Announcements

- New schedule posted
- Assignment 2 due tomorrow at midnight
- Midterm on Tuesday (study! bring 1 page of notes)
- Project instructions, Assignment 3 posted on Tuesday
- Two interesting talks:
  - 1/29 @ 11A in E2 475: Rama Akkiraju, "We know your personality, writing tone, emotions and how you make decisions. What next?"
  - 2/1 @ 11A in E2 180: Jeff Ullman, "Computing Marginals using MapReduce"

Statistical Learning for Classification

Review of Basic Prob. Concepts

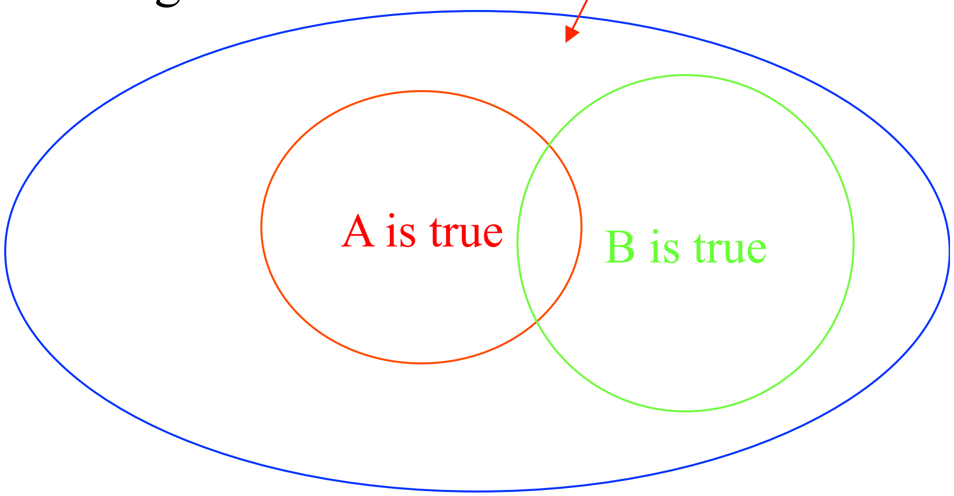
- Marginal probability P(A): “the fraction of possible world in which A is true”
  - Examples

  A = Your paper will be accepted by SIGKDD 2015

  A = It rains in Santa Cruz

  A = A document contains the word “mining”
- Joint probability P(A, B)
- Conditional probability
  - P(A|B)=P(A, B)/P(B)
- Bayes’ rule
  - P(A|B)=P(B|A)P(A)/P(B)

Event space of all possible worlds. The area is 1.



Classification as Supervised Statistical Learning

- Supervised learning
  - Given: input and output variables pairs (training data) $\{(x_1,y_1)(x_2,y_2)\ldots(x_N,y_N)\}$
  - Learning: infer a function *f(X)* from the training data
  - Prediction: predict future outcomes y=*f(x)*

$$\text{classfication}: \quad R^{|V|} \rightarrow \{0,1\}$$

X

Prediction of category c

$$\text{regression}: \quad R^{|V|} \rightarrow R$$

X

Prediction of real value output

f

y

Major Steps for Supervised Learning

- Gathering a training set (input objects and corresponding outputs) from human or from other measurements
- Determine the input feature of the learned function (What's X)
  - Controlled vocabulary? Bag of words? Title only? Stemming? Stopping? Phrases? Linguistic Features? Meta data? Other contextual information?
  - Typically, the input object is transformed into a feature vector
  - This step influence the final performance of the system greatly
- Determine the functional form of the learned algorithm
  - Logistic regression? Neural network? Support Vector Machines?
- Determine the corresponding learning algorithm (ML or MAP or else)
- Learn: run the learning algorithm on the gathered training set
  - Optional: adjust the parameter via cross validation
- Test the performance on a test set

Two Statistical Learning Approaches

- Generative models
  - Model the joint probabilistic distribution p(c, X), and derive the conditional probability

$$P(c_j \mid X) = \frac{P(X \mid c_j) P(c_j)}{\sum_i P(X \mid c_i) P(c_i)}$$

  - Examples: Naive Bayes
- Discriminative models
  - Model the conditional probability P(c|X) directly
  - Examples: decision tree, neural networks, support vector machines, boosting or bagging, regression (linear, polynomial …), k nearest neighbor

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - $P(A_1, A_2, \ldots, A_n \mid C_j) = P(A_1 \mid C_j)\, P(A_2 \mid C_j) \ldots P(A_n \mid C_j)$
  - Can estimate $P(A_i \mid C_j)$ for all A_i and C_j.
  - New point is classified to C_j if $P(C_j)\, \Pi\, P(A_i \mid C_j)$ is maximal.

