**SVM ALG:**

potrebujeme spravit decision boundary, čiaru, ktorá oddeluje output 1 alebo 0 (bin. klas.)

idea SVMiek je nájsť the **widest street approach** that sepparates the positive samples from negative samples **(4:30 vid ilus.)**

Existuje nejaký vektor **W**, ktory je kolmy k tejto rozhodujucej čiare

Mame daný neznámy bod **X**(sample input) a vektor **U** ktorý smeruje k nemu (zo zac. sur. sys.)

chceme zistit ci X je na lavej alebo pravej strane rozhodujucej priamky

zistime to tak, ze projektneme vektor U k W (tým sa zistí vzdialenosť vektora U vzhľadom na vektor W), ak tá vzdialenosť presiahne urcitu hranicu (**B**), potom mozme povedat, ze je sample pozitivny

- toto cele sa da zapisat jednou rovnicou

skalarny sucin - projekcia - nepoznáme veľkosť vektora **W** kedže je to priamka, vieme len, ze jej smer je kolmy k rozhodujucej ciare.., takto mi vsak ta rovnica nepride uplna ale.. asi lebo nepozname ten vektor W, inak by k tomu bolo aj / ||W||

**(1. DECISION RULE)**

| B - vzdialenosť/posun po vektore W

W\*U -+ B >= 0

B - urcuje vzdialenost/posun po vektore W

W - urcuje smer vektora

U - vektor atributov neznamej vzorky

mame teda 2 nezname premenne, ake B a W pouzit, vieme len ze vektor W je normalizovany k rozhodujucej priamke, takych je tu vsak nekonecne vela kedže môže mať akúkoľvek velkosť

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

potrebujeme navyse nejake constraints pomocou ktorych bude mozne vyp. W a B

ak vezmeme W\*X\_+(positive sample) + b >= 1 ~~(alebo nejake zvolene delta/2)~~ delta = sirka celej cesty

ak vezmeme W\*X\_-(negative sample) + b <= -1~~(delta/2)~~

bolo by vhodne spravit iba jednu vseobecnu rovnicu pre obe pripady

predpoklad: var Y\_i such that Y\_i = +1 for (+) samples, -1 for (-) samples

pre kazdy sampel mame premennu Y\_i, je urcena tym, ci sa jedna o + alebo - sampel

spravime si ohranicenie, ak mam nejaky natrenovany sampel, musi nadobudat cislo vacsie alebo rovne jednej, takze mame

yi(Xi\*W + b) >= 1

yi(Xi\*W + b) <= -1 (vynasobime nerovnicu -1 a sipky sa otocia).. rovnice su rovnake!

yi(Xi\*W + b) >= 1

mozme teda povedat ze Yi(Xi\*W+b)-1 >= 0 pre Yi ktore je pre (-) sample -1, pre (+) sample +1

**(2. ROVNICA GUTTEROV)**

a tym aj **Yi(Xi\*W+b)-1 = 0** for Xi in gutter (na oddelovacej priamke s pridanou vzdialenostou delta [koniec cesty ulice na oboch stranach, na oboch stranach preto, lebo Y podmienka])

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

co je cielom tohto celeho vlastne? Potrebujeme najst co najvacsiu vzdialenost dvoch koncov ulice

mali by sme zistit vyjadrenie tejto vzdialenosti medzi tymito dvoma koncami ulice

X\_+(vektor od poč.súr k bodu X\_+) - X\_-(vektor od poč.súr k bodu X\_-) = je vektor, ktory vyjadruje rozdiel medzi tymito dvoma vektormi,

ked chceme zistit jeho vzdialenost vzhľadom na vektor W a spravíme k nemu projekciu, dostaneme tak vzdialenosť ktorá určuje šírku cesty

WIDTH = (X(+) - X(-)) \* **(W**(priemet, projekcia na W) **/** **||W||**(velkost vektora W)

