Classification Methods

- Approximating Bayes Decision Rule: (model the likelihood)
  - Linear Discriminant Analysis/ QDA
  - Fisher's linear discriminant
  - **Naïve Bayes**
  - **Bayesian Belief Networks**
- Approximating Bayes Decision Rule: (model the posterior)
  - **Logistic Regression**
  - **Feedforward neural networks, including deep learning**
  - K-Nearest Neighbor
- Focus just on Class Boundaries:
  - Decision trees
  - **Support Vector Machines**

Quiz (Bayes Rule)

- Suppose 0.01% of Austin's population have cancer. A new test for cancer shows positive 90% of the time when a person actually has cancer, and correctly indicates "negative" 95% of the time when run on someone who do not have cancer.

This test is conducted on an Austinite and the results come out positive.

- What is the probability that this person actually has cancer?

Revisiting Bayes Decision Rule

- Let input **x** have d features or attributes $(x_1, x_2,\ldots,x_d)$;
  C is a random variable over class labels.

- **Bayes Decision rule:** Choose value of C that
  maximizes $P(x_1, x_2, \ldots, x_d|C)\, P(C)$

- Problem: how to estimate $P(x_1, x_2,\ldots,x_d\,|C)$ for each class?
  - Especially in high dimensions, interacting variables?

(Conditional) Independence

Naïve Bayes Approach

- Conditional Independence:
  - X is cond. Indep of Y given Z if P (X|Y,Z) = P(X|Z)
    - X, Y, Z could be sets of variables too

Naïve Bayes:

1. Assume independence among attributes x_i when class is given: ("independence of attributes conditioned on class variable").
   - $P(x_1, x_2, \ldots, x_d | C_j) = P(x_1 | C_j)\, P(x_2 | C_j) \ldots P(x_d | C_j) = \Pi_i\, P(x_i | C_j)$

   - Note: Conditional independence not equal to attribute independence

2. Estimate probabilities directly from data.

Estimating Probabilities from Data (Discrete Attributes)

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

(Example from TSK)

- Class: $P(C) = N_c/N$
  - e.g., $P(\text{No}) = 7/10$, $P(\text{Yes}) = 3/10$

- For discrete attributes:

  $P(x_i = v \mid C_k)$ = fraction of examples of class k for which attribute x_i takes value v.
  - Examples:

$P(\text{Status=Married}|\text{No}) = 4/7$

$P(\text{Refund=Yes}|\text{Yes})=0$

Naïve Bayes Example

- Worked out in Backup Slides

Naïve Bayes (Comments)

- Independence assumption often does not hold
  - Poorer estimate of P(C|x), often unrealistically close to 0 or 1
    - but still may pick the right "max"!!
  - If too restrictive, use other techniques such as Bayesian Belief Networks (BBN)
- Somewhat robust to isolated noise points, and irrelevant attributes
- Tries to finesse "curse of dimensionality"
- Most popular with binary or small cardinality categorical attributes
- **Requires only single scan of data; also streaming version is trivial.**

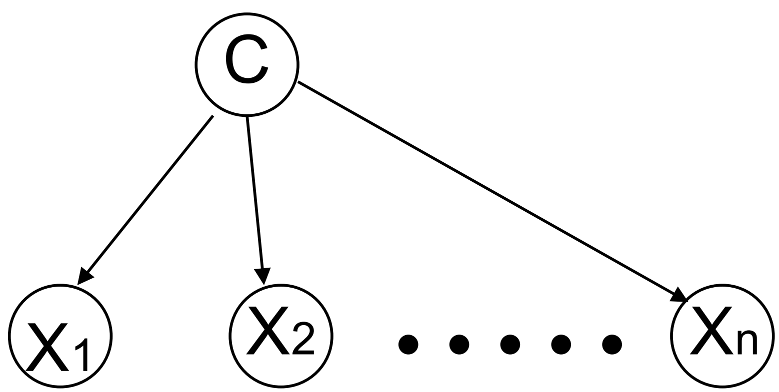
- Notable "Success": Text (bag-of-words representation + multinomial model per class).