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10, P(Yes) = 3/10
- For discrete attributes, one approach is:

$$P(A_i \mid C_k) = |A_{ik}| / N_c$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

P(Status=Married|No) = 4/7
P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption

k

  - Two-way split: $(A < v)$ or $(A > v)$
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal (or other) distribution
    - Use data to estimate parameters of the distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i,c_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record: $X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No) × P(Married| Class=No) × P(Income=120K| Class=No) = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes) × P(Married| Class=Yes) × P(Income=120K| Class=Yes) = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)
=> Class = No

Exercise: Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Exercise: Naïve Bayes Classifier

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(A|M) = \frac{6}{7} \times \frac{6}{7} \times \frac{2}{7} \times \frac{2}{7} = 0.06$$

$$P(A|N) = \frac{1}{13} \times \frac{10}{13} \times \frac{3}{13} \times \frac{4}{13} = 0.0042$$

$$P(A|M)P(M) = 0.06 \times \frac{7}{20} = 0.021$$

$$P(A|N)P(N) = 0.004 \times \frac{13}{20} = 0.0027$$

P(A|M)P(M) > P(A|N)P(N)

=> Mammals

Naïve Bayes Classifier

- If one of the conditional probability is zero, then the entire expression becomes zero
- Solution: probability estimation with smoothing:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m-estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

c: number of classes

p: prior probability

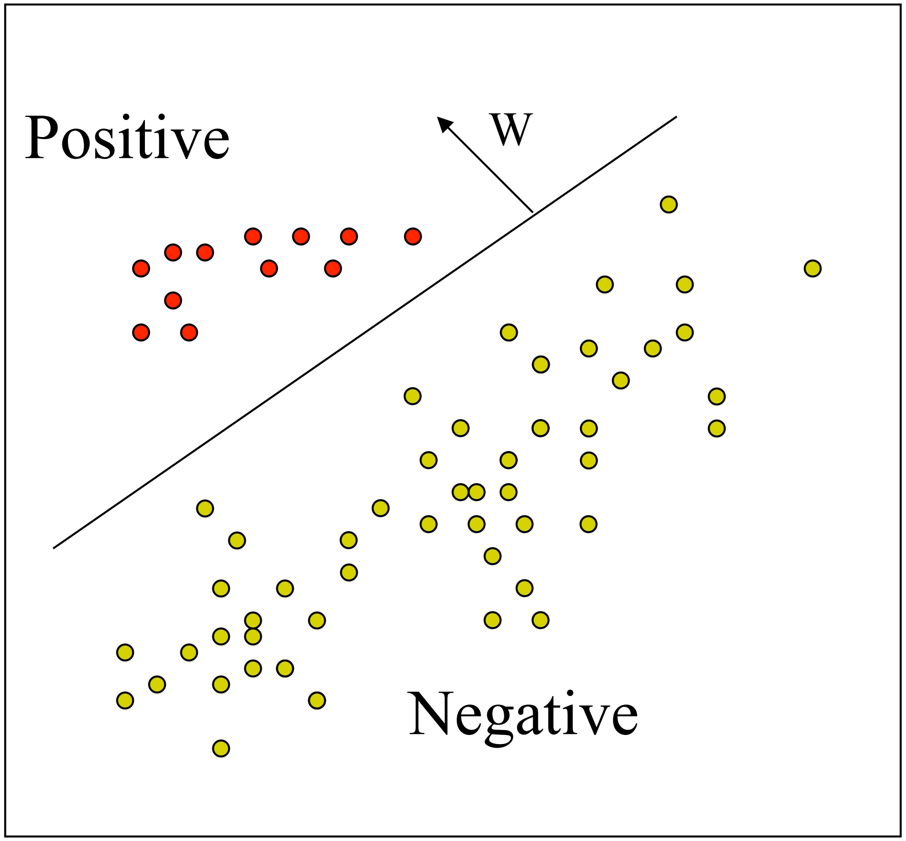
m: parameter

Naïve Bayes Classifier

- Pros:
  - Clear theoretical foundation
  - Relatively effective, robust to isolated noise points
  - Very simple
  - Handle missing values by ignoring the instance during probability estimate calculations
  - Fast: handles 10,000 attributes easily
- Cons
  - Wrong assumptions: Independence assumptions, models assumption (Multi-Variate Bernoulli, Multinomial, Gaussian etc.), One class per document assumption
  - Classification accuracy is usually worse than many other methods, such as logistic regression, linear regression or support vector machines
  - Bad probabilistic estimation of P(c|x)

Discriminative Models

- Focusing on estimating the decision boundary between classes or $P(y|\mathbf{X})$
- No explicit assumption on how the documents are generated
- For text classification task, usually the decision boundary is a linear separator
  - Assign a document to the positive class if $h(X)=W^TX>0$



Document space

Linear models: linear regression

- Work most naturally with numeric attributes
- Standard technique for numeric prediction
  - Outcome is linear combination of attributes

$$x = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_k x_k$$

- Weights are calculated from the training data

- Predicted value for first training instance $\mathbf{x}^{(1)}$

$$y^{(1)} = w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)} + \ldots + w_k^{(1)} x_k = w_0 + \sum_{j=1}^{k} w_j x_j^{(1)}$$

