Supervised Machine Learning

AIMA Chapter 18, 20

[Adapted from slides by Dan Klein and Pieter Abbeel at UC Berkeley]

Machine Learning

- Up until now: how to use a model to make optimal decisions
  - Except reinforcement learning

- Machine learning: how to acquire a model from data / experience

learning from data

- Related courses
  - SI151 Optimization and Machine Learning
  - CS282 Machine Learning
  - CS280 Deep Learning

Types of Learning

- Supervised learning
  - Training data includes desired outputs
- Unsupervised learning
  - Training data does not include desired outputs
- Semi-supervised learning
  - Training data includes a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions

标签(监督信号)

学习 X→Y之间的映射

upgrade ⇒ 自监督：依赖变化的X学习.

只有data x，无y，从本身找规律

X→Y（只有一部分Y打了标签）

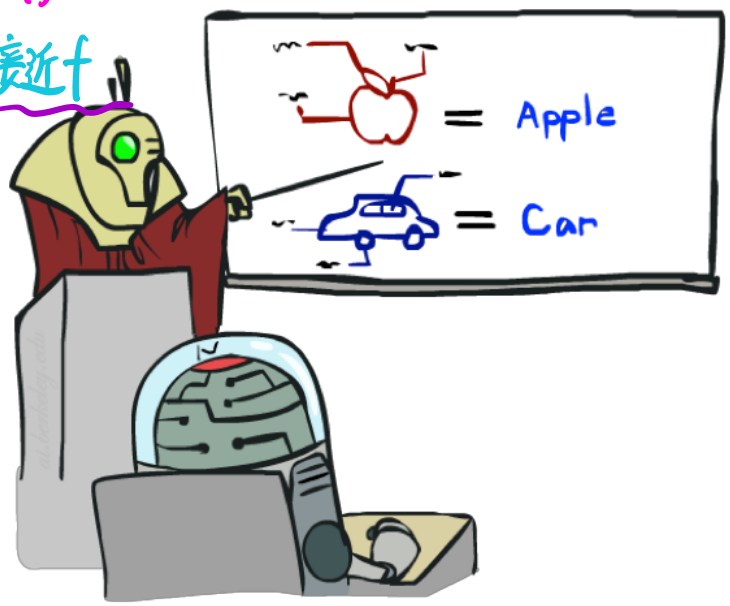
Supervised learning

- To learn an unknown *target function* f
- Input: a *training set* of *labeled examples* (x_j, y_j) where $y_j = f(x_j)$
- Output: *hypothesis* h that is "close" to f

通过拟合程度来判定.

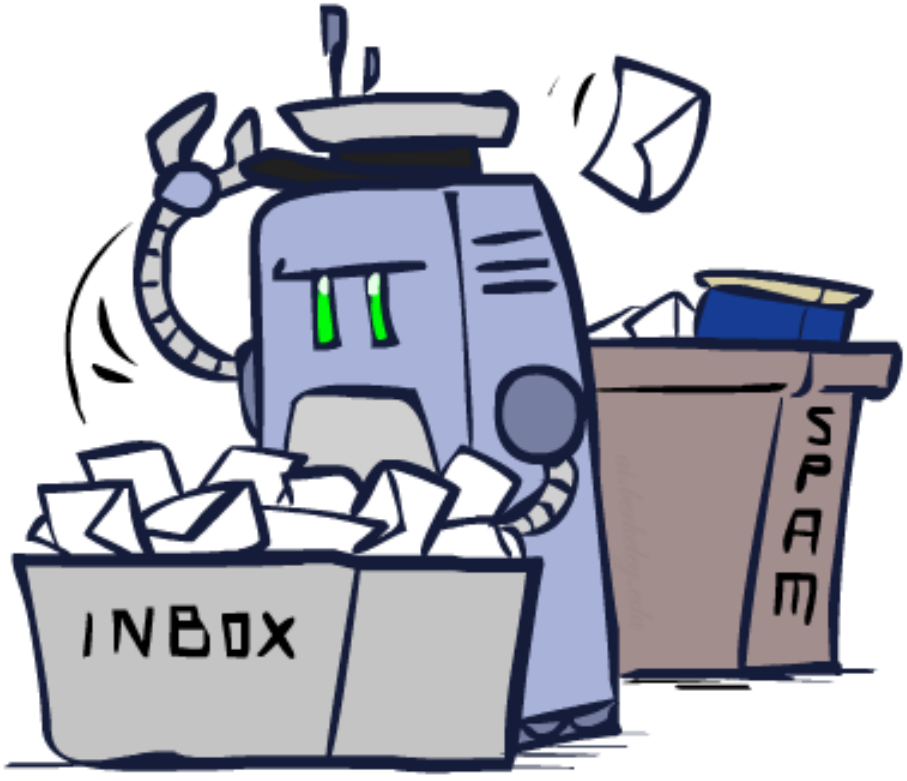
f为实际的映射关系

h为假设的映射关系, St. h接近f



- Types of supervised learning
  - Classification = learning f with discrete output value 分类算法(对于离散的output)
  - Regression = learning f with real-valued output value 回归算法(对于连续的output)
  - Structured prediction = learning f with structured output 结构化预测 eg. NLP中可能输出词的树结构.

Classification



Example: Spam Filter

- Input: an email 通过邮件
- Output: spam/ham 输出垃圾or not

- Setup:
  - Get a large collection of example emails, each labeled "spam" or "ham" (by hand)准备:人工标记好的mail
  - Want to learn to predict labels of new, future emails

- Features: The attributes used to make the ham / spam decision
  - Words: FREE!
  - Text Patterns: $dd, CAPS
  - Non-text: SenderInContacts
  - …

Dear Sir.

First, I must solicit your confidence in this transaction, this is by virture of its nature as being utterly confidencial and top secret. …

TO BE REMOVED FROM FUTURE MAILINGS, SIMPLY REPLY TO THIS MESSAGE AND PUT "REMOVE" IN THE SUBJECT.

99 MILLION EMAIL ADDRESSES FOR ONLY $99

Ok, Iknow this is blatantly OT but I'm beginning to go insane. Had an old Dell Dimension XPS sitting in the corner and decided to put it to use, I know it was working pre being stuck in the corner, but when I plugged it in, hit the power nothing happened.

Example: Digit Recognition

- Input: images / pixel grids
- Output: a digit 0-9

- Setup:
  - Get a large collection of example images, each labeled with a digit
  - Want to learn to predict labels of new, future digit images

- Features: The attributes used to make the digit decision
  - Pixels: (6,8)=ON
  - Shape Patterns: NumComponents, AspectRatio, NumLoops
  - …


0


1


2

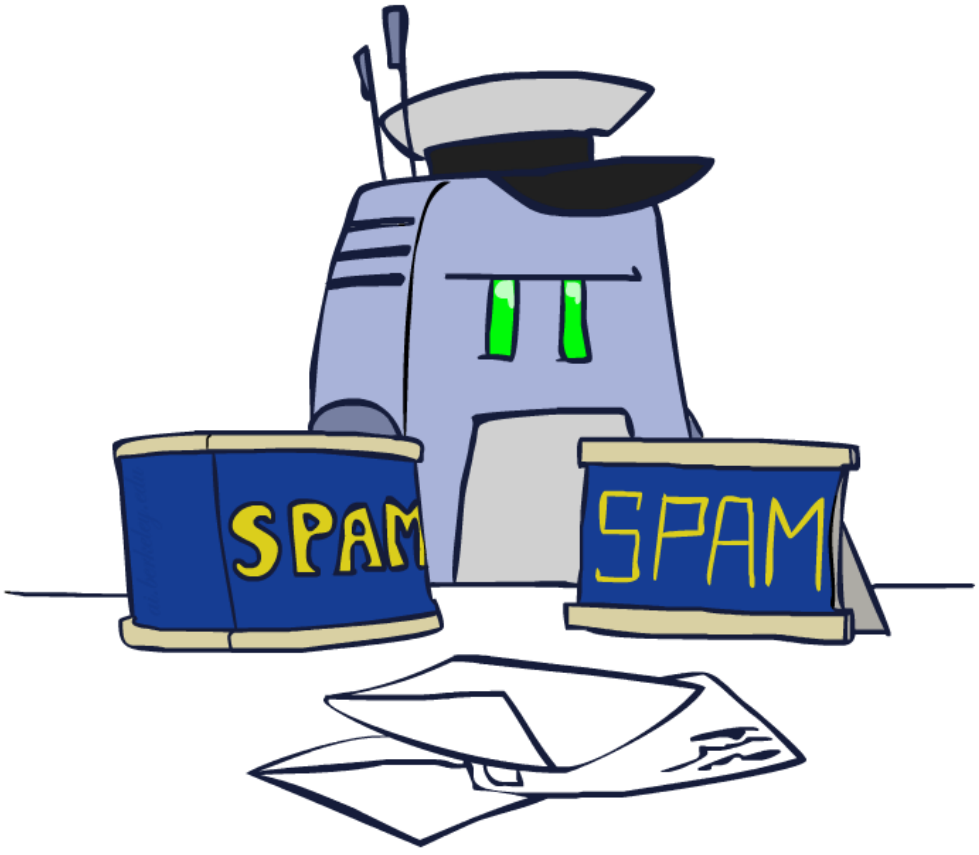

1


??

Other Classification Tasks

- Medical diagnosis
  - input: symptoms
  - output: disease
- Automatic essay grading
  - input: document
  - output: grades
- Fraud detection
  - input: account activity
  - output: fraud / no fraud
- Email routing
  - input: customer complaint email
  - output: which department needs to ignore this email
- Fruit and vegetable inspection
  - input: image (or gas analysis)
  - output: moldy or OK
- … many more

Naïve Bayes Classifier



Model-Based Classification

- Model-based approach
  - Build a model (e.g. Bayes' net) where both the label and features are random variables 建立一个贝叶斯网络，包含标签和特征
  - Instantiate any observed features 包含所有观察的特征
  - Query for the distribution of the label conditioned on the features 查询label的条件概率
- Challenges
  - What structure should the BN have?
  - How should we learn its parameters?

Naïve Bayes for Digits

- Naïve Bayes: Assume all features are independent effects of the label

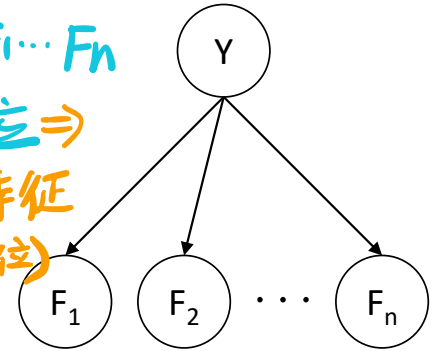
朴素贝叶斯：假设所有特征独立

- Simple digit recognition version:
  - One feature (variable) F_{ij} for each grid position <i,j>
  - Feature values are on / off, based on whether intensity is more or less than 0.5 in underlying image
  - Each input maps to a feature vector, e.g.

$$\rightarrow \langle F_{0,0} = 0 \;\; F_{0,1} = 0 \;\; F_{0,2} = 1 \;\; F_{0,3} = 1 \;\; F_{0,4} = 0 \;\; \ldots F_{15,15} = 0 \rangle$$

  - Here: lots of features, each is binary valued

given Y, F1 … Fn 都条件独立⇒ 朴素(所有特征之间都相互独立)



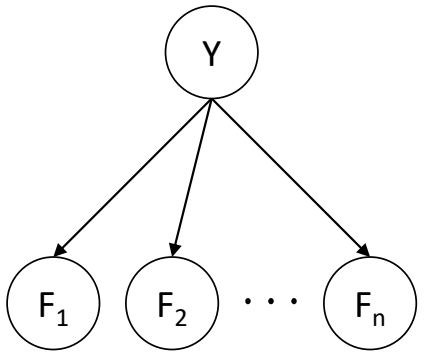
- Naïve Bayes model: $P(Y|F_{0,0} \ldots F_{15,15}) \propto P(Y) \prod_{i,j} P(F_{i,j}|Y)$

不用考虑feature之间的联系.

- What do we need to learn?

General Naïve Bayes

- A general Naive Bayes model:



|Y| parameters

$$P(Y, F_1 \ldots F_n) = P(Y) \prod_i P(F_i|Y)$$

|Y| x |F|^n values

n x |F| x |Y| parameters

有n个位置，每个位置有F种feature.

- We only have to specify how each feature depends on the class
- Total number of parameters is *linear* in n
- Model is very simplistic, but often works anyway

Inference for Naïve Bayes

- Goal: compute posterior distribution over label variable Y
  - Step 1: get joint probability of label and evidence for each label

$$P(Y, f_1 \dots f_n) = \begin{bmatrix} P(y_1, f_1 \dots f_n) \\ P(y_2, f_1 \dots f_n) \\ \vdots \\ P(y_k, f_1 \dots f_n) \end{bmatrix} \Rightarrow \begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}$$

$$\Rightarrow \frac{\begin{bmatrix} P(y_1) \prod_i P(f_i|y_1) \\ P(y_2) \prod_i P(f_i|y_2) \\ \vdots \\ P(y_k) \prod_i P(f_i|y_k) \end{bmatrix}}{P(f_1 \dots f_n)} \quad (+)$$

  - Step 2: normalization

$$\Rightarrow P(Y|f_1 \dots f_n)$$

归一化

General Naïve Bayes

- What do we need in order to use Naïve Bayes?
  - Inference method (we just saw this part) 根据features inference 出 label
    - Start with a bunch of probabilities: P(Y) and the $P(F_i|Y)$ tables
    - Use standard inference to compute $P(Y|F_1 \ldots F_n)$
    - Nothing new here
  - Estimates of local conditional probability tables
    - P(Y), the prior over labels
    - $P(F_i|Y)$ for each feature (evidence variable)
    - These probabilities are collectively called the *parameters* of the model and denoted by θ 共同的
    - Up until now, we assumed these appeared by magic, but…
    - …they typically come from training data counts: we'll look at this soon

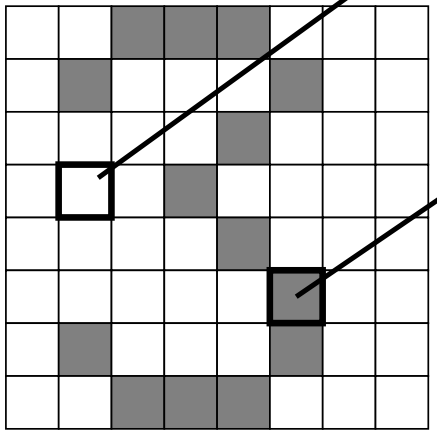
Example: Conditional Probabilities

Or 根据所有数字label的频率当概率.

