرد. سمت چپ منها entropy قبل از دیدن این ویژگی است و سمت راست entropy بعد از دیدن این ویژگی است.

یعنی این ویژگی را ببینم چه قدر از entropy کم میشود. هر چه قدر بیشتر کم شود اون ویژگی موثر تر است و بی نظمی از بین میرود. الگوریتم حریصانه میرویم.

یک تابع دیگر در کنار information gain، gini است که میگوید p\*1-p اگر p احتمال + شدن برچسب باشد که یک تابع درجه 2 هست و روی نیم ماکسیموم میشود روی صفر و یک مقدار صفر است و روی نیم مقدارش ¼ است و ماکسیموم خودش است این هم یک معیار برای نشان دادن عدم یکنواختی در برچسب ها وجود دارد.

Entropy یک یعنی مثلا اگر 2 تا کلاس داشتیم از هر کلاس به صورت 50، 50 نمونه وجود دارد در این شاخه.

تو فرمول entropy یا information gain: سمت چپ همیشه entropy پدر است. سمت راست منهای سیگما تعداد نمونه ها در هر شاخه ضربدر entropy اون شاخه.

اونی را انتخاب میکنیم که gain بیشتری دارد.

Mutal information: h(y)- h(y|x) یعنی اینکه entropy y منهای وقتی که x دیده شده است بکنیم. یعنی کاهش مقدار entropy y بعد از دیده شدن مقدار x.

دقت کن information gain منهای سمت چپ و راست دارد که سمت چپ میشود پدر خود entropy میشود p\*logp یک سیگمای منفی پشتش.

در mutal information: وقتی حداقل مقدارش میشود که در log دو تا متغیر از همدیگر مستقل باشند که 1 بشوند و log 1 میشود 0. به عبارت دیگه یعنی دنبال ویژگی برویم که یعنی ویژگی از برچسب مستقل نیست و اگر بر اساس اون ویژگی بشکنیم انگار بر اساس اون برچسب هم شکستیم. Mutual information زیاد یعنی از برچسب مستقل نیست و یک جوری مقدار آن دارد مقدار برچسب را میگوید. پس اون mutual information هایی که از برچسب مستقل هستند info خاصی برای ما ندارند.

پس در کل دنبال ویژگی هستیم که condition entropy را حداکثر میکند.

در جاهایی که میخواهیم گسترش دهیم جمله اول information gain با هم یکسان نیست و باید بر اساس هر کدام نود محاسبه کنیم و ببینیم کدوم بیشتر است از نظر gain.

ولی اگر در یک نود هستیم و میخواهیم بدانیم در این نود کدام ویژگی را قرار بدهیم فرقی نمیکند که info gain را حداکثر کنیم یا conditional entropy را حداقل بکنیم.

خصوصیات الگوریتم ID3: اگر مجموعه داده سازگار داشتیم یعنی اگر دو تا نمونه داشتیم که شباهت زیادی با هم داشتند دو تا برچسب مختلف نداشته باشند و یکی باشند از لحاظ برچسبی، یعنی در یک نقطه 2 تا برچسب نداشته باشیم یک برچسب باشد. در این شرایط همیشه مقدار خطای آموزشی آن 0 است.