Minimizing the squared error

- Choose *k* +1 coefficients to minimize the squared error on the training data
- Squared error:

$$\sum_{i=1}^{n}\left(y^{(i)} - w_0 + \sum_{j=1}^{k} w_j x_j^{(i)}\right)^2$$

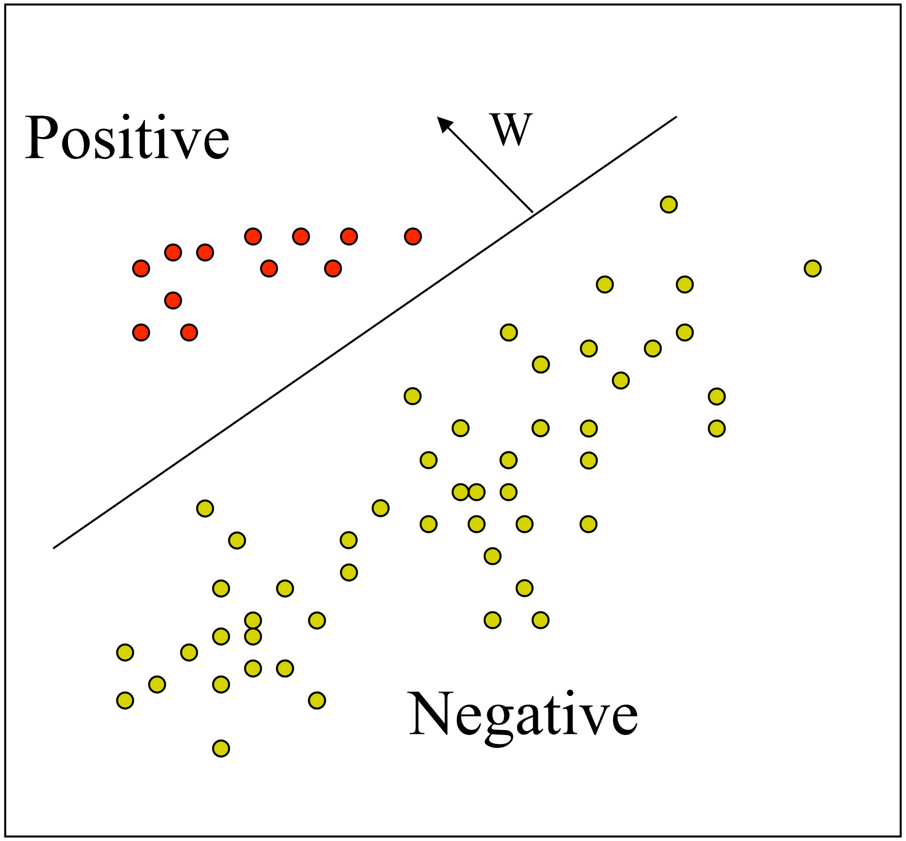
- Derive coefficients using standard matrix operations
- Can be done if there are more instances than attributes (roughly speaking)
- Minimizing the *absolute error* is more difficult

Classification

- *Any* regression technique can be used for classification
  - Training: perform a regression for each class, setting the output to 1 for training instances that belong to class, and 0 for those that don’t
  - Prediction: predict class corresponding to model with largest output value (*membership value*)
- For linear regression this is known as *multi-response linear regression*
- Problem: membership values are not in [0,1] range, so aren't proper probability estimates

Discriminative Models

- Focusing on estimating the decision boundary between classes or P(*y*|**X**)
- No explicit assumption on how the documents are generated
- For text classification task, usually the decision boundary is a linear separator
  - Assign a document to the positive class if $h(X)=W^{T}X>0$



Document space

Linear models: logistic regression

- Builds a linear model for a transformed target variable
- Assume we have two classes
- Logistic regression replaces the target

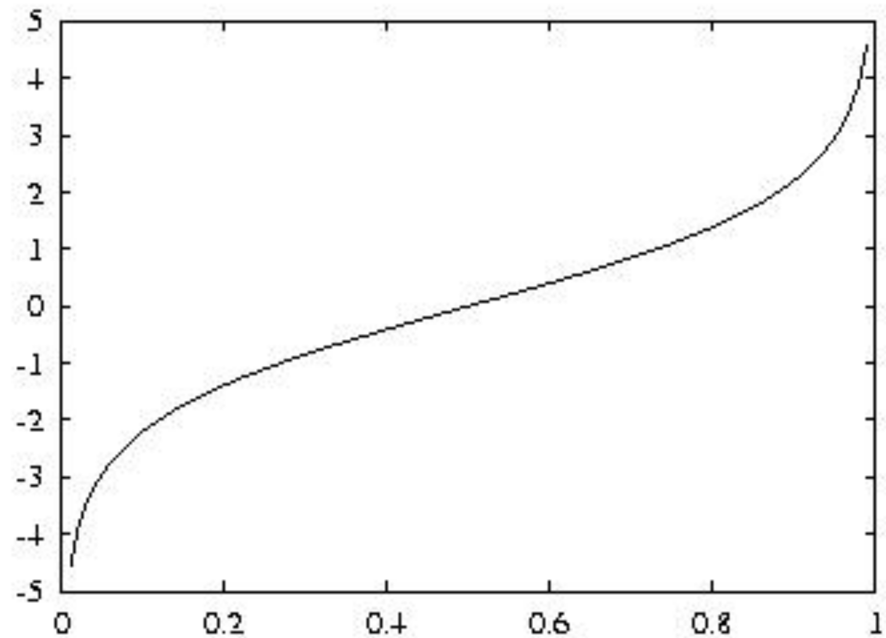
$$P(1|x_1, x_2, \ldots, x_k)$$

by this target

$$\log\left(\frac{P(1|x_1, x_2, \ldots, x_k)}{1 - P(1|x_1, x_2, \ldots, x_k)}\right)$$

- *Logit transformation* maps [0,1] to (-∞ , +∞)