所有1中在(3,1)出现的次数 / 共计所有1

$P(Y)$

1	0.1
2	0.1
3	0.1
4	0.1
5	0.1
6	0.1
7	0.1
8	0.1
9	0.1
0	0.1



$P(F_{3,1} = on|Y)$

1	0.01
2	0.05
3	0.05
4	0.30
5	0.80
6	0.90
7	0.05
8	0.60
9	0.50
0	0.80

$P(F_{5,5} = on|Y)$

1	0.05
2	0.01
3	0.90
4	0.80
5	0.90
6	0.90
7	0.25
8	0.85
9	0.60
0	0.80

Naïve Bayes for Text

- Bag-of-words Naïve Bayes:
  - Features: W_i is the word at positon i

$$P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$$

Word at position i, not i^{th} word in the dictionary!

  - Usually, each variable gets its own conditional probability distribution P(F|Y)
  - Here
    - Each position is identically distributed
    - All positions share the same conditional probabilities P(W|Y)
  - Called "bag-of-words" because model is insensitive to word order or reordering

每一个位置都有相同的P(W|Y)

只统计单词的频率，不用考虑单词间的顺序了.

Example: Spam Filtering

- Model: $P(Y, W_1 \dots W_n) = P(Y) \prod_i P(W_i|Y)$

$P(Y)$

```
ham : 0.66
spam: 0.33
```

$P(W|\text{spam})$

```
the :  0.0156
to  :  0.0153
and :  0.0115
of  :  0.0095
you :  0.0093
a   :  0.0086
with:  0.0080
from:  0.0075
...
```

$P(W|\text{ham})$

```
the :  0.0210
to  :  0.0133
of  :  0.0119
2002:  0.0110
with:  0.0108
from:  0.0107
and :  0.0105
a   :  0.0100
...
```

Spam Example

$P(Y)$
$P(W_1|Y)$
$P(W_2|Y)$

Word	P(w|spam)	P(w|ham)	Tot Spam	Tot Ham
(prior)	0.33333	0.66666	-1.1	-0.4
Gary	0.00002	0.00021	-11.8	-8.9
would	0.00069	0.00084	-19.1	-16.0
you	0.00881	0.00304	-23.8	-21.8
like	0.00086	0.00083	-30.9	-28.9
to	0.01517	0.01339	-35.1	-33.2
lose	0.00008	0.00002	-44.5	-44.0
weight	0.00016	0.00002	-53.3	-55.0
while	0.00027	0.00027	-61.5	-63.2
you	0.00881	0.00304	-66.2	-69.0
sleep	0.00006	0.00001	-76.0	-80.5

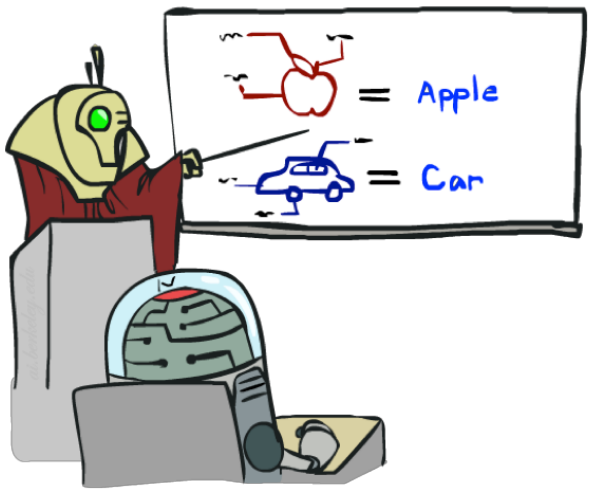
$-1.1 = \ln(0.3333)$ $-0.4 = \ln(0.6666\ldots)$

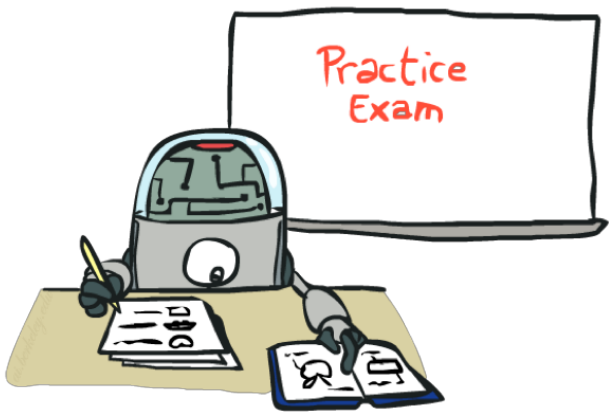
Log product of probabilities

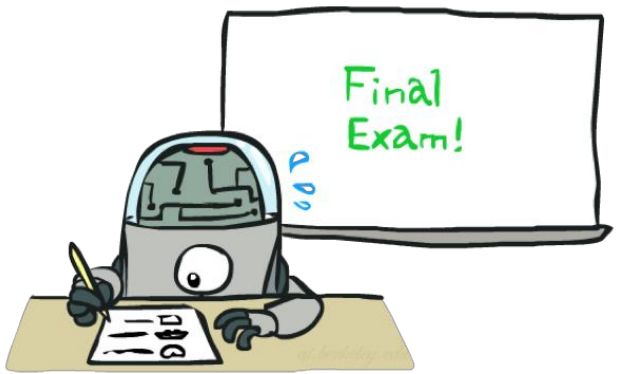
$\log[P(Y)\prod_i P(w_i|Y)] = \log P(Y) + \sum_i \log P(w_i|Y)$

P(spam | w) = 98.9

Training and Testing





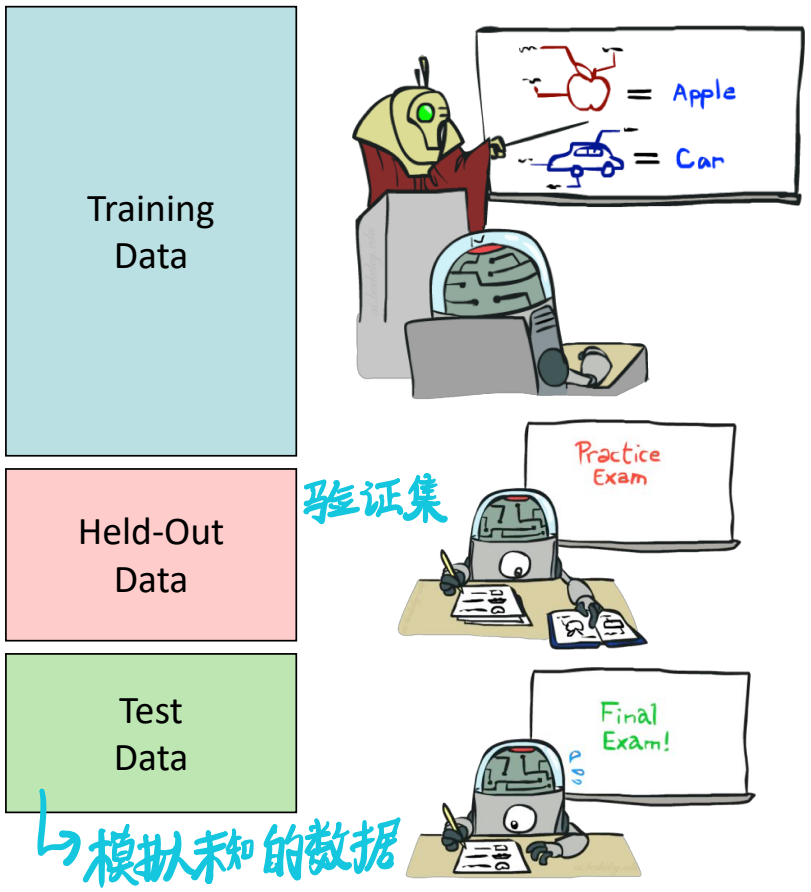


Important Concepts

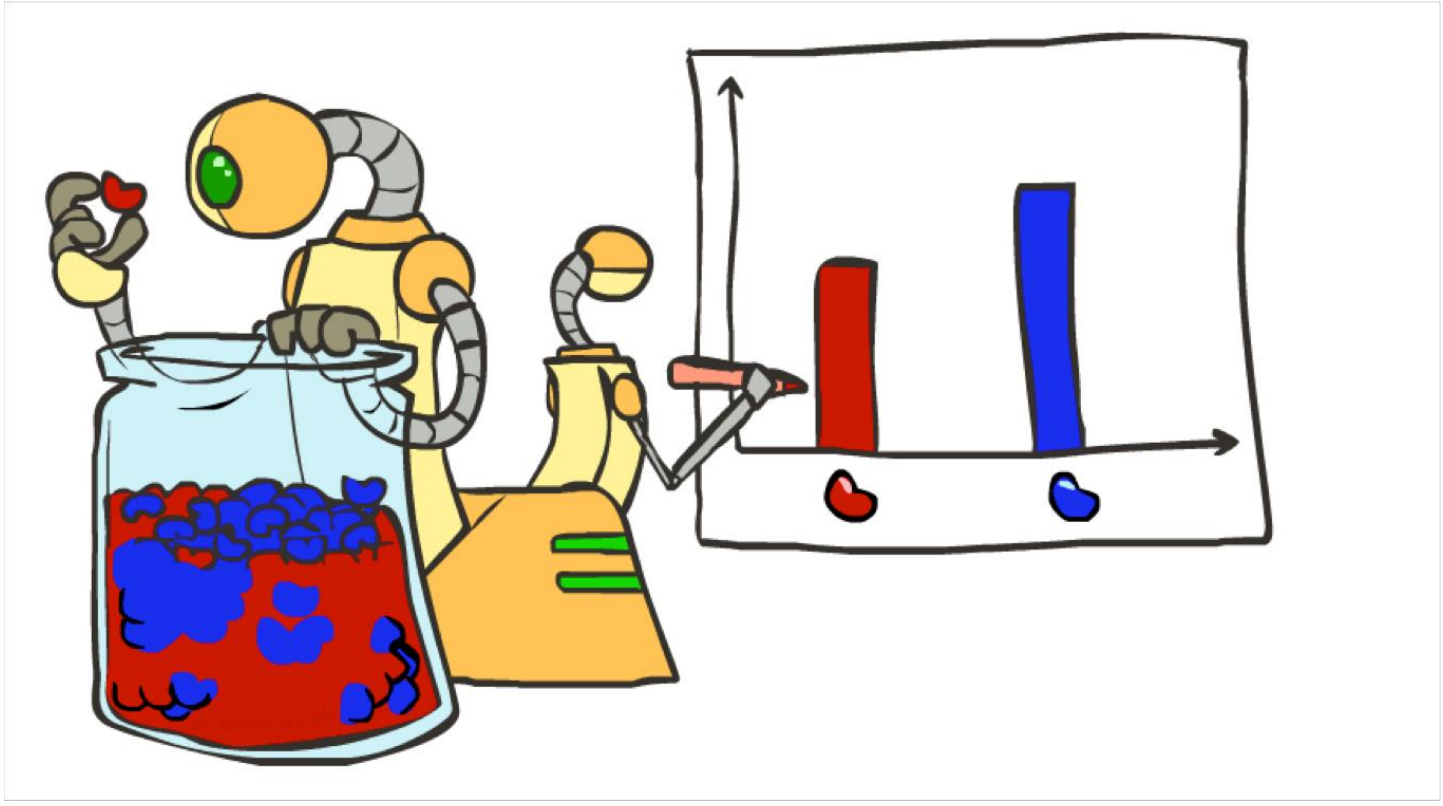
- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out set
  - Test set
- Experimentation cycle
  - Learn parameters (e.g. model probabilities) on training set
  - Tune hyperparameters on held-out set
  - Compute accuracy of test set (fraction of instances predicted correctly)
  - Very important: never "peek" at the test set!

超参数在held-out调整

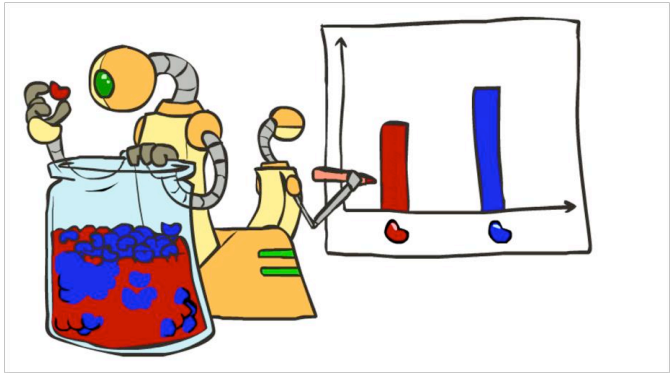
不能在测试集上选择模型.



Training



Parameter Estimation



- Estimating the distribution of a random variable
- *Elicitation:* ask a human (this is hard…)
- *Empirically:* use training data (learning!)
  - For each outcome x, look at the *empirical rate* of that value

$$P_{\text{ML}}(x) = \frac{\text{count}(x)}{\text{total samples}}$$

  - Ex:
    - We've seen 1000 words from spam emails, among which we see "money" for 50 times
    - So we set P(money | spam) = 0.05
  - This is the estimate that maximizes the *likelihood of the data*
    - Likelihood: conditional probability of the data given the parameters

化简: Given一组参数, 看到样本的概率.