Graphical Models

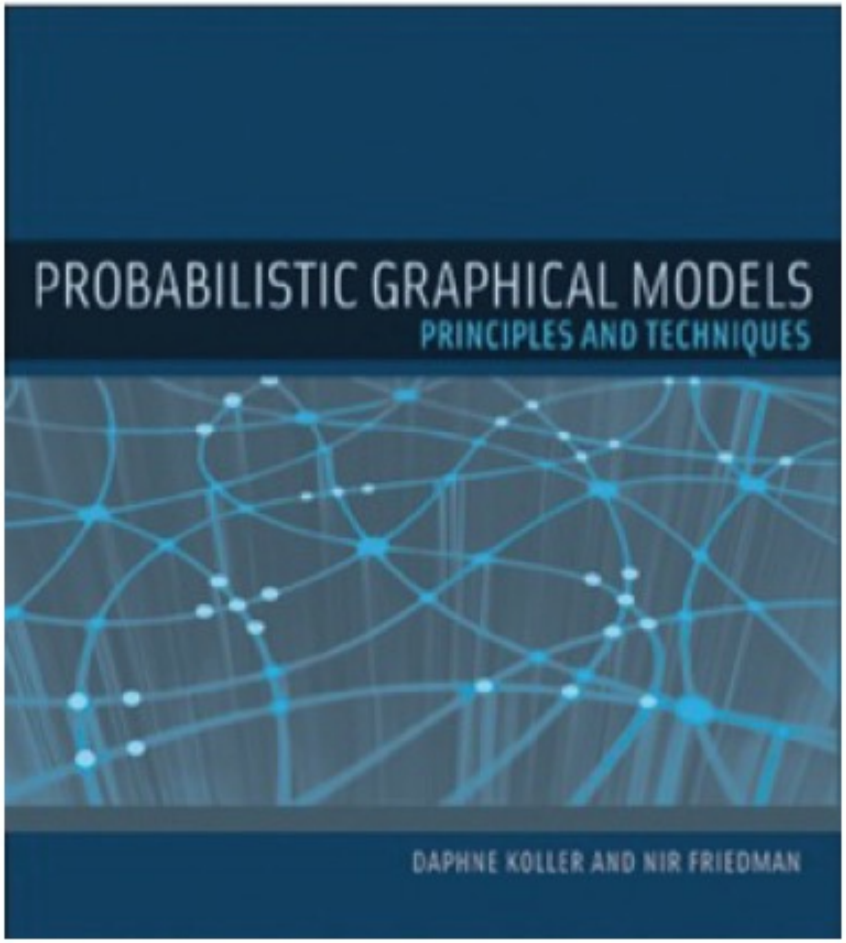
- Graphical models (see Kevin Murphy's survey, 2001) are (directed or undirected) graphs in which nodes represent random variables, and the lack of arcs represent (**assumed**) conditional independences: two sets of nodes are **conditionally independent** given a third set D, if all paths between nodes in A and B are separated by D.
  - Graphical models include mixture models, hidden Markov models, Kalman filters, etc

  Several R packages: http://cran.at.r-project.org/web/views/gR.html

- Naïve Bayes is a simple directed graphical model



(Way Beyond Classification)



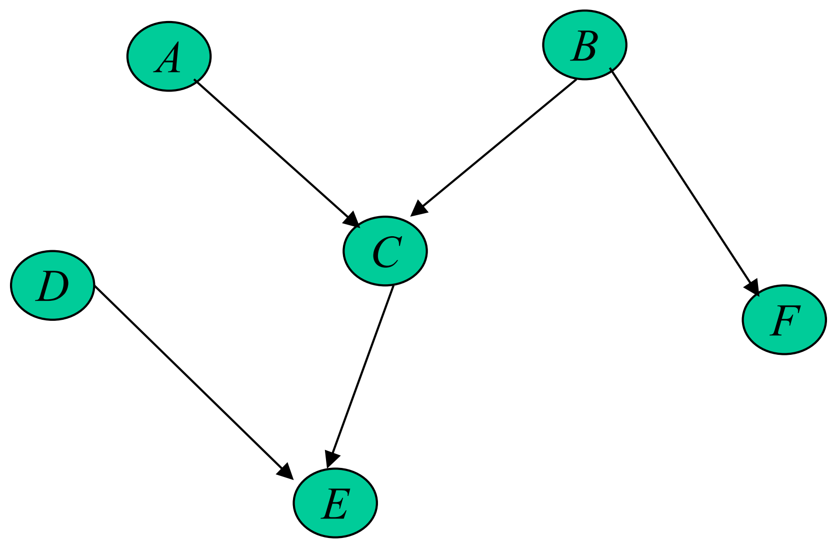
Daphne Koller
Computer Science Dept.
Stanford University

Nir Friedman
School of Computer Science & Engineering
Hebrew University

MIT Press, 2009, 1231 pages

Bayesian (Belief) Networks

- Directed Acyclic Graph on **all** variables.
- Allows combining prior knowledge about (in)dependences among variables with observed training data



1. Any variable is conditionally independent of all non-descendent variables given its parents.
2. Graph also imposes partial ordering, e.g. A,B,C,D,E,F

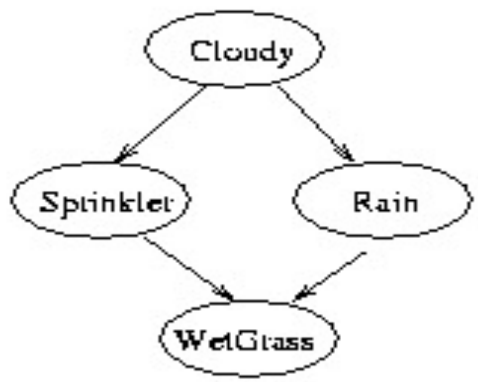
From (1) and (2), get P(A,B,C,D,E,F) = P(A)P(B)P(C|A,B)P(D)P(E|C,D)P(F|B)

Π_i P(node i | parents of node i)

Example – Wet Grass (Murphy 01)

- [See http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html](http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html)
- From data, get Conditional Probability Distribution/Table (CPD or CPT) for each variable.

P(C=F)	P(C=T)
0.5	0.5



C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Takeaways

- Network structure is modeling assumption
  - Exploit domain knowledge
  - Few edges means more independence among variables , so smaller CPTs
- Very flexible: can infer in any direction and involving any subset of variables.
  - Also suggest explanations
- Training data used to fill up the CPTs

Network Properties and Usage

- Network structure is a modeling assumption, not necessarily unique
  - Some are better than others (good data fit + low complexity)
- If causality is known, make network from root causes …. to end effects.
- Usage: Inferencing
  - Infer the (probabilities of) values of one or more variables given observed values of some others.

Software: OpenBUGS http://mathstat.helsinki.fi/openbugs/
Python package
Infer.net http://research.microsoft.com/en-us/um/cambridge/projects/infernet/default.aspx
R packages such as bnlearn

Inference: Effect to Cause (Bottom Up)

We observe the grass is wet. Is this because of sprinkler or because of rain? (T=1, F=0).

$P(S=1|W=1) =$

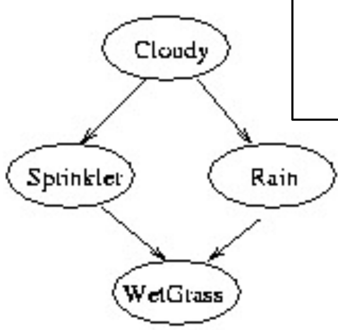
$\Sigma_{c,r}\, P(C=c, S=1, R=r, W=1)$ / **$P(W=1)$ = 0.2781/ .6471 = .43**

$P(R=1|W=1) =$

$\Sigma c,s\ P(C=c, S=s, R=1, W=1) / P(W=1)$ **= 0.4581/ .6471 = .708**

So more likely it is because of rain!

P(C=F)	P(C=T)
0.5	0.5



C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Special Types of inference

Diagnostic - B is evidence of A (bottom-up)

previous example

Predictive - A can cause B (top-down)

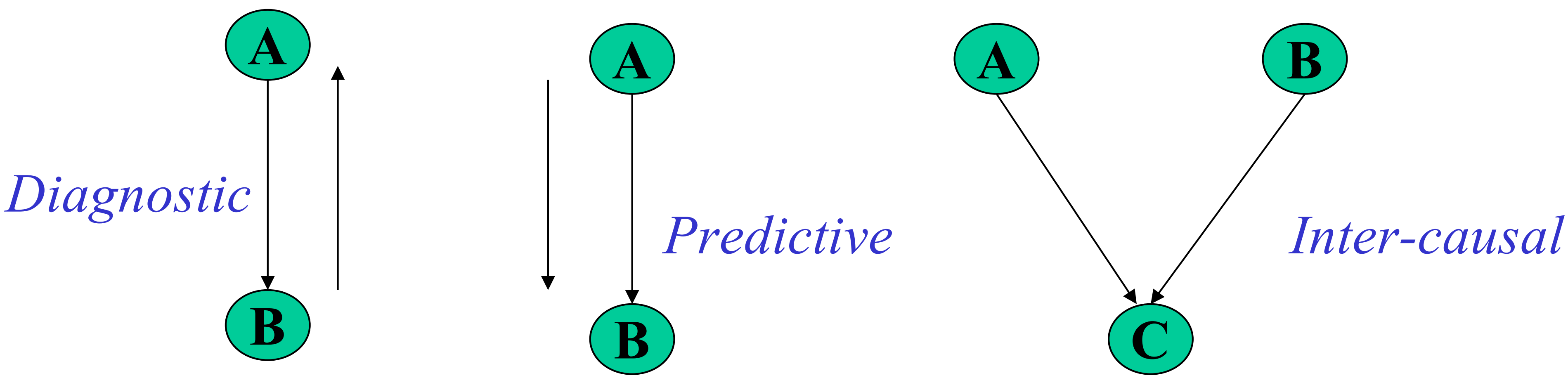
- e.g. P(grass wet | cloudy)

Inter-causal - suppose both A and B can cause C

if A “explains” C, it is evidence against B