Logit transformation

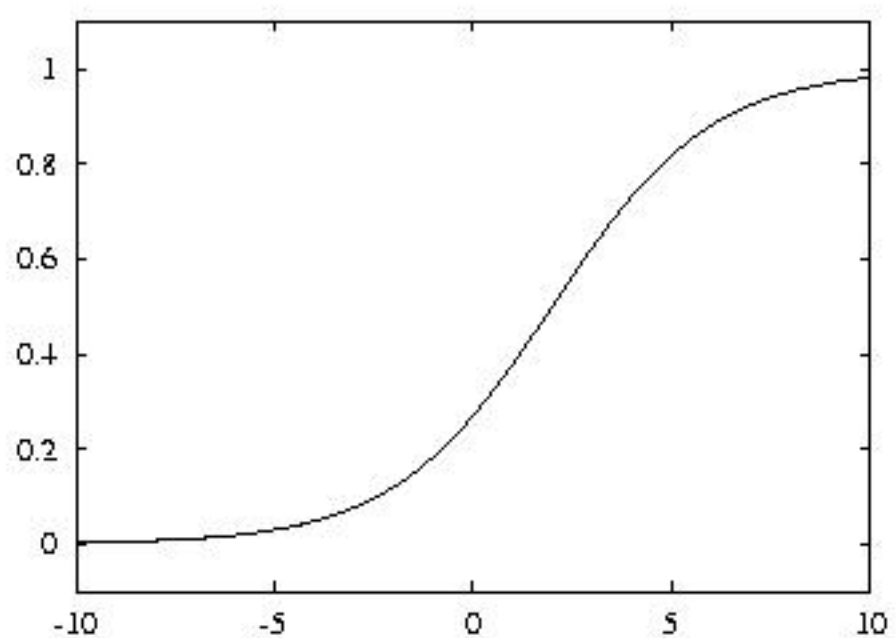


- Resulting model:

$$P(1|x_1, x_2, ..., x_k) = \frac{1}{1 + e^{-w_0 - w_1 x_1 - ... - w_k x_k}}$$

Example logistic regression model

- Model with $w_0 = 0.5$ and $w_1 = 1$:



- Parameters are found from training data using *maximum likelihood*

Linear models are hyperplanes

- Decision boundary for two-class logistic regression is where probability equals 0.5:

$$P(1|x_1, x_2, ..., x_k) = 1/(1 + \exp(-w_0 - w_1 x_1 - ... - w_k x_k))$$

which occurs when $-w_0 - w_1 x_1 - ... - w_k x_k = 0$

- Thus logistic regression can only separate data that can be separated by a hyperplane

Maximum likelihood

- Aim: maximize probability of training data wrt parameters
- Can use logarithms of probabilities and maximize *log-likelihood* of model:

$$\sum_{i=1}^{n} y^{(i)}\log(P(1|x_1^{(i)}, x_2^{(i)}, ..., x_k^{(i)}) \ +$$

$$(1 - y^{(i)})\log(1 - P(1|x_1^{(i)}, x_2^{(i)}, ..., x_k^{(i)})$$

  where the $y^{(i)}$ are either 0 or 1
- Weights w_i need to be chosen to maximize log-likelihood (relatively simple method: *iteratively re-weighted least squares*)

How to Learn?

- Given evidence (data) E, find hypothesis (model) h
- Maximum likelihood (ML) estimation

$$h_{ML} = \arg\max_{h} \boxed{P(E \mid h_i)}$$

Data likelihood

- Maximum a posteriori (MAP) estimation

$$h_{MAP} = \arg\max_{h} \boxed{P(h \mid E)}$$

Posterior probability of hypothesis

Bayes' rule

$$= \arg\max_{h} \frac{P(E \mid h)P(h)}{P(E)}$$

$$= \arg\max_{h} P(E \mid h)P(h)$$

Data likelihood

Prior probability of hypothesis

Learning Logistic Regression Model

- Maximum likelihood estimation

$$W_{ML} = \arg\max_{w} \prod_{i=1}^{t} p(y_i \mid x_i, W) = \arg\max_{w} \sum_{i=1}^{t} \log(p(y_i \mid x_i, W))$$

likelihood of training data

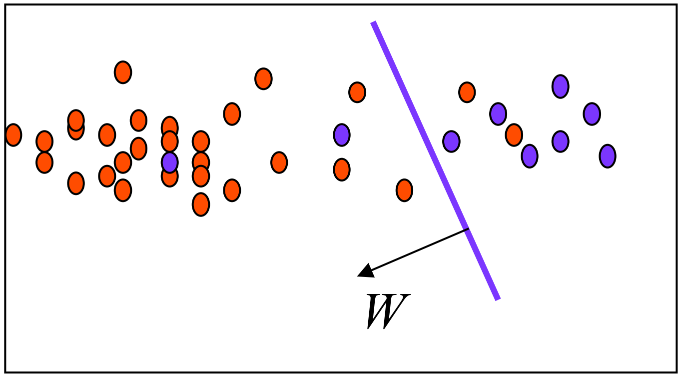
- Maximum a posteriori (MAP) estimation

$$W_{MAP} = \arg\max_{w} \prod_{i=1}^{t} p(y_i \mid x_i, W) P(W)$$

$$= \arg\max_{w} \sum_{i=1}^{t} \log(p(y_i \mid x_i, W)) + \log(P(W))$$

likelihood of training data

Usually a Gaussian prior



Document space (N)