Maximum Likelihood Estimation

- Coin flipping:
  - P(Heads) = θ, P(Tails) = 1-θ
- Flips are *i.i.d.*
  - Independent events
  - Identically distributed according to unknown distribution
- Sequence *D* of α_H Heads and α_T Tails

$$P(\mathcal{D} \mid \theta) = \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

Maximum Likelihood Estimation

- **MLE:** Choose θ to maximize probability of *D*

极大似然估计

对数不会影响单调性，但是可以简化求导.

$$\hat{\theta} = \arg\max_{\theta} \ln P(\mathcal{D} \mid \theta)$$
$$= \arg\max_{\theta} \ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}$$

- Set derivative to zero, and solve!

$$\frac{d}{d\theta} \ln P(\mathcal{D} \mid \theta) = \frac{d}{d\theta} [\ln \theta^{\alpha_H}(1-\theta)^{\alpha_T}]$$

$$= \frac{d}{d\theta} [\alpha_H \ln \theta + \alpha_T \ln(1-\theta)]$$

$$= \alpha_H \frac{d}{d\theta} \ln \theta + \alpha_T \frac{d}{d\theta} \ln(1-\theta)$$

$$= \frac{\alpha_H}{\theta} - \frac{\alpha_T}{1-\theta} = 0$$

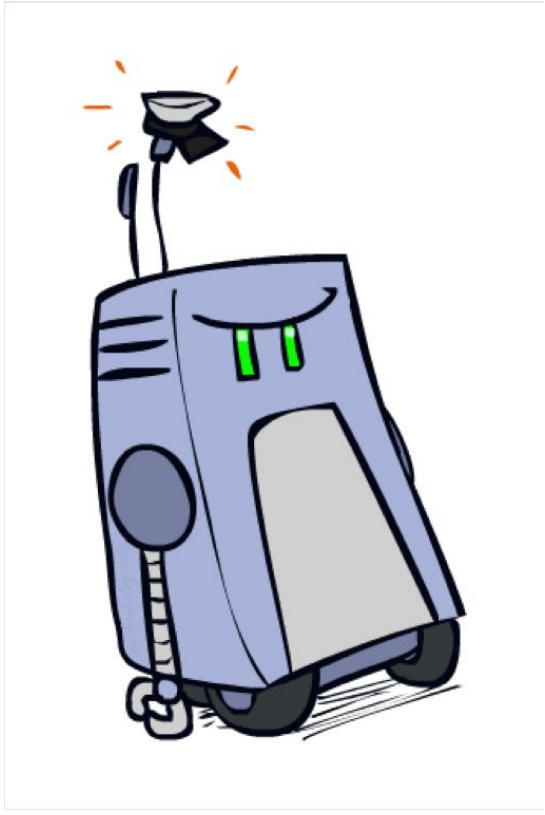
$$\hat{\theta}_{MLE} = \frac{\alpha_H}{\alpha_H + \alpha_T}$$

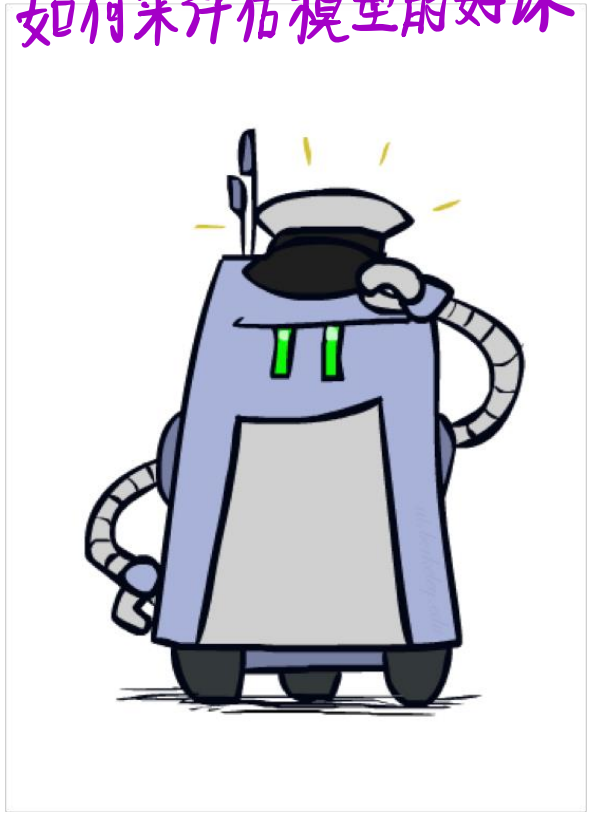
(泛化) (过拟合: 追求训练集表现)

Generalization and Overfitting

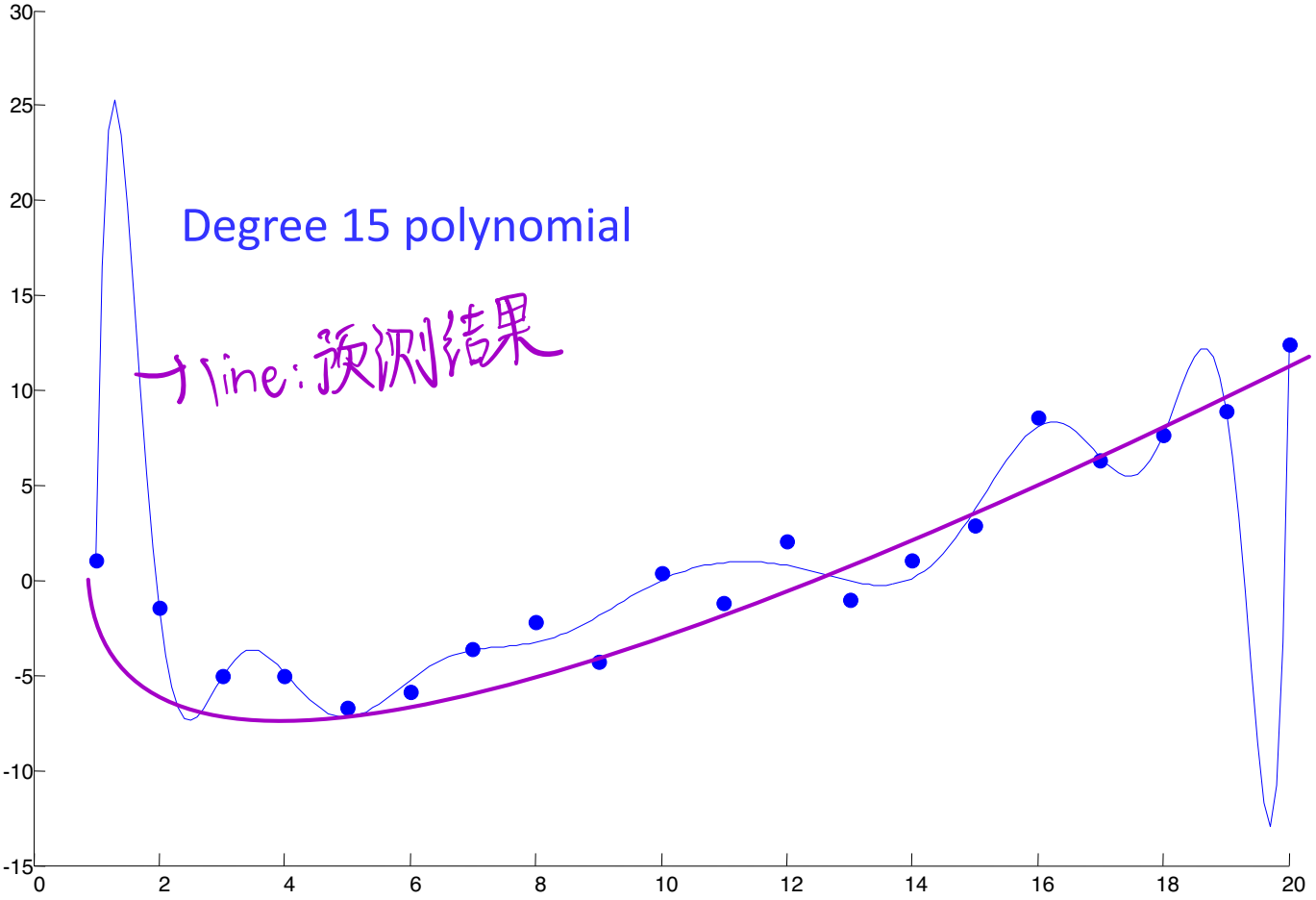
tradeoff

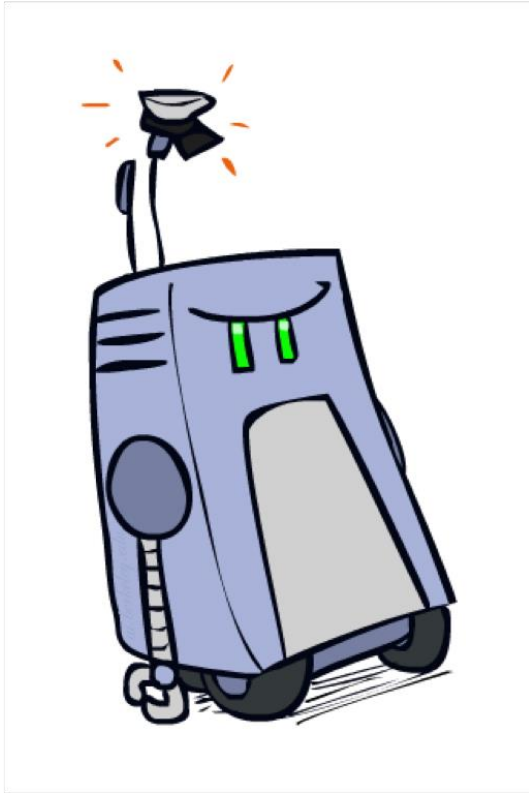
如何来评估模型的好坏





Overfitting





Example: Overfitting

$P(\text{features}, C = 2)$

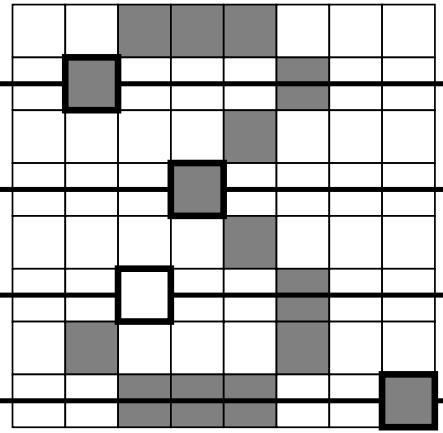
$P(C = 2) = 0.1$ (均匀分布)

$P(\text{on}|C = 2) = 0.8$

$P(\text{on}|C = 2) = 0.1$

$P(\text{off}|C = 2) = 0.1$

$P(\text{on}|C = 2) = 0.01$



$P(\text{features}, C = 3)$

$P(C = 3) = 0.1$

$P(\text{on}|C = 3) = 0.8$

$P(\text{on}|C = 3) = 0.9$

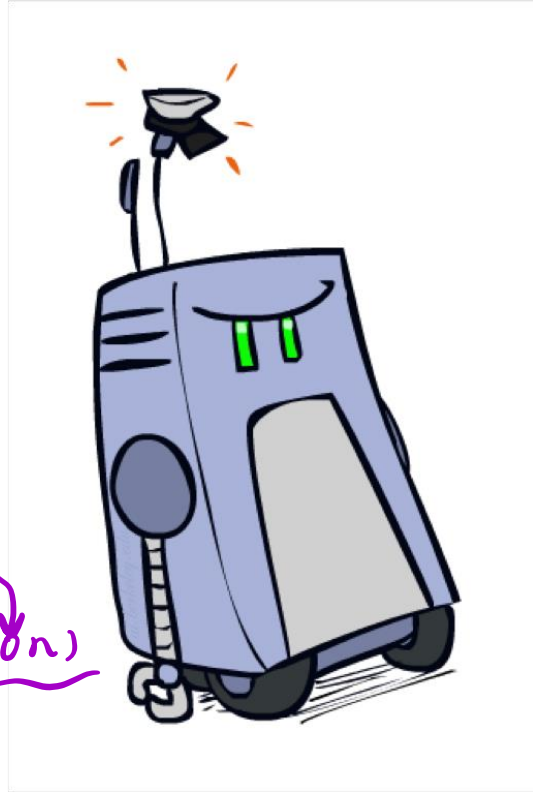
$P(\text{off}|C = 3) = 0.7$

$P(\text{on}|C = 3) = 0.0$

☆ $P(C=3|\cdots,(8,8)=on)$
$=0$

test 3: 失败 (∵过拟合)

2 wins!!



Example: Overfitting

- Posteriors determined by *relative* probabilities (odds ratios):

分母一旦有零就不成立了.

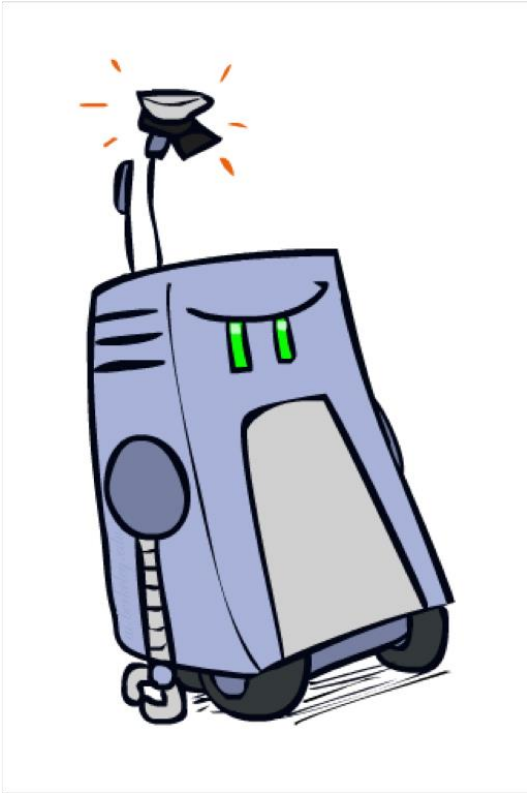
$$\frac{P(W|\text{ham})}{P(W|\text{spam})}$$

```
south-west : inf
nation     : inf
morally    : inf
nicely     : inf
extent     : inf
seriously  : inf
...
```

$$\frac{P(W|\text{spam})}{P(W|\text{ham})}$$

```
screens    : inf
minute     : inf
guaranteed : inf
$205.00    : inf
delivery   : inf
signature  : inf
...
```

What went wrong here?