انتخاب ویژگی به صورت حریصانه بود. بر اساس info gain انتظار داریم سایز درخت کوچک باشد چون انتظار داریم در یک سری از شاخه ها سریع به یکنواختی برسیم و دیگه درخت گسترش پیدا نکند. این امر سبب تعمیم پذیری بهتر میشود چرا چون فرض کن فضای فرضیه ما خطای آموزشی همه آنها صفر میشد و میتوانستیم همه را گزارش کنیم اما مشکل چی بود؟ با عوض کردن مجموعه آموزشی فرضیه های مختلفی انتخاب میشد و واریانس بین فرضیه ها زیاد میشد و تعمیم پذیری میومد پایین. یعنی اولویت میدهیم به درخت های کوچک تر و واریانس کم میشود و تعمیم پذیری بالا تر میرود. ولی اینقدر خوب نیست چون اینقدر جلو میریم که همه در هر برگ یک برچسب بشوند از نمونه های باقی مانده و این باز باعث میشود درخت پیچیده شود. زیرا حتی اگر از 10 تا داده یک برچسب غیر یکنواخت هم دیده باشیم باید دوباره بشکنیم پس اینکه بخاطر یکنواخت شدن برچسب ها هعی بشکنیم مناسب نیست و درخت پیچیده تر میشود یعنی درست است از info gain استفاده کردیم اما شرط خاتمه بدی داریم.

دقت کن id3 کلا دنبال این است برود تو یک زیر فضا و برچسب آنها را یکی گزارش کند و بقیه ها برچسب متفاوت باشند و اینجوری حساس میشود صرفا به جاهایی که در مجموعه آموزشی دیده مثبت بودند به اون نواحی مثبت اتلاق میکند.

فضای فرضیه میشوند and یک سری ویژگی از ریشه تا برگ داریم یعنی مسیر از ریشه تا برگ را نشان میدهد. که بین آنها or است که نشان گر برگ های مثبت مختلف است. یعنی اگر میخواهی برچسب این مثبت شود از این اینها یکی را میتوانی انتخاب بکنی نگاه توابع Boolean داشته باش. پس وقتی ما فقط صرفا بر اساس value ویژگی ها بخواهیم فضا را دسته بندی کنیم فضای ما میشود یک سری نواحی از ابر مکعب ها که اونجا مثبت هستند و بقیه جاها منفی و مثبت میشود اجتماع این فضا ها.

اگر بدون هیچکاری الگوریتم id3 را اجرا کنیم ممکن است منجر به بیش پردازش شود. چرا چون بخاطر یک نمونه که برچسب یکنواخت نیست دارد دوباره میشکند و پیچیدگی زیاد میشود.

2 راه وجود دارد برای بیش پردازش: یک زود متوقف شویم و دیگه اگه gain بیشتر نبود شکسته نکنیم. دو اینکه تا آخر درخت را بسازیم بعد هرس کنیم.

معیار هرس: یک زیر درخت را فرض کن یا میتوانیم بشکنیم به زیر درخت های دیگه دو اینکه میتوانیم نشکنیم و تبدیل به برگ کنیم. تبدیل به برگ میکنیم و خطا روی validation را بدست میاوریم. بار بعدی زیر درخت را میشکنیم به زیر درخت های دیگه و validation را میسنجیم اگر خطای اولی که برگ بود کمتر شد دیگه تبدیل به برگ میکنیم و سودی ندارد که بشکنیم. در موقع حذف بین زیر درخت های مختلف یا نود های مختلف نگاه میکند اونی که کمترین error validation را دارد اگر تبدیل به برگ شود.

رویکرد توقف زود هنگام درخت را کامل نمیسازد.

الگوریتم c4.5: درخت تصمیم را به صورت یک مجموعه rule در میاورد یعنی and یک سری لیترال و or بین آنها.

در این الگوریتم هرس کردن را بر اساس node ها انجام نمیدهد حذف کردن را بر اساس حذف اون لیترال ها انجام میدهد. از یک rule حذف میکند. یعنی a2 تو یک rule باشد تو یک rule نباشد و باعث میشد یکی از یک جا حذف شود و از دیگری حذف نشود اگر بر اساس rule ها باشیم صرفا، همچنین در الگوریتم قبلی اگر نود بالا را حذف میکردیم از همه rule ها حذف میشد اما اینجا ترتیبی در rule ها نداریم و از هر ruleیی که بخواهیم میتوانیم حذف کنیم.