Logistic Regression Classifier

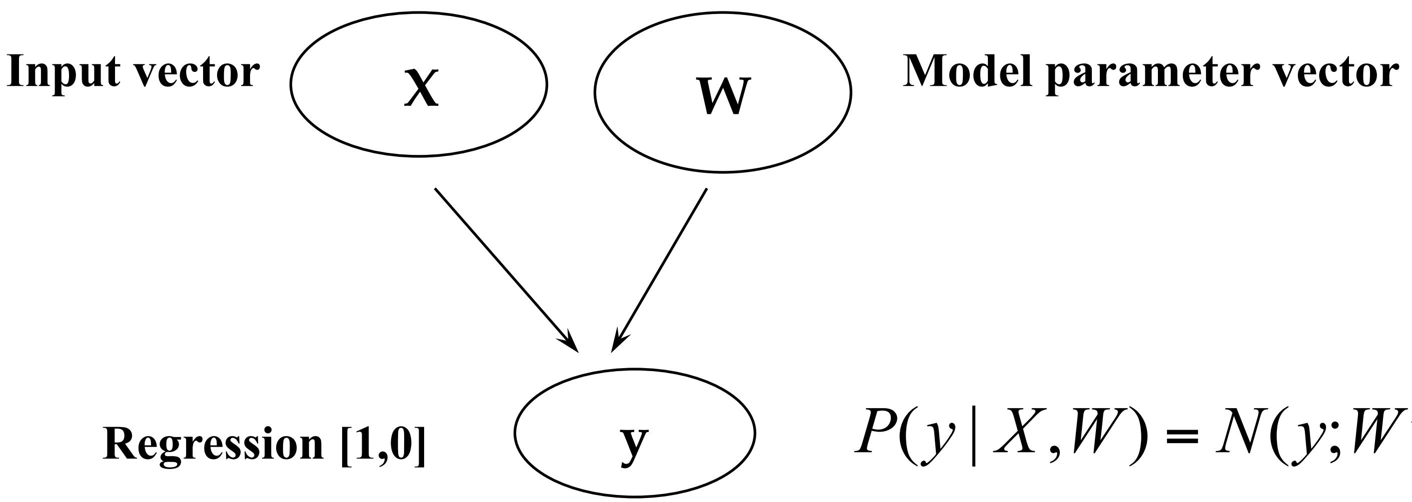
- If the goal is to minimize classification error, classify X to the class if:

$$P(y = yes \mid X, W) = \frac{1}{1 + e^{-W^T X}} > 0.5 \Leftrightarrow W^T X > 0$$

linear separator

Linear Regression Model

- Modeling the conditional probability p(y|X) as a Normal distribution



$$P(y \mid X, W) = N(y; W^T X, \sigma^2)$$

Learning Linear Regression Model

- Maximum likelihood estimation

sum square error

$$W_{ML} = \arg\max_{w} \sum_{i=1}^{t} \log(p(y_i \mid x_i, W)) = \arg\max_{w} -\sum_{i=1}^{t} (y_i - W^T x_i)^2$$