Generalization and Overfitting

经验率

- Using empirical rate will overfit the training data!
  - Just because we never saw a 3 with pixel (15,15) on during training doesn't mean we won't see it at test time
  - Just because we never saw a word in spam emails during training doesn't mean we won't see it at test time
  - Therefore, we can't give unseen events zero probability 我们不能给未出现的事件0概率
  - More generally, rates in the training data may not exactly match rates at test time

即训练中的参数不能完全与测试匹配

Generalization and Overfitting

- Overfitting: learn to fit the training data very closely, but fit the test data poorly
  - Generalization: try to fit the test data as well
- Why does overfitting occur?
  - Training data is not representative of the true data distribution
    - Too few training samples
    - Training data is noisy
  - Too many attributes, some of them irrelevant to the classification task
  - The model is too expressive
    - Ex: the model is capable of memorizing all the spam emails in the training set

overfitting：训练集ac↑
验证集 ac↓

generalization：训练集ac↓
验证集 ac↑

太少训练样本

太多噪音

特征太多，毫无关系的纳入了特征

假设的模型太过于复杂(例如2阶→10阶)

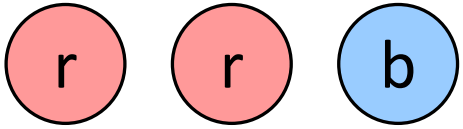
Generalization and Overfitting

- Avoid overfitting 如何避免过拟合
  - Acquire more training data (not always possible) 获得更多的数据
  - Remove irrelevant attributes (not always possible) 移除无关的特征
  - Limit the model expressiveness by regularization, early stopping, pruning, etc.
    减少复杂的模型(提前结束,剪枝)
- In our previous example, we may smooth the empirical rate to improve generalization
  做平滑来降低拟合率.

Laplace Smoothing

- Laplace's estimate:
  - Pretend you saw every outcome once more than you actually did

↗ 假设每个样本都多看见了一次



$$P_{LAP}(x) = \frac{c(x)+1}{\sum_x [c(x)+1]}$$

$$= \frac{c(x)+1}{N+|X|}$$

$$P_{ML}(X) = \frac{C(X)}{N} \quad \left(\frac{2}{3}, \frac{1}{3}\right)$$

$$P_{LAP}(X) = \frac{C(X)+1}{N+|X|} \quad \left(\frac{3}{5}, \frac{2}{5}\right)$$

  - Can derive this estimate with *Dirichlet priors*

Laplace Smoothing

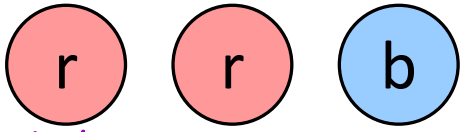
- Laplace's estimate (extended):
  - Pretend you saw every outcome k extra times

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

  - k is the strength of the prior
  - What's Laplace with k = 0?

- Laplace for conditionals:
  - Smooth each condition independently:

$$P_{LAP,k}(x|y) = \frac{c(x,y) + k}{c(y) + k|X|}$$



$P_{LAP,0}(X) = \left(\frac{2}{3}, \frac{1}{3}\right)$

$P_{LAP,1}(X) = \left(\frac{3}{5}, \frac{2}{5}\right)$

$P_{LAP,100}(X) = \left(\frac{102}{203}, \frac{101}{203}\right)$

Linear Interpolation 线性插值

- In practice, Laplace often performs poorly for P(X|Y): 空间很大，样本却很小，failed
  - When |X| is very large
  - When |Y| is very large

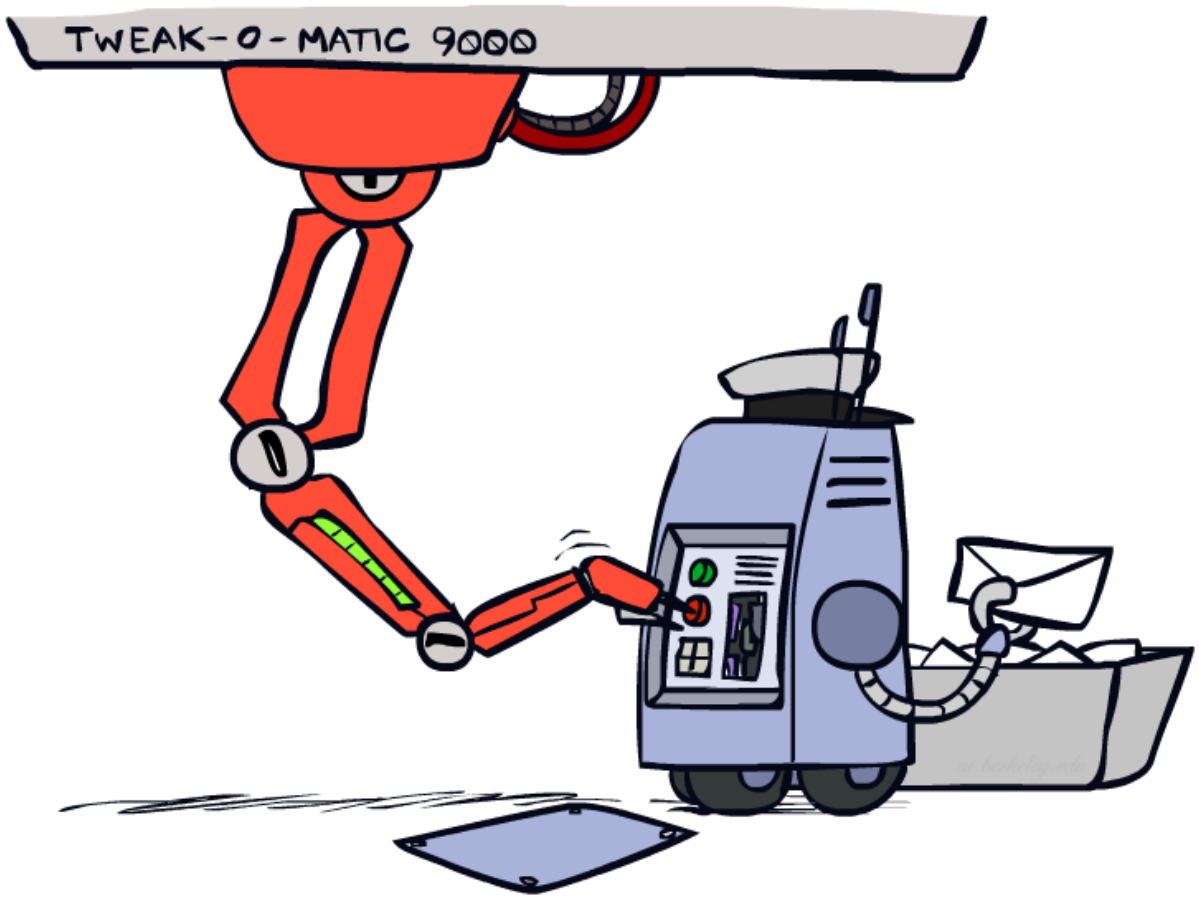
$$P_{LAP,k}\,P(X|Y) = \frac{P(X,Y)+k}{P(Y)+k|X|}$$

- Another option: linear interpolation
  - Also get the empirical P(X) from the data
  - Make sure the estimate of P(X|Y) isn't too different from the empirical P(X)

$$P_{LIN}(x|y) = \alpha\hat{P}(x|y) + (1.0-\alpha)\hat{P}(x)$$

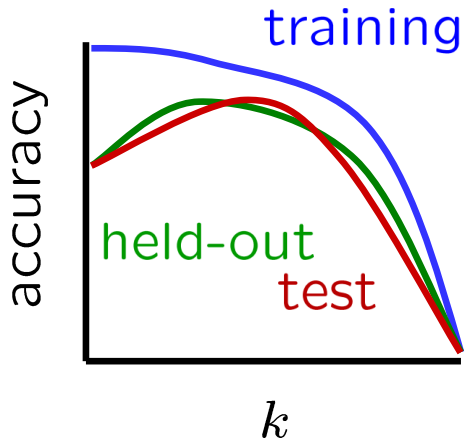
$$P_{linear}(x|y) = \alpha\hat{P}(x|y) + (1.0-\alpha)\hat{P}(x)$$

Tuning

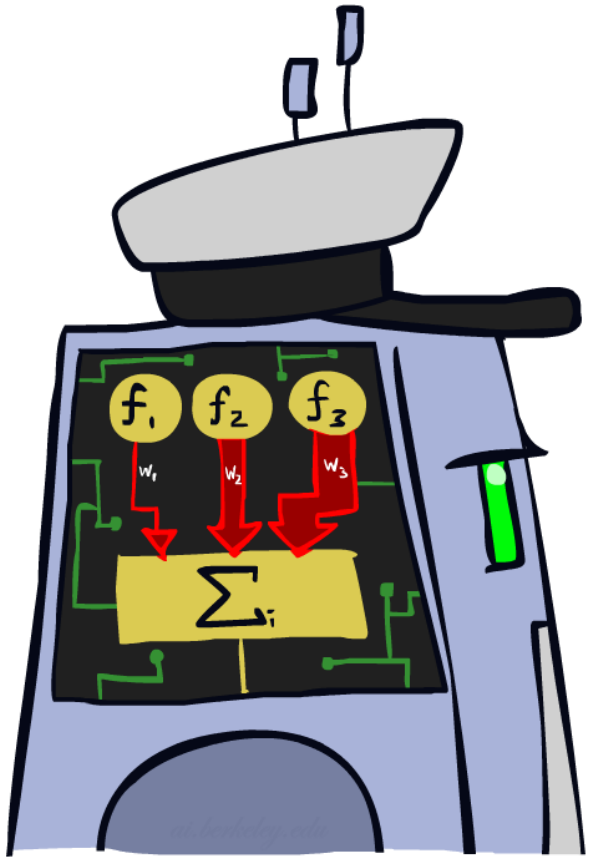


Tuning on Held-Out Data

- Now we've got two kinds of unknowns
  - Parameters: the probabilities P(X|Y), P(Y)
  - Hyperparameters: e.g. the amount / type of smoothing to do, k, α
- What should we learn where?
  - Learn parameters from training data
  - Tune hyperparameters on different data
    - Why?
  - For each value of the hyperparameter, train on the training data and test on the held-out data
  - Choose the best hyperparameter value and do a final test on the test data



Linear Classifiers 分类的边界是线性的

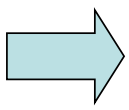


Feature Vectors

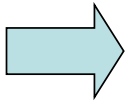
x $f(x)$ (特征) y

学习f(x)到y的映射.

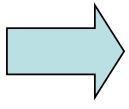
```
Hello,

Do you want free printr
cartriges?  Why pay more
when you can get them
ABSOLUTELY FREE!  Just
```



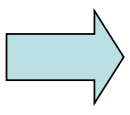
```
# free       : 2
YOUR_NAME    : 0
MISSPELLED   : 2
FROM_FRIEND  : 0
...
```



SPAM
or
+



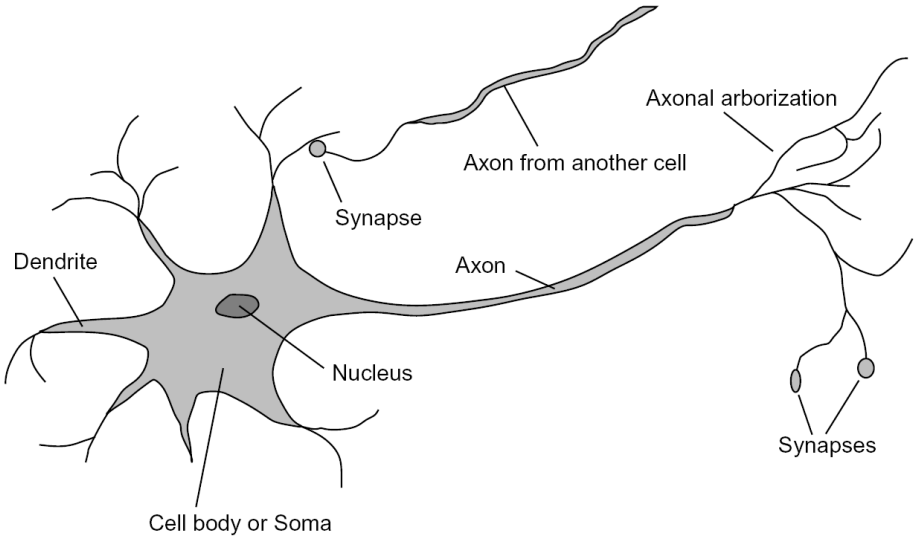
```
PIXEL-7,12   : 1
PIXEL-7,13   : 0
...
NUM_LOOPS    : 1
...
```

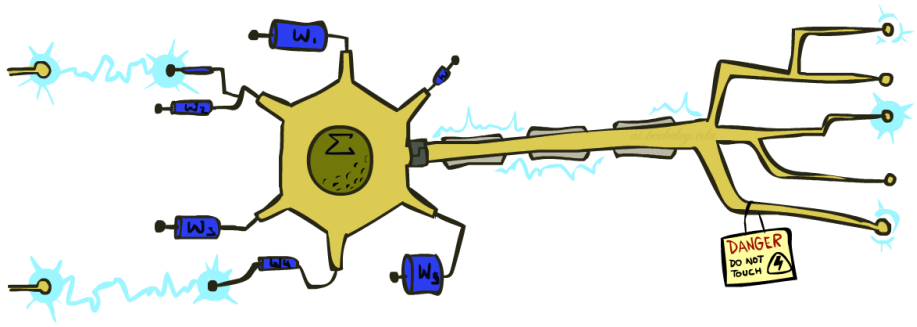


“2”

Some (Simplified) Biology

- Very loose inspiration: human neurons 神经元 ⇒ 感知器.