**pouzijeme predchadzajucu rovnicu 2.**

X(+)\*W + b - 1 -> X(+)\*W = 1 - b

X(-)\*W - b - 1 -> X(+)\*W = 1 + b

therefore

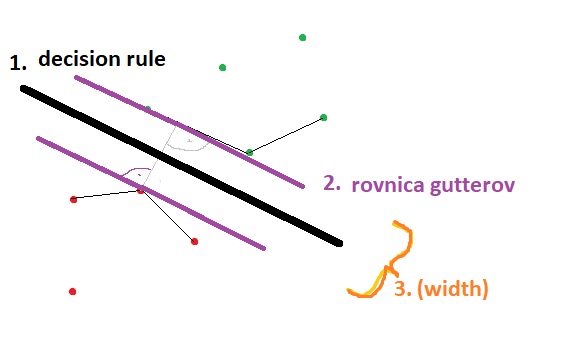
**WIDTH =** 2 / ||W||

i decided i was going to enforce this constraint. I noted that the width of the street has got to be this difference vector times a unit vector. Then I used the constraint to plug back some values here. And I discovered to my delight and amazement that the width of the street is 2 over the magnitude of W

a my chceme maximalizovat tento width max 2/||W||, to znamena ze je to ok maximalizovat 1/||W|| pretoze len upustime konstantu 2, a nasledne je mozne **minimalizovat ||W||** kvoli inverzii

1/2 \* ||W||^2 (for mathematical convenience)

**(3. MINIMIZE WIDTH OF THE STREET)**

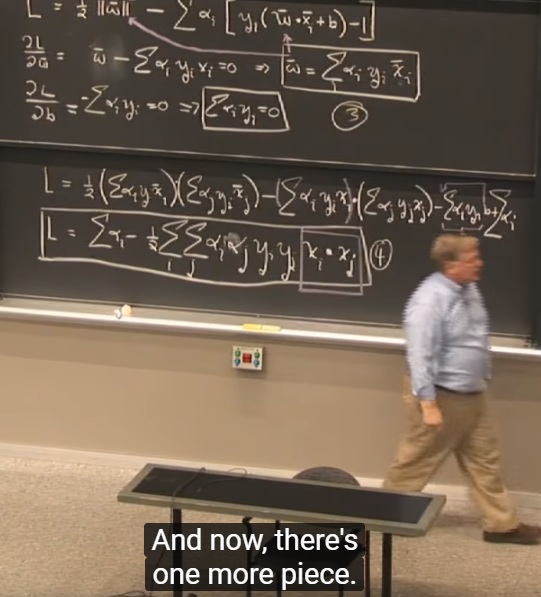
F,{4f850dc2-8705-4fda-9fc5-50adf83385c0}{169},3.125,3.125

22:35 lagrange

Chceme minimalizovat **3.** za podmienky **2.** rovnice

Ak chceme najst extrem funkcie s urcitymi podmienkami, pouzijeme Lagrange multiplikatory, ktore nam daju novy vyraz, ktory mozme maximalizovat alebo minimalizovat bez toho aby sme mysleli na podmienky

v prvom riadku obrazku sa vyraz [y\_i(W\*X+b)-1]} rovna nule - tie ktore sa nerovnaju nule, su vektory ktore lezia na tych dvoch GUTTEROCH, vsetky ostatne sa rovnaju nule

F,{4f850dc2-8705-4fda-9fc5-50adf83385c0}{189},3.125,3.125

- vyraz W dosadime do **decision rule 1.**

W\*U -+ B >= 0 **(1. DECISION RULE)**

sum{alpha\_i\*y\_i\*X\_i} \* U + b >= 0 then + sample

ZOSIT PODROBNEJSIE VYSVETLUJE (podla prednasky matfyzu)

KERNEL TRICKS 5.7.2 deep learning book online

The kernel trick is powerful for two reasons. First, it enables us to learn modelsthat are nonlinear as a function ofxusing convex optimization techniques thatare guaranteed to converge eﬃciently. This is possible because we considerφﬁxedand optimize onlyα, that is, the optimization algorithm can view the decisionfunction as being linear in a diﬀerent space. Second, the kernel functionkoften

ANALYTICKA GEOMETRIA

máme normálový vektor Theta, to je nejaký vektor kolmý na tú nadrovinu. A teda čo robíme stým norm. vektorom? máme nejakú rovnicu priamky ktorá nám char. všetky body ktoré na danej priamke ležia. To zn. vtedy vystupuje norm. vekt. a keď zobereme lub. bod X a urobíme skalárny súčin, urobíme nejaký posun b, alebo tak všetky body ktoré vyhovujú tejto rovnici, tak sú body na tejto nadrovine. ROVNICA NADROVINY, na jednej strane +, na druhej -, 0 leží

znamienko určuje na ktorej strane nadroviny leží.

<https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf>

Text Categorization with Support Vector Machines: Learning with Many Relevant Features

**4 Why Should SVMs Work Well for Text Categorization?**

To find out what methods are promising for learning text classiers, we should nd out more about the properties of text.

**High dimensional input space:** When learning text classiers, one has to deal with very many (more than 10000) features. Since SVMs use overtting protection, which does not necessarily depend on the number of features, they have the potential to handle these large feature spaces.

**Few irrelevant features:** One way to avoid these high dimensional input spaces is to assume that most of the features are irrelevant. Feature selection tries to determine these irrelevant features. Unfortunately, in text categorization there are only very few irrelevant features. Figure 1 shows the results of an experiment on the Reuters \acq" category (see section 5). All features are ranked according to their (binary) information gain. Then a naive Bayes classier [2] is trained using only those features ranked 1-200, 201-500, 501- 1000, 1001-2000, 2001-4000, 4001-9962. The results in gure 1 show that even features ranked lowest still contain considerable information and are somewhat relevant. A classier using only those \worst" features has a performance much better than random. Since it seems unlikely that all those features are completely redundant, this leads to the conjecture that a good classier should combine many features (learn a \dense" concept) and that aggressive feature selection may result in a loss of information.

**Document vectors are sparse:** For each document, the corresponding docu- ment vector contains only few entries which are not zero. Kivinen et al. [4] give both theoretical and empirical evidence for the mistake bound model that \additive" algorithms, which have a similar inductive bias like SVMs, are well suited for problems with dense concepts and sparse instances.

**Most text categorization problems are linearly separable:** All Ohsumed categories are linearly separable and so are many of the Reuters (see section 5) tasks. The idea of SVMs is to nd such linear (or polynomial, RBF, etc.) separators.

These arguments give theoretical evidence that SVMs should perform well for text categorization.

LinearSVC difference between LinearSVC and SVC(kernel='linear')

The key principles of that difference are the following:

* By default scaling, LinearSVC minimizes the squared hinge loss while SVC minimizes the regular hinge loss. It is possible to manually define a 'hinge' string for loss parameter in LinearSVC.
* LinearSVC uses the One-vs-All (also known as [One-vs-Rest](https://en.wikipedia.org/wiki/Multiclass_classification%23One-vs.-rest)) multiclass reduction while SVCuses the [One-vs-One](https://en.wikipedia.org/wiki/Multiclass_classification%23One-vs.-one) multiclass reduction. It is also noted [here](http://scikit-learn.org/stable/modules/svm.html%23multi-class-classification). Also, for multi-class classification problem SVC fits N \* (N - 1) / 2 models where N is the amount of classes. LinearSVC, by contrast, simply fits N models. If the classification problem is binary, then only one model is fit in both scenarios. multi\_class and decision\_function\_shape parameters have nothing in common. The second one is an aggregator that transforms the results of the decision function in a convenient shape of (n\_features, n\_samples). multi\_class is an algorithmic approach to establish a solution.
* The underlying estimators for LinearSVC are **liblinear**, that do in fact penalize the intercept. SVC uses **libsvm** estimators that do not. **liblinear** estimators are optimized for a linear (special) case and thus converge faster on big amounts of data than **libsvm**. That is why LinearSVC takes less time to solve the problem.

In fact, LinearSVC is not actually linear after the intercept scaling as it was stated in the comments section.