- Maximum a posteriori (MAP) estimation

Log likelihood of data

Usually a Gaussian prior

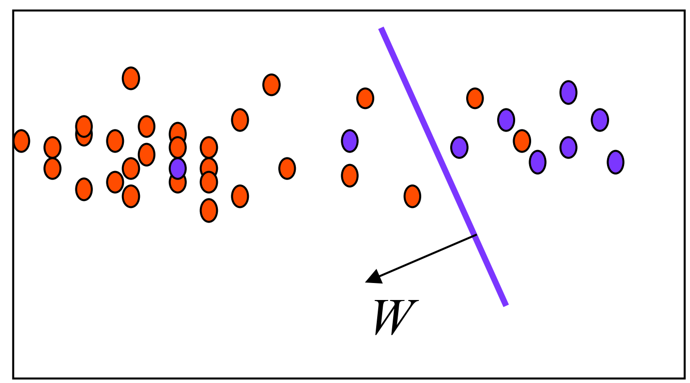
$$W_{MAP} =$$

$$\arg\max_{w} \sum_{i=1}^{t} \log(p(y_i \mid x_i, W)) + \log P(W)$$

$$= \arg\max_{w} -\sum_{i=1}^{t} (y_i - W^T x_i)^2 - \lambda(W - U)^2$$

Sum square error on training data

Deviation from the prior mean

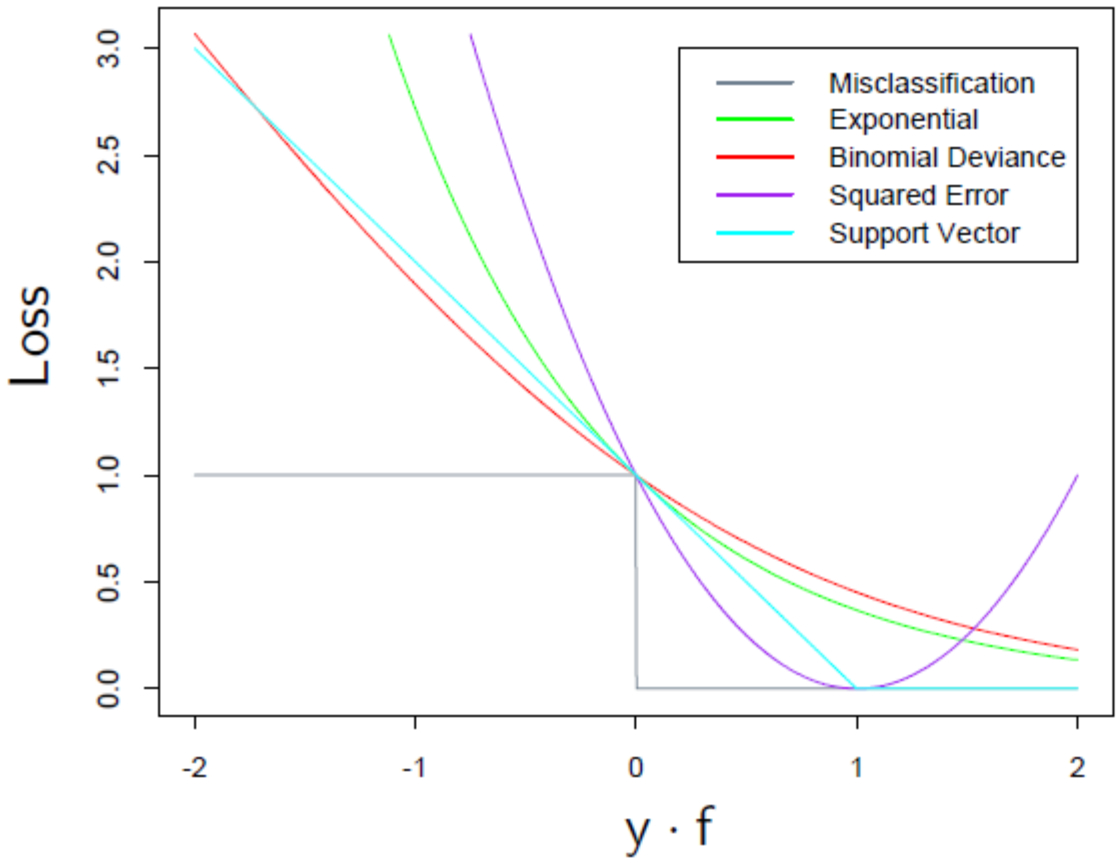


Document space (N)

The Benefit of Using Prior P(W)

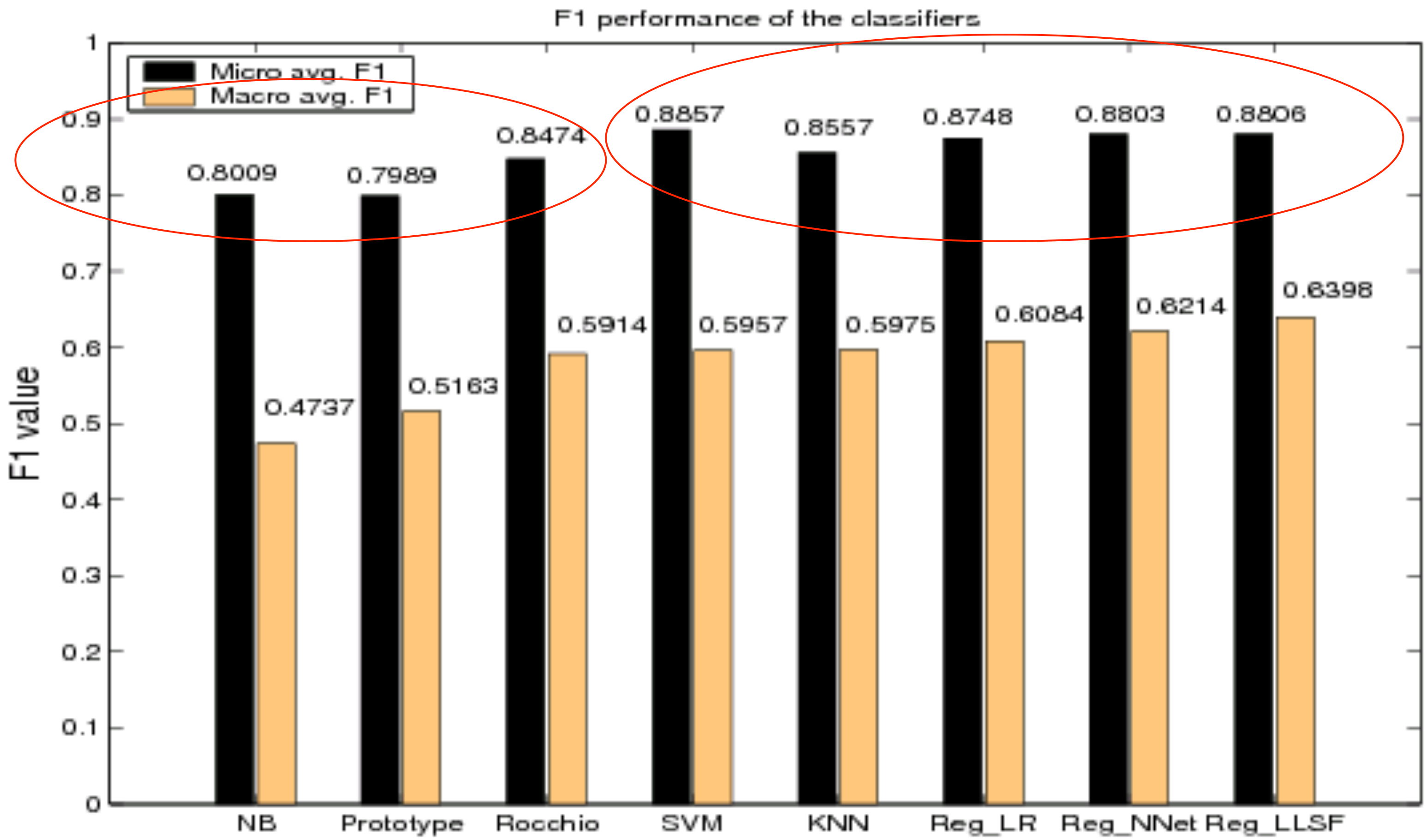
- Controlling model complexity: avoid overfitting
  - Similar to having a regularizer
- Avoiding the pain of zero probability
- Integrating expert/prior knowledge
- Integrating two classification algorithms
- Transfer learning
  - Using one task to help another task