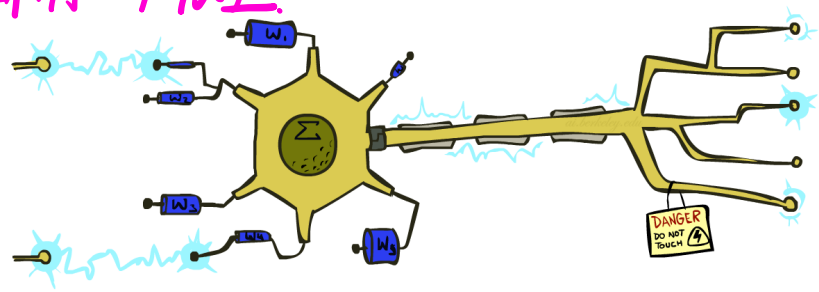




Linear Classifiers

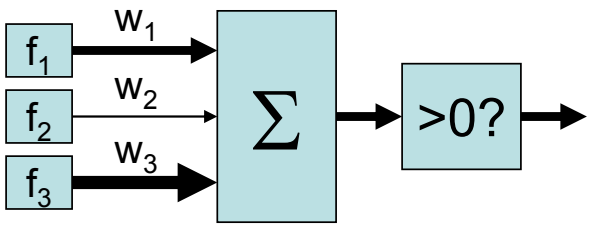
输入的是带有特征的值，每个特征都有一个权重

- Inputs are feature values
- Each feature has a weight
- Sum is the activation 激活函数



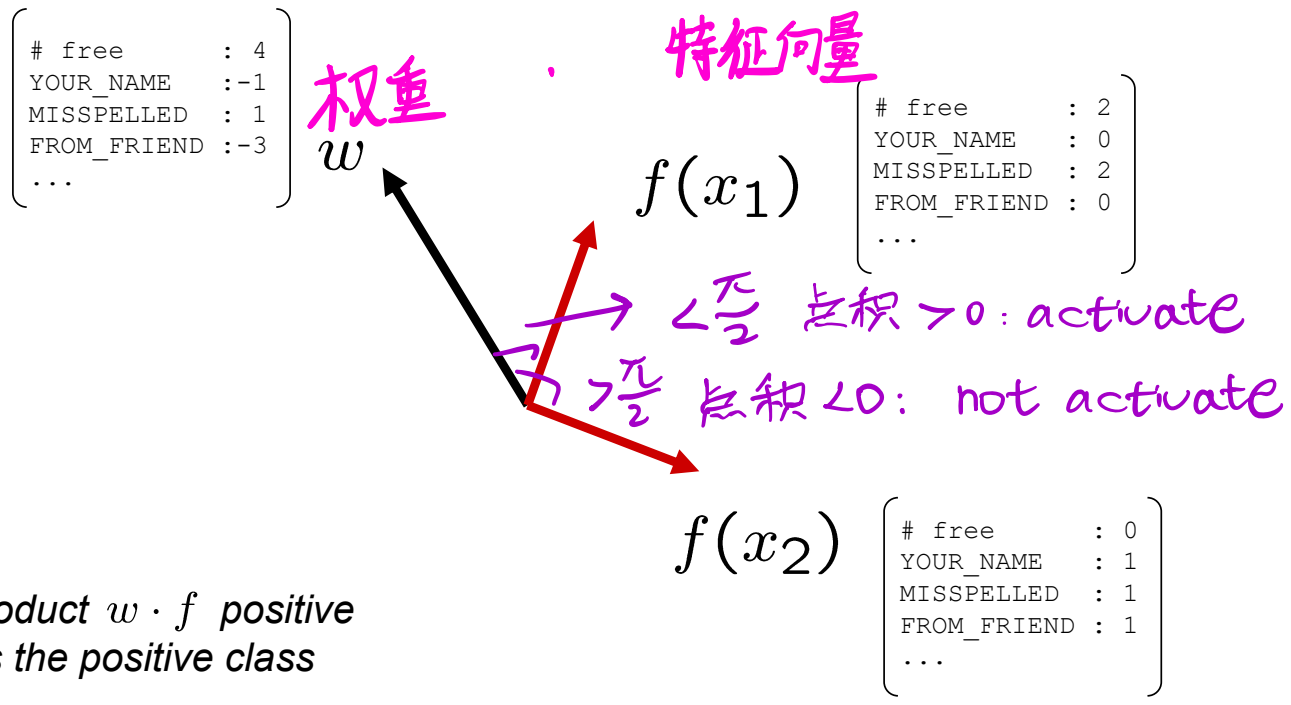
$$\text{activation}_w(x) = \sum_i w_i \cdot f_i(x) = w \cdot f(x)$$

- Binary case: if the activation is:
  - Positive, output +1
  - Negative, output -1



Weights

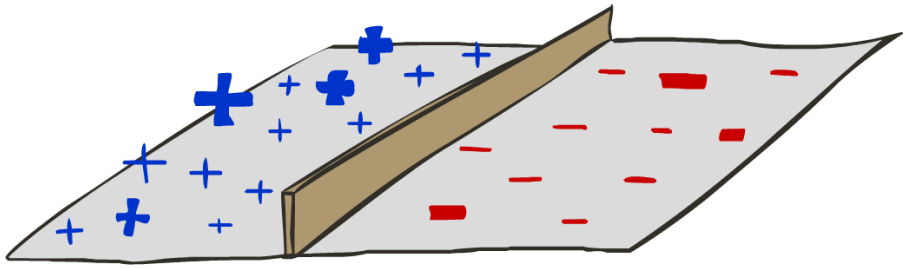
- Binary case: compare features to a weight vector
- Learning: figure out the weight vector from examples



Dot product $w \cdot f$ positive means the positive class

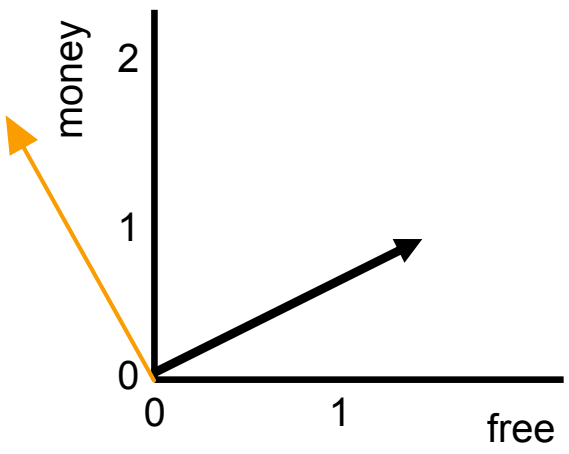
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



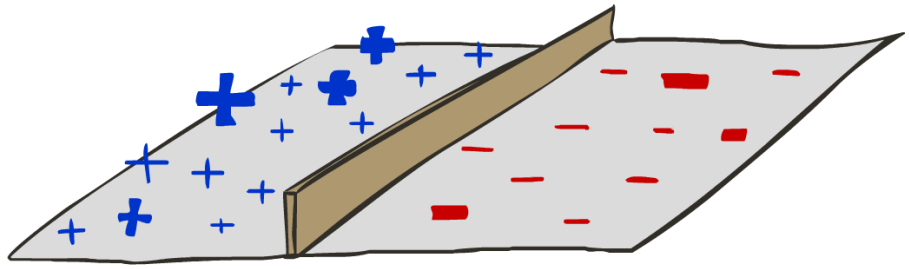
$$w$$

```
BIAS  : -3
free  :  4
money :  2
...
```



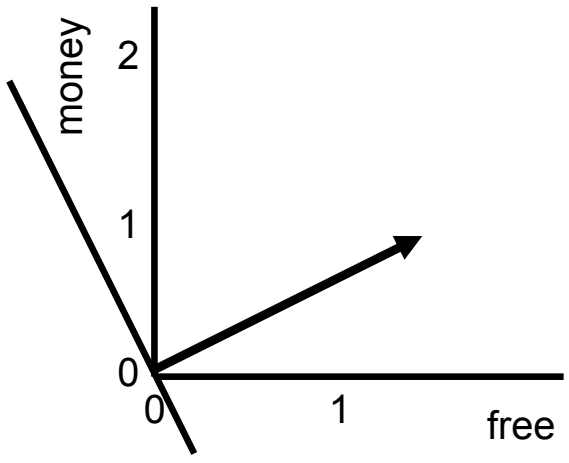
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



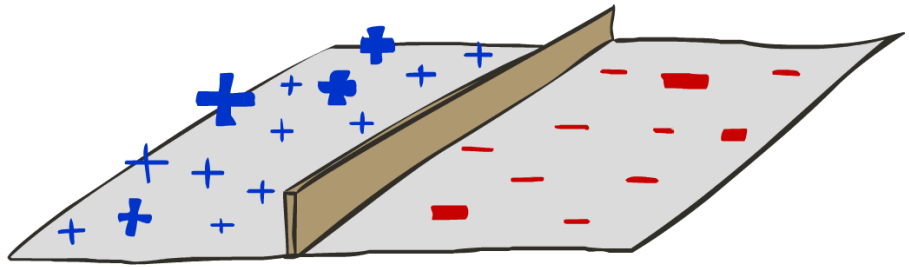
$$w$$

```
BIAS  : -3
free  :  4
money :  2
...
```



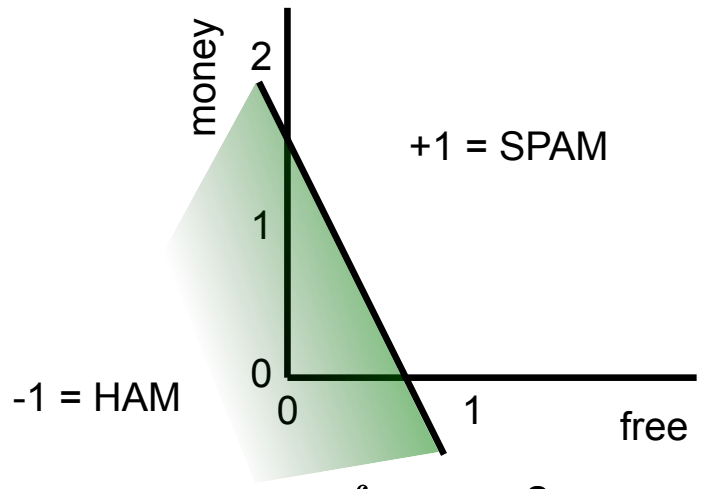
Binary Decision Rule

- In the space of feature vectors
  - Examples are points
  - Any weight vector is a hyperplane
  - One side corresponds to Y=+1
  - Other corresponds to Y=-1



w

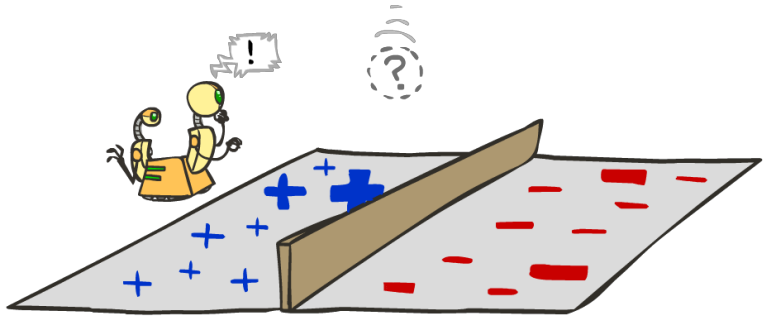
```
BIAS   : -3
free   :  4
money  :  2
...
```



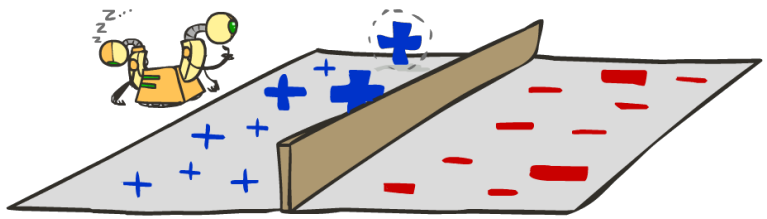
$f \cdot w = 0$

Learning: Binary Perceptron

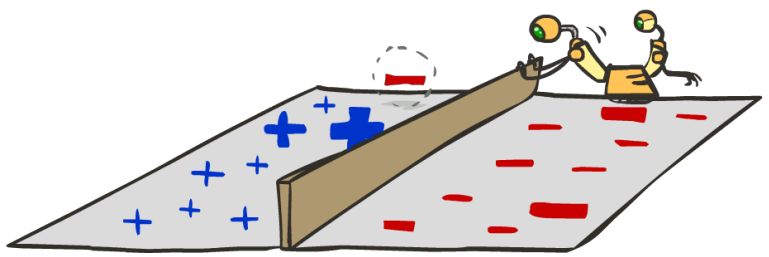
- Start with weights = 0 (初始化 w为0)
- For each training instance:
  - Classify with current weights



  - If correct (i.e., y=y*), no change!



  - If wrong: adjust the weight vector

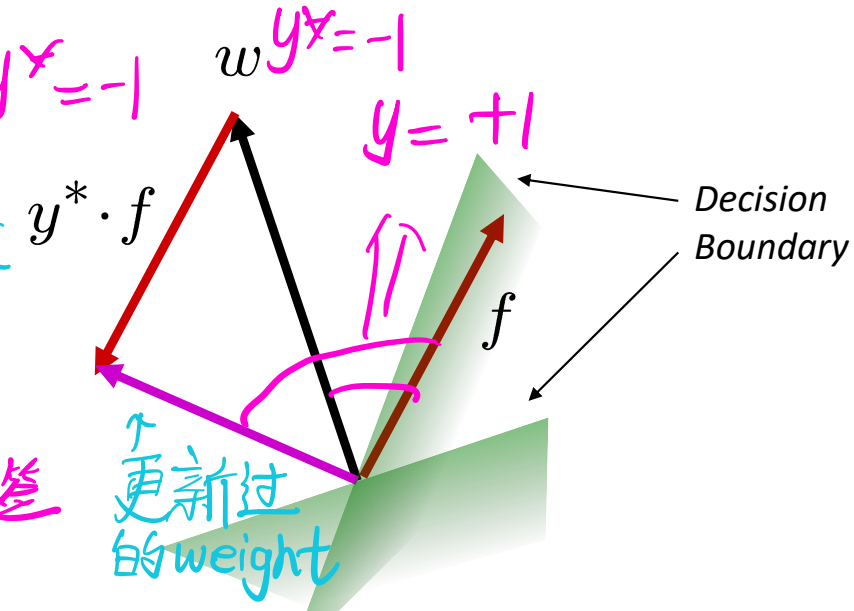


Learning: Binary Perceptron

- Start with weights = 0
- For each training instance:
  - Classify with current weights

$$y = \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) < 0 \end{cases}$$

↗激活函数



  - If correct (i.e., y=y*), no change!
  - If wrong: adjust the weight vector by adding or subtracting the feature vector.

y*为真实标签

如果不正确，则往正确的方向进行更新.

(更靠近正确但不一定变为正确)

$$w = w + y^* \cdot f$$

$W_2 = W_1 + y^* f$　　$y = W_1 f$　　$y' = (W_1 + y^* f) f = W_1 f - f^2$ 更接近负数 ($y^* = -1$)

Properties of Perceptrons 感知器的性质

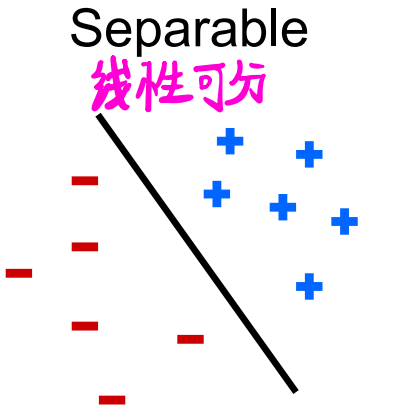
- Separability: true if some parameters get the training set perfectly correct 如果训练集正确，则可分
- Convergence: if the training is separable, perceptron will eventually converge (binary case) 训练集可分，则可收敛
- Mistake Bound: the maximum number of mistakes (binary case) related to the *margin* or degree of separability

最大错误量与边界和可分的维度有关.