نکته: در c4.5 ما همیشه E T را همیشه بر اساس target محاسبه میکنیم بر اساس کلاس هدف بله و خیر نه بر اساس اون چیزی که split کردیم بر اساس آن. در محاسبه entropy کلی یا t اصلا کاری به ویژگی که به عنوان نود انتخاب کردیم نداریم. یعنی بعد از اینکه تبدیل به rule شد همه هم اولویت میشوند و postprocess است یعنی بر اساس نتیجه پیش پردازش میکنیم بعد از اجرا شدن ID3. دیگر درخت نیست اصلا یک سری rule است.

ویژگی های پیوسته: اگر ویژگی پیوسته بودند چی یعنی تا الان فکر میکردیم تعداد محدودی هر ویژگی مقدار دارد یا 2 تا مقدار دارد. یک راه حل این است که یک آستانه تعریف کنیم اگر بیشتر بود برن راست اگر کمتر بود برن سمت چپ. اگر بخواهیم برای اون آستانه که تعریف میکنیم بشکنیم ثابت میشود که اگر بخواهیم بین هم برچسب ها بشکنیم در اونجا ها info gain هیچوقت بیشینه نمیشود، نسبت به مرز ها که جایی که برچسب دارد عوض میشود بیشینه info gain را داریم. چون تا مرز که بیایم همش یکنواختی است و همه را در یک سمت قرار میدهیم. در گسسته شرط ها روی هر attribute یک شرط بود ولی الان برای یک attribute شرط های مختلفی داریم یعنی آستانه های مختلف داریم و info gain را حساب میکنیم و بیشینه را نسبت به آستانه های مختلف و نسبت به بقیه ویژگی های مختلف بررسی و انتخاب میکنیم و دقت کن بر خلاف گسسته که در هر زیر درخت یکبار میتوانستیم از یک ویژگی استفاده کنیم اینجا چون آستانه های مختلف داریم در یک زیر درخت بیش از 1 بار میتواند یک ویژگی ظاهر شود با آستانه های مختلف.

معیار mutual information سعی دارد ویژگی های را در اول انتخاب کند که مجموعه value بیشتری دارد یعنی اونکه 10 تا value دارد اولویت بیشتری دارد نسبت به اون ویژگی که 2 تا value دارد. بخاطر یکنواختی اما این باعث میشود که درخت ما پیچیده شود و با شرط ما ناسازگار است چه کار باید بکنیم؟

دسته بند خوبی نیست و بیش پردازش میشود.

**روش instance-based learning:**

جز روش های بدون پارامتر هستند.

یک توزیع پارامتری یک بعدی داشتیم و دنبال پیدا کردن پارامتر های آن بودیم قبلا. ما مستقیم از داده آموزشی استفاده میکنیم.

این توزیعی که تخمین میزنیم به صورت بدون پارامتر هم برای generative و هم discriminative قابل حل هستند .فقط نیاز داری جای داده های آموزشی را نگه داریم. فقط همون ها را ذخیره میکنیم.

4 5 تا اسلاید اول را بخوان.

H عرض یک bean است چرا تقسیم بر h؟ چون جمع p b ها برابر با 1 است و عرض پنجره h هست و دنبال این هستیم مساحت زیر آن را برابر 1 نگه داریم تقسیم بر h میکنیم.

فرض کنید یک ناحیه r داریم احتمال اینکه نمونه داخل ناحیه r بیفتد انتگرال احتمال اون داده آموزشی است.

با توزیع احتمال بایونومیال داده ها در ناحیه r قرار میگرفتند.

Expectation k یا متوسط برابر این هست که تعداد کل داده ها ضربدر احتمال این ناحیه ها. وقتی داده ها به سمت بینهایت برود واریانس به سمت صفر میل میکند پس میتوانیم مقدار k را برابر با np بگیریم و p را تخمین بزنیم. پس احتمال افتادن سمپل داخل این ناحیه اگر k تا قراره بیفتند و n تا داده داریم میشوند k/n احتمال افتادن این سمپل داخل این ناحیه. دنبال بدست آوردن تخمین هستیم که kn/n بود کی دقیق بود؟ وقتی تعداد سمپل هایی که در این ناحیه بیفتند زیاد باشند اگر کم باشند که چیزی بدست نمیاد پس باید به بینهایت برود مقدار همسایگی هم باید به صفر میل بکند.