Loss Function for Different Discriminative Classifiers



From Trevor Hastie

Some Empirical Performance



Reuters-21578

From Li and Yang SIGIR03

Comparison of Generative Models and Discriminative Models

	Generative Models (Naïve Bayes)	Discriminative Models (LR, LR)
Focus	P(X, y)	P(y|X)
Training efficiency	☺	☹
Effectiveness	OK	Good

Improving Simple Classifiers

- *Bagging:* Fit many large *t*rees to bootstrap-resampled versions of the training data, and classify by majority vote. (Variance reduction)
- *Boosting:* Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote. (Bias reduction)

Multiple classes

- Can perform logistic regression independently for each class
  (like multi-response linear regression)
- Problem: probability estimates for different classes won't sum to one
- Better: train coupled models by maximizing likelihood over all classes
- Alternative that often works well in practice: *pairwise classification*

Multi-Class Classification

$$\text{classfication}: \quad R^{|V|} \to \{0,1\}^K$$

$$\text{regression}: \quad R^{|V|} \to R^K$$

V : the set of vocabularies

K : the number of classes

$X_j \in R^{|V|}$: a document

	c_1	c_2	…	c_j	…	c_K
X_1	0	1	…	1		0
X_2	1	0		0		0
…						
X_i	1	0		1		1
…						
X_M						

Pairwise classification

- Idea: build model for each pair of classes, using only training data from those classes
- Problem? Have to solve $k(k-1)/2$ classification problems for k-class problem
- Turns out not to be a problem in many cases because training sets become small:
  - Assume data evenly distributed, i.e. $2n/k$ per learning problem for n instances in total
  - Suppose learning algorithm is linear in n
  - Then runtime of pairwise classification is proportional to $(k(k-1)/2)\times(2n/k) = (k-1)n$

Multiclass regression: hyperplane

- Multi-response linear regression has the same problem. Class A is assigned if:

$$w_0^{(c=A)} + w_1^{(c=A)}x_1 + w_2^{(c=A)}x_2 + ... + w_k^{(c=A)}x_k >$$
$$w_0^{(c=B)} + w_1^{(c=B)}x_1 + w_2^{(c=B)}x_2 + ... + w_k^{(c=B)}x_k$$

- Which happens when:

$$(w_0^{(c=A)} - w_0^{(c=B)}) + (w_1^{(c=A)} - w_1^{(c=B)})x_1 + (w_2^{(c=A)} - w_2^{(c=B)})x_2 +$$
$$... + (w_k^{(c=A)} - w_k^{(c=B)})x_k > 0$$

Multi-Class Classification Approaches

- Learn one binary classifier for each class
  - Assign a document to all classes that the corresponding classifiers says “yes”; or
  - Assign a document to the “best” class
- Many against many
  - Learn multiple classifiers
  - Each classifier assign a document to a set of classes
  - Assign a document to the class with the biggest votes from classifiers

Multinomial Logistic Regression

$$\Pr(Y_i = c) = \frac{e^{\boldsymbol\beta_c \cdot \mathbf{X}_i}}{\sum_{k=1}^{K} e^{\boldsymbol\beta_k \cdot \mathbf{X}_i}}$$

Because probabilities must sum to 1, one of β_c is completely determined once all the rest are known. Thus one can also use:

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol\beta'_1 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol\beta'_k \cdot \mathbf{X}_i}}$$

$$\cdots\cdots$$

$$\Pr(Y_i = K-1) = \frac{e^{\boldsymbol\beta'_{K-1} \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol\beta'_k \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = K) = \frac{1}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol\beta'_k \cdot \mathbf{X}_i}}$$

Linear models: the perceptron

- Don't actually need probability estimates if all we want to do is classification
- Different approach: learn separating hyperplane
- Assumption: data is *linearly separable*
- Algorithm for learning separating hyperplane: *perceptron learning rule*
- Hyperplane:

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots w_k x_k = 0$$

- If sum is greater than zero we predict the first class, otherwise the second class

The algorithm

```
Set all weights to zero
Until all instances in the training data are classified correctly
  For each instance I in the training data
    If I is classified incorrectly by the perceptron
      If I belongs to the first class add it to the weight vector
      else subtract it from the weight vector
```

- Why does this work?

  Consider situation where instance *a* pertaining to the first class has been added:

$$(w_0 + 1) + (w_1 + x_1)x_1 + (w_2 + x_2)x_2 + ...(w_k + x_k)x_k$$

  This means output for *a* has increased by:

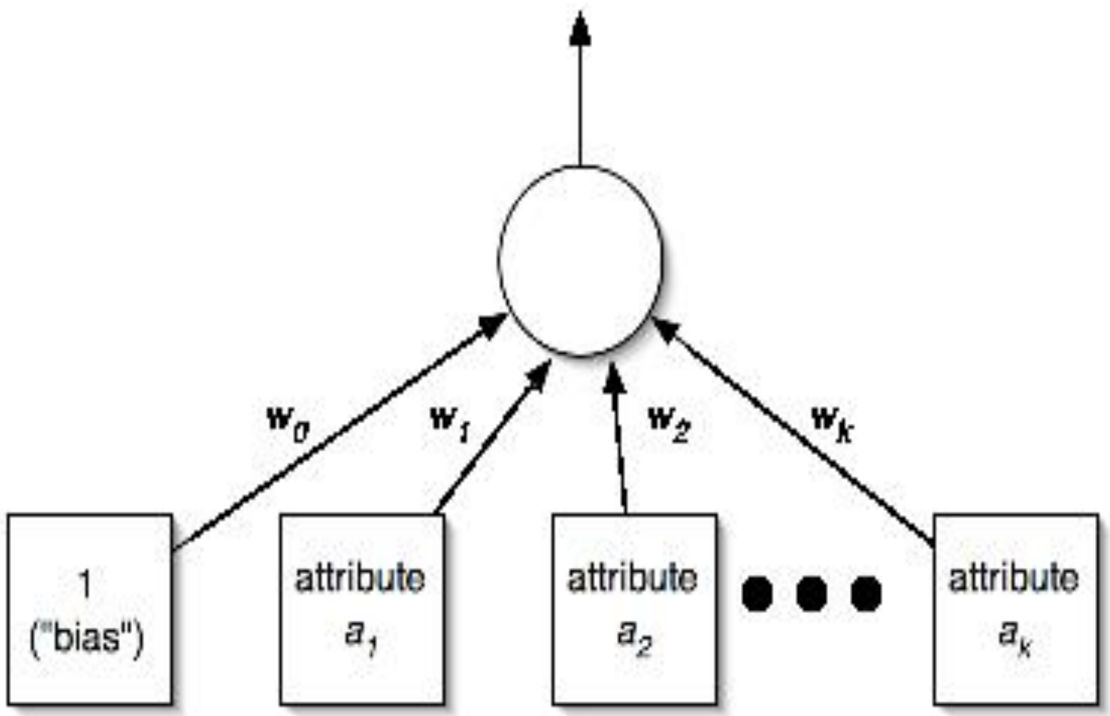
$$1 + x_1x_1 + x_2x_2 + ...x_kx_k$$

  This number is always positive, thus the hyperplane has moved into the correct direction (and we can show output decreases for instances of other class)

Perceptron as a neural network

Output layer

Input layer



Linear models: Winnow

- Another *mistake-driven* algorithm for finding a separating hyperplane
  - Assumes binary data (i.e. attribute values are either zero or one)
- Difference: *multiplicative* updates instead of *additive* updates
  - Weights are multiplied by a user-specified parameter $\alpha > 1$ (or its inverse)
- Another difference: user-specified threshold parameter θ
  - Predict first class if

$$w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_k x_k > \theta$$

The algorithm

```
while some instances are misclassified
  for each instance a in the training data
    classify a using the current weights
    if the predicted class is incorrect
      if a belongs to the first class
        for each a_i that is 1, multiply w_i by alpha
        (if a_i is 0, leave w_i unchanged)
      otherwise
        for each a_i that is 1, divide w_i by alpha
        (if a_i is 0, leave w_i unchanged)
```

- Winnow is very effective in homing in on relevant features (*it is attribute efficient*)
- Can also be used in an on-line setting in which new instances arrive continuously (like perceptron)

Balanced Winnow

- Winnow doesn't allow negative weights and this can be a drawback
- *Balanced Winnow* maintains two weight vectors, one for each class: Instance is classified as belonging to the first class (of two classes) if:

$$(w_0^+ - w_0^-) + (w_1^+ - w_1^-)x_1 + (w_2^+ - w_2^-)x_2 + \ldots + (w_k^+ - w_k^-)x_k > \theta$$

```
while some instances are misclassified
  for each instance a in the training data
    classify a using the current weights
    if the predicted class is incorrect
      if a belongs to the first class
        for each a_i that is 1, multiply w_i^+ by alpha and divide w_i^- by alpha
          (if a_i is 0, leave w_i^+ and w_i^- unchanged)
      otherwise
        for each a_i that is 1, multiply w_i^- by alpha and divide w_i^+ by alpha
          (if a_i is 0, leave w_i^+ and w_i^- unchanged)
```

Other Practical Concerns

- Controlling model complexity
  - Feature selection, smoothing (coefficient shrinkage): Ridge regression, lasso regression, Bayesian prior
- Over fitting (cross-validation, leave one out)
- Cost sensitive learning
  - Classify a ham as spam is more costly than the other way around
- Unbalanced samples and rare classes: 0.01% positive vs. 99.99% negative samples
- Biased samples
  - User only provides feedback on documents she reads
  - While she may not not read randomly
- Noisy label