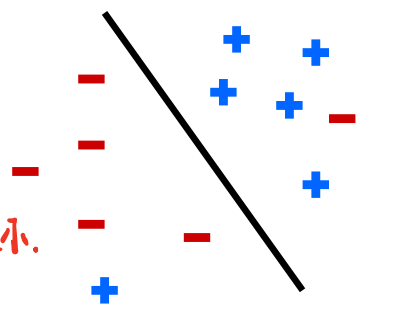
$$\text{mistakes} < \frac{k}{\delta^2}$$

k → 特征的维度

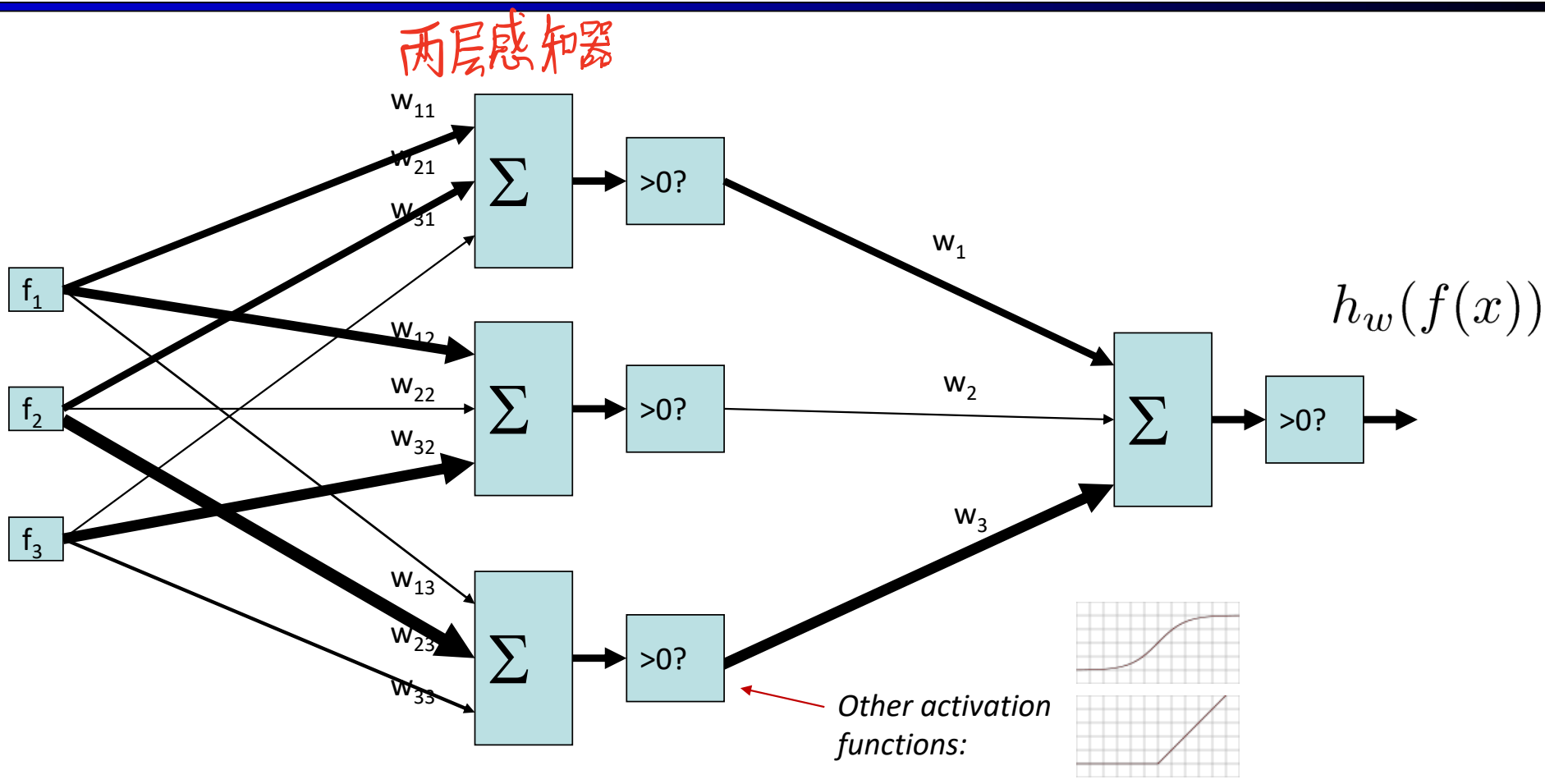
δ^2 → bound越大，可行解越多，mistake越小.



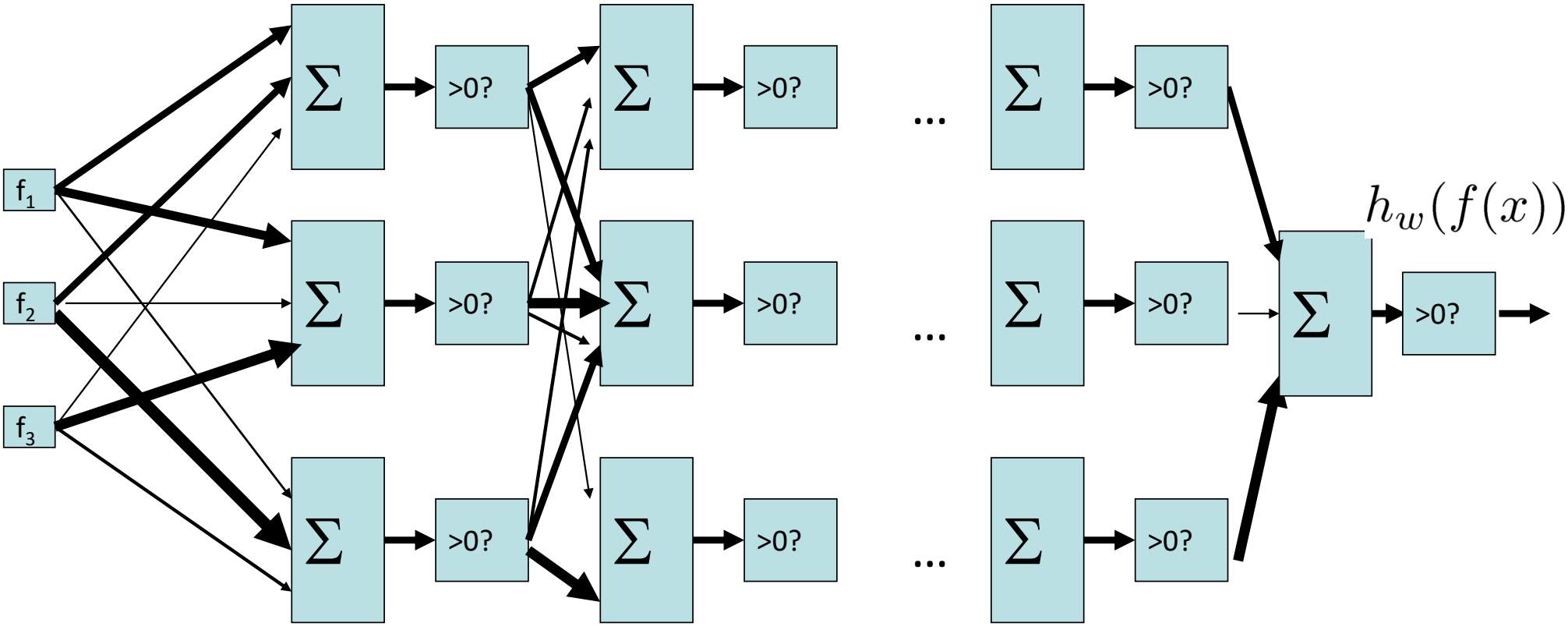
Non-Separable



Two-Layer Perceptron Network

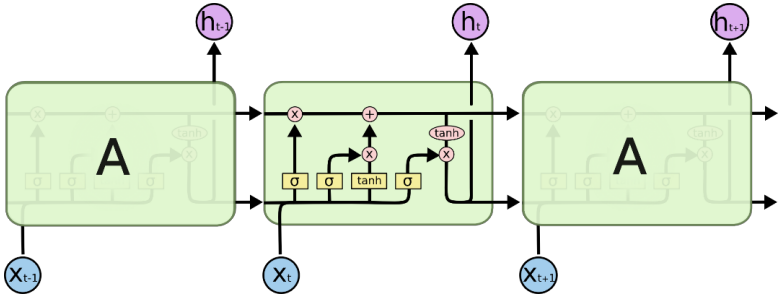


Deep Neural Network



1-page Overview of Deep Learning

- Deep Learning
  - A large number of layers of neural networks
    - Ex. 1000 layers in ResNet
  - More complicated connections between layers
    - Ex. LSTM



1-page Overview of Deep Learning

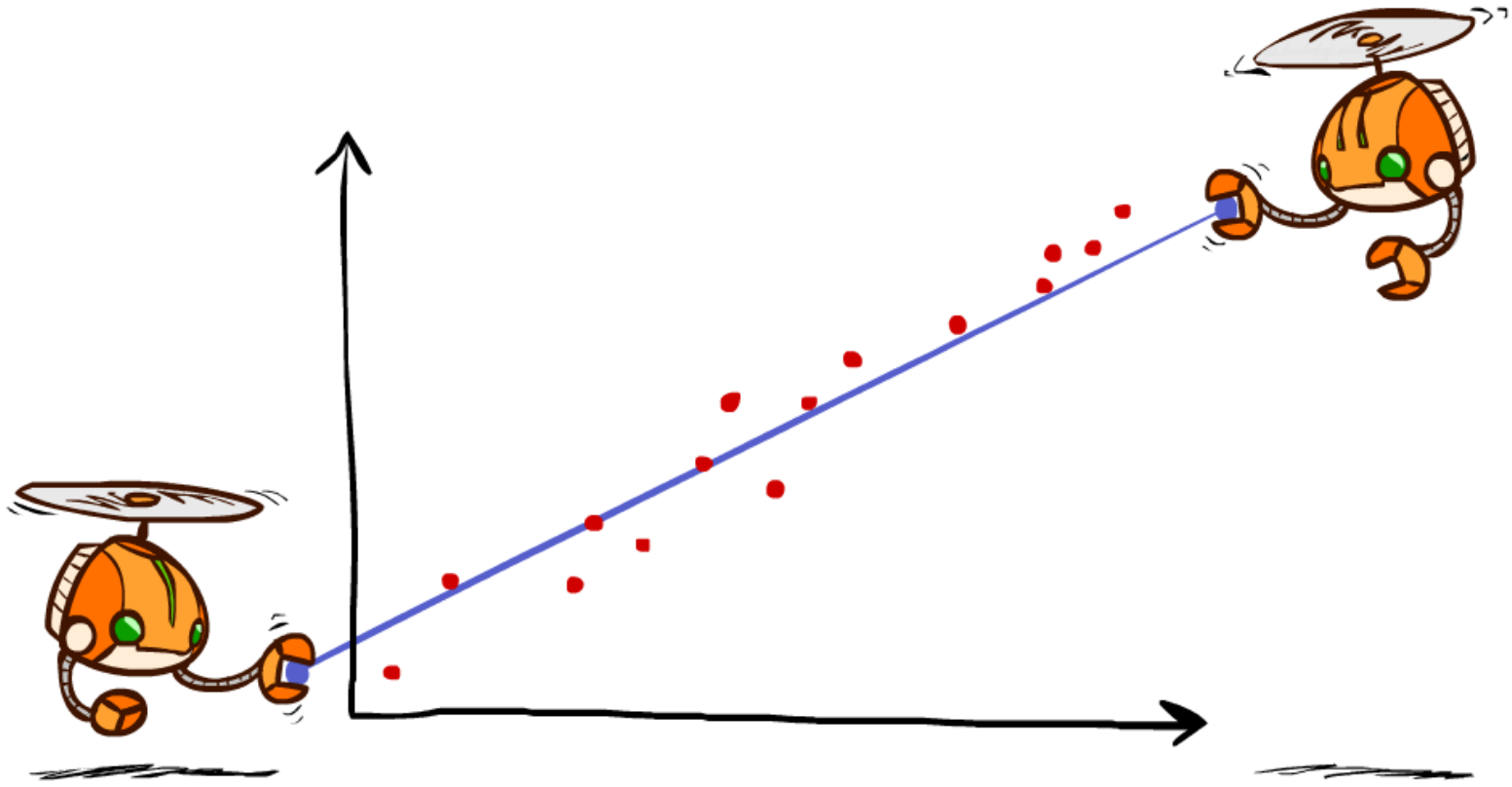
Take CS280 Deep Learning!