2 راه حل داریم برای تخمین:

ناحیه حول هر نقطه را خودمان باید تنظیم کنیم یا این ناحیه را اینقدر بزرگ میکنیم که k تا نقطه ثابت توی آن بیفتد و یا ناحیه را فیکس در نظر میگیریم میشمریم ببینیم چند تا نقطه توش هست.

راه اول همون knn است و راه حل دوم همان parzen window است یعنی V را ثابت در نظر میگیریم میخواهیم ببینیم چند تا نقطه داخل این همسایگی وجود دارند. وقتی گوسی میگیریم دیگر لزوما فقط افتادن در ناحیه مهم نیست بلکه فاصله اینها و میزان تاثیری که رو داده فعلی دارند هم مهم میشود و یک وزنی میگیرند.

پنجره کوچک واریانس را زیاد میکند چون تا جای داده آموزشی تغییر کند این هم تغییر میکند. اگر پنجره بزرگ شود بایاس زیاد میشود و underfit میشویم و یک سری الگو از دست ما میرود.

سایز همسایگی در knn به چگالی بستگی دارد.

در روش بدون پارامتر چون کل مجموعه داده را داریم با تغییر آن تابع ما عوض میشود.

نگاه generative یعنی دقیقا میخواد تابع چگالی احتمال را تخمین بزند.

روش discriminative روشی بود که بیز میزدیم اون postrior که بیشترین بود کلاس را بهش اتلاق میکردیم.

روش generative تابع چگالی احتمال را تخمین میزند. Knn اینقدر ناحیه را بزرگ میکند تا k تا نقطه در آن بیفتد. محاسباتی سنگین دارد knn چون به ازای هر داده باید فاصله را تا همه نقاط محاسبه کنیم. و همچنین مشکل curse of dimensionality که باعث میشود مرز تصمیم گیری ما خیلی پیچیده شود.

وقتی تعداد نمونه ها به سمت بینهایت برود خطای 1nn یک چیزی بین خطای بیز و .. است، خطای بیز وقتی توزیع داده ها را میدانستیم کمترین خطای ممکن بود یعنی تابع چگالی احتمال اون 2 تا کلاس یک همپوشانی با هم داشت که اون همپوشانی را هیچ موقع نمیتوانستیم از بین ببریم. خطای 1nn از این بیشتر است اما کمتر از 2 برابر خطای بیز است بعلاوه یک bound. وقتی k را زیاد بکنیم به همان خطای بیز نزدیک میشود برای وقتی که نمونه ها به سمت بینهایت میرود.

همسایگی وزن دار knn: یعنی همسایه ها وزن یکسانی نداشته باشند مثلا نزدیک تر ها وزن بیشتری داشته باشند و اثر بیشتری داشته باشند. بر اساس این میگویند دیگه k تا در نظر نگیر روی کل داده ها در نظر بگیر چون به نزدیک تر ها وزن بیشتری میدهی دیگه.



در روش های بدون پارامتر جفت g و d پیچیده هستند و تاثیر خاصی ندارند.

Knn برای رگرشن:

اولین کار ساده این بود k تا همسایه در نظر بگیریم بعد میانگین بگیریم. یک راه دیگر میانگین وزن دار بود.

SVM با وجود اینکه روش با پارامتر است اما در یک زمینه شبیه به بدون پارامتر هست چون اون support vector هاش را نیاز دارد حتی از بعد مرحله یادگیری برای بهبود اون مرز پس به داده های آموزشی نیاز دارد برای انتساب برچسب جدید.