- Deep Learning
  - A large number of layers of neural networks
    - Ex. 1000 layers in ResNet
  - More complicated connections between layers
    - Ex. LSTM
  - Lots of new techniques and tricks
    - ReLU, Dropout, Batch Normalization, Adam, …
  - Big data
    - ImageNet (2009): 14 million images
    - NMT (a 2019 paper): 25 billion sentence pairs
  - GPU parallelization
- Performance: superior to human experts in some tasks

More classification methods

- Naive Bayes
- Perceptron / Neural networks
- Decision trees / Random forest
- Support Vector Machines
- Nearest neighbors
- Model ensembles: bagging, boosting, etc.
- ……

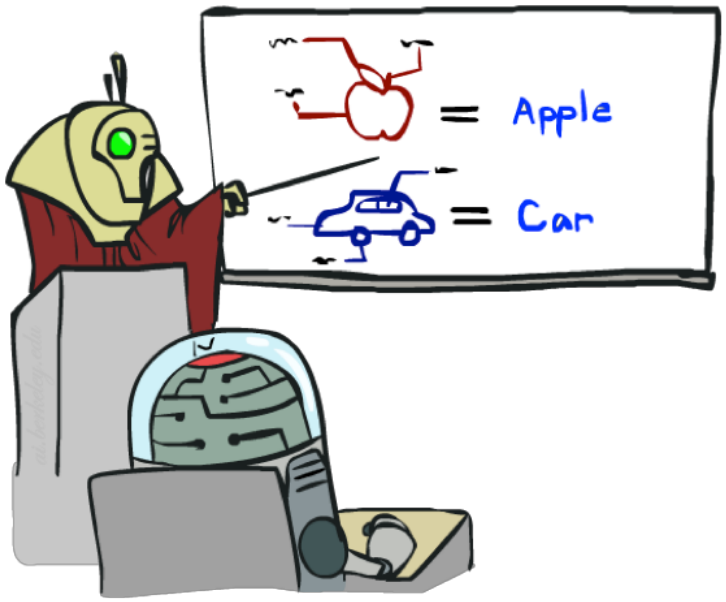
Regression 回归



Supervised learning

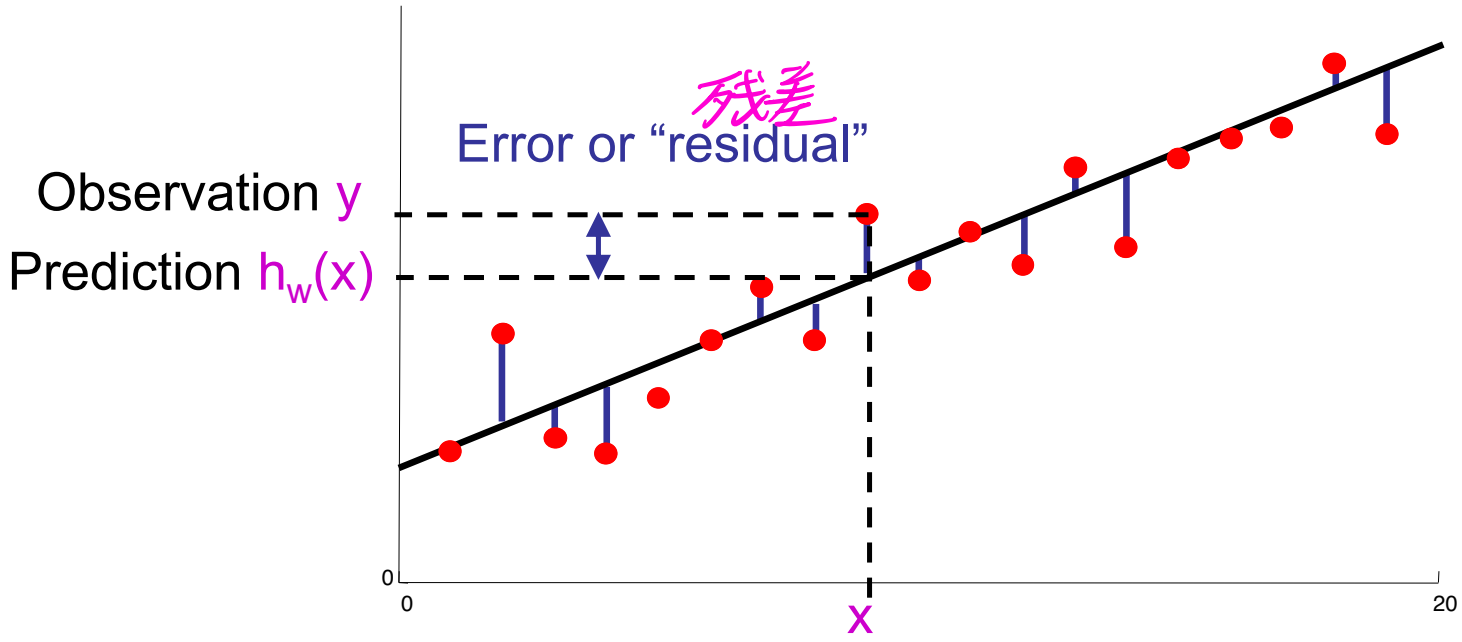
- To learn an unknown ***target function*** f 学习目标函数f
- Input: a ***training set*** of ***labeled examples*** (x_j, y_j) where $y_j = f(x_j)$
- Output: ***hypothesis*** h that is "close" to f

- Two types of supervised learning
  - Classification = learning f with discrete output value
  - Regression = learning f with real-valued output value



Linear Regression

Prediction: $h_w(x) = w_0 + w_1 x$



Error on one instance: $|y - h_w(x)|$

Least squares: Minimizing squared error

$h_w(x) = W_0 + W_1 x$

- L2 loss function: sum of squared errors over all examples

Least square Errors

$$L(\boldsymbol{w}) = \sum_i \left(y_i - h_w(\boldsymbol{x}_i)\right)^2 = \sum_i (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

自变量是w，我们要找到w^*去最小化 L2 loss

- We want the weights w* that minimize loss
- Analytical solution: at w* the derivative of loss w.r.t. each weight is zero
  - **X** is the data matrix (all the data, one example per row); **y** is the vector of labels
  - $\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

封闭解，求导为零的点

Least squares: Minimizing squared error

- $J(w) = \|Xw - y\|_2^2$

自变量：w → 对w求导 → $2X^T\|Xw-y\|$

$$\nabla J(w) = 2X^T(Xw - y)$$

自变量是w，$\vec{X}$和$\vec{Y}$是固定了的，通过改变w的值，使得J(w)的结果最小，也就是L_2对于Wi的误差最小。

$$X^TXw - X^Ty = 0$$

$$X^TXw = X^Ty$$

$$w = (X^TX)^+X^Ty$$

转置矩阵，伪逆

求逆

Regularized Regression

- Overfitting is also possible in regression
  - Extreme case: *n* features, *n* training examples
- Regularization can be used to alleviate overfitting

正则

- LASSO (Least Absolute Shrinkage and Selection Operator)

$$L(\boldsymbol{w}) = \sum_i (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 + \lambda \sum_k |w_k|$$

- Ridge Regression

$$L(\boldsymbol{w}) = \sum_i (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2 + \lambda \sum_k w_k^2$$

样本太少(导致overfitting)
解与非常依赖于样本

约束 $\sum |w_k| < C$
$\sum w_k^2 < C$
即C与λ有联系

W取值过大?

①

λ越大,越注重避免overfit

考虑绝对值:
容易得到稀疏解

$w_1+w_2=1$

L_2损失

使W也要小,约束W的取值

$|w_1|+|w_2|=1$

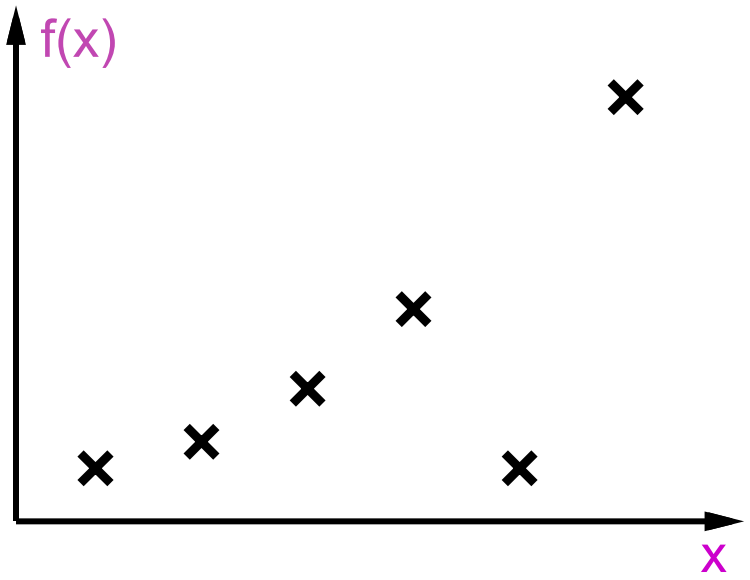
my opinion: W都落在边界上,而实际上菱形边长>圆形边长,相同个数W,菱形更稀疏.

Non-linear least squares

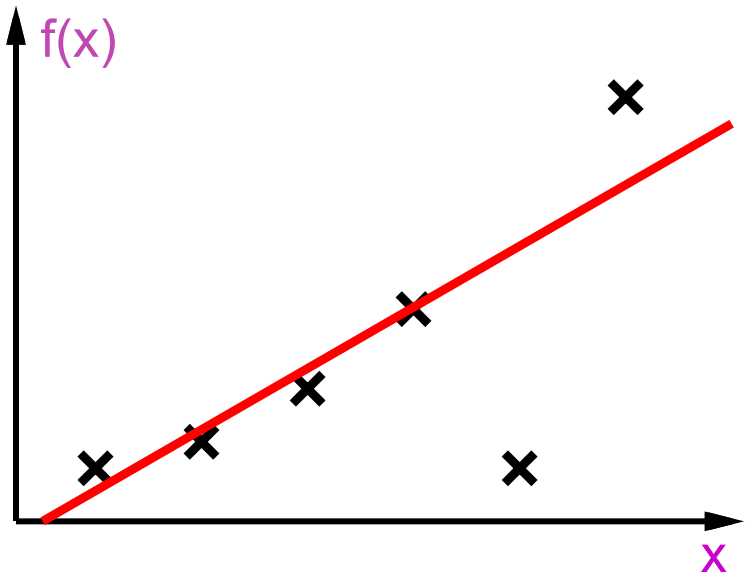
非线性回归无封闭解

- No closed-form solution in general
- Numerical algorithms are typically used
  - Choose initial values for the parameters and then refine the parameters iteratively
  - Gradient descent
  - Gauss–Newton method
  - Limited-memory BFGS
  - Derivative-free methods
  - etc.

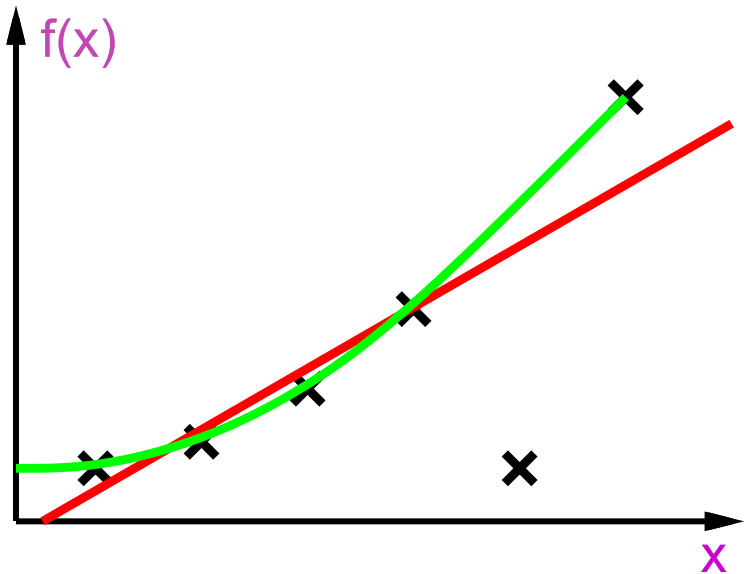
Overfitting in non-linear regression



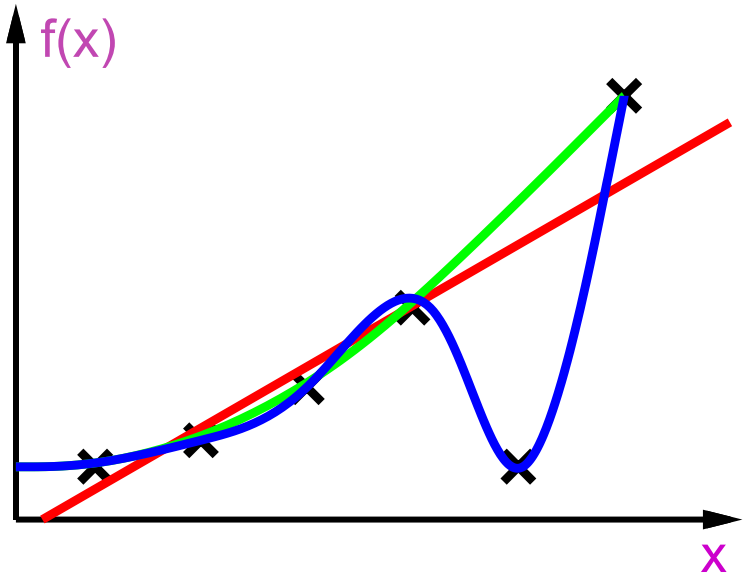
Overfitting in non-linear regression



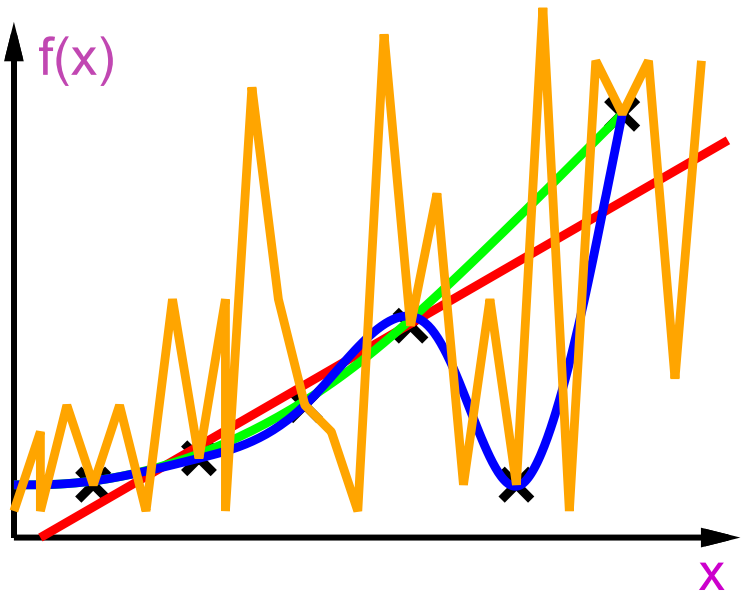
Overfitting in non-linear regression



Overfitting in non-linear regression



Overfitting in non-linear regression



Fit vs. complexity: a tradeoff

"*Ockham's razor*": prefer the *simplest* hypothesis consistent with the data

永远选择与数据一致但更简单的模型.

Summary

- Supervised learning:
  - Learning a function from labeled examples
- Classification: discrete-valued function
  - Naïve Bayes
  - Generalization and overfitting, smoothing
  - Perceptron
- Regression: real-valued function
  - Linear regression

损失函数之-----L1 loss和L2 loss和smooth L1 loss -----用于回归任务

1. L1 loss：

公式和求导公式：

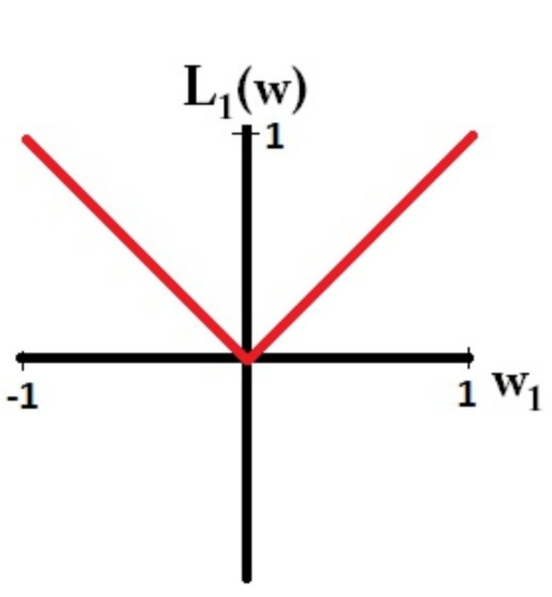
（带绝对值求导时，先去掉绝对值符号，再分情况求导）

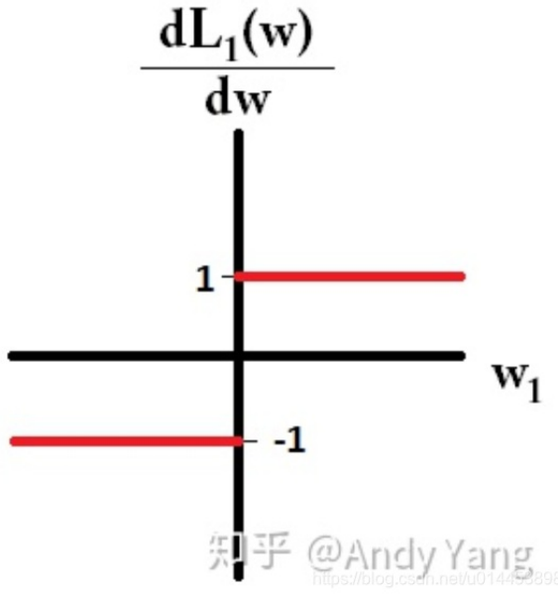
$$L_1 = |f(x) - Y|$$
$$L_1' = \pm f'(x)$$

一个batch的形式：

$$loss(x, y) = \frac{1}{n}\sum_{i=1}^{n}|y_i - f(x_i)|$$

L1 loss 图形和求导图形如下：





图的底部是预测值和label的差值。我们可以看到L1 loss的底部是尖的。底部是不存在导数的。而在其他地方，导数大小都是一样的。

优缺点：

优点：

1. L1 loss的鲁棒性(抗干扰性)比L2 loss强。概括起来就是L1对异常点不太敏感，而L2则会对异常点存在放大效果。因为L2将误差平方化，当误差大于1时，误会会放大很多，所以使用L2 loss的模型的误差会比使用L1 loss的模型对异常点更敏感。如果这个样本是一个异常值，模型就需要调整以适应单个的异常值，这会牺牲许多其它正常的样本，因为这些正常样本的误差比这单个的异常值的误差小。如果异常值对研究很重要，最小均方误差则是更好的选择。

缺点：

1. L1 loss 对 x(损失值)的导数为常数，在训练后期，x较小时，若学习率不变，损失函数会在稳定值附近波动，很难收敛到更高的精度。

2. L2 loss的稳定性比L1 loss好。概括起来就是对于新数据的调整，L1的变动很大，而L2的则整体变动不大。

2. L2 loss：

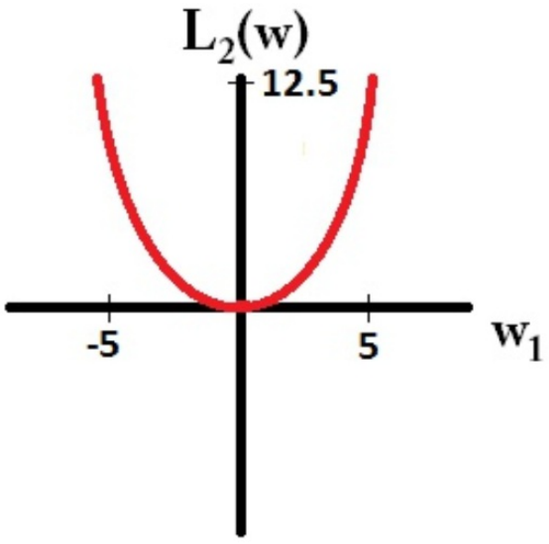
公式和求导公式：

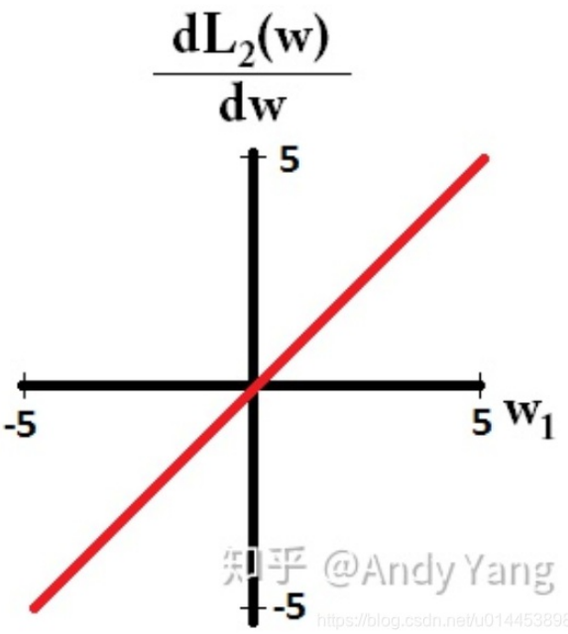
$$L_2 = |f(x) - Y|^2$$
$$L_2' = 2f'(x)(f(x) - Y)$$

一个batch的形式：

$$loss(x, y) = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

L2 loss图形和求导图像如下：

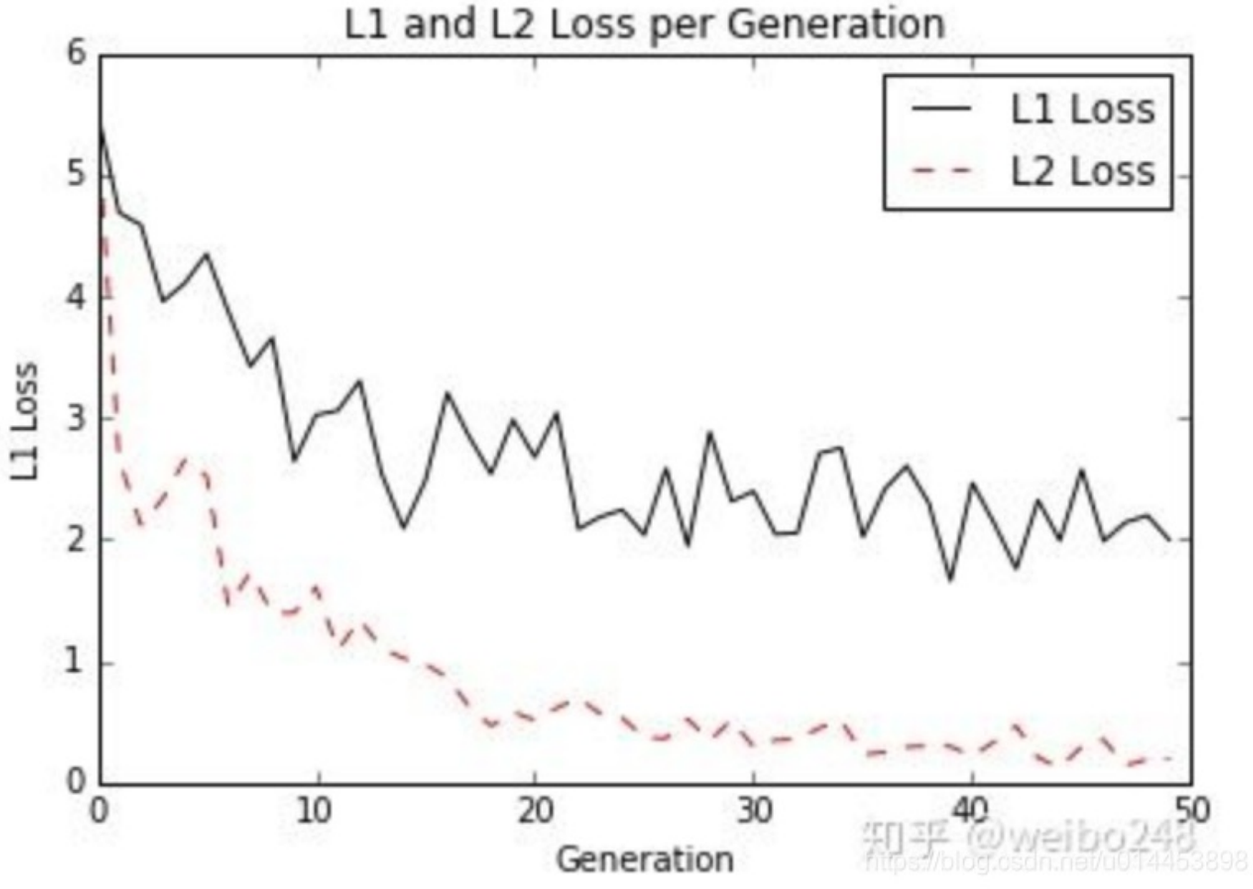




缺点：

从L2 loss的图像可以看到，图像(上图左边红线)的每一点的导数都不一样的，离最低点越远，梯度越大，使用梯度下降法求解的时候梯度很大，可能导致**梯度爆炸**。

L1 loss一般用于简单的模型，但由于神经网络一般是解决复杂的问题，所以很少用L1 loss，例如对于CNN网络，一般使用的是L2-loss，因为L2-loss的收敛速度比L1-loss要快。如下图：



3. Smooth L1 loss

计算公式与求导公式：

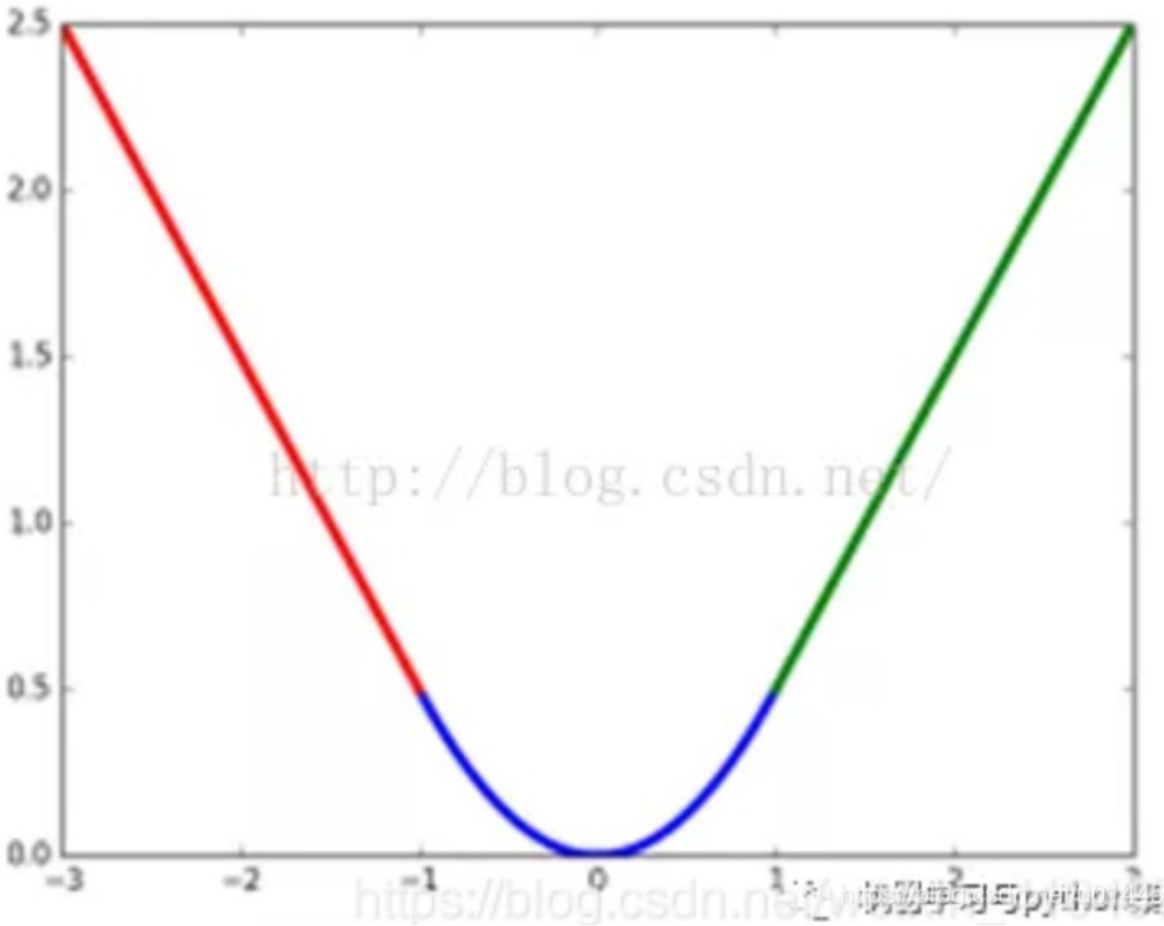
$$smooth_{L_1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

$$\frac{dsmooth_{L_1}(x)}{dx} = \begin{cases} x & |x| < 1 \\ \pm 1 & otherwise \end{cases}$$

一个batch(n个)数据时：

$$loss(x, y) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} .5 * (y_i - f(x_i))^2, & if\ |y_i - f(x_i)| < 1 \\ |y_i - f(x_i)| - 0.5, & otherwise \end{cases}$$

smooth L1 loss的图如下：



仔细观察可以看到，当预测值和ground truth差别较小的时候（绝对值差小于1），其实使用的是L2 Loss，当绝对值差小于1时，由于L2会对误差进行平方，因此会得到更小的损失，有利于模型收敛。而当差别大的时候，是L1 Loss的平移，因此相比于L2损失函数，其对离群点（指的是距离中心较远的点）、异常值（outlier）不敏感，可控制梯度的量级使训练时不容易跑飞。。**SooothL1Loss其实是L2Loss和L1Loss的结合**，它同时拥有L2 Loss和L1 Loss的部分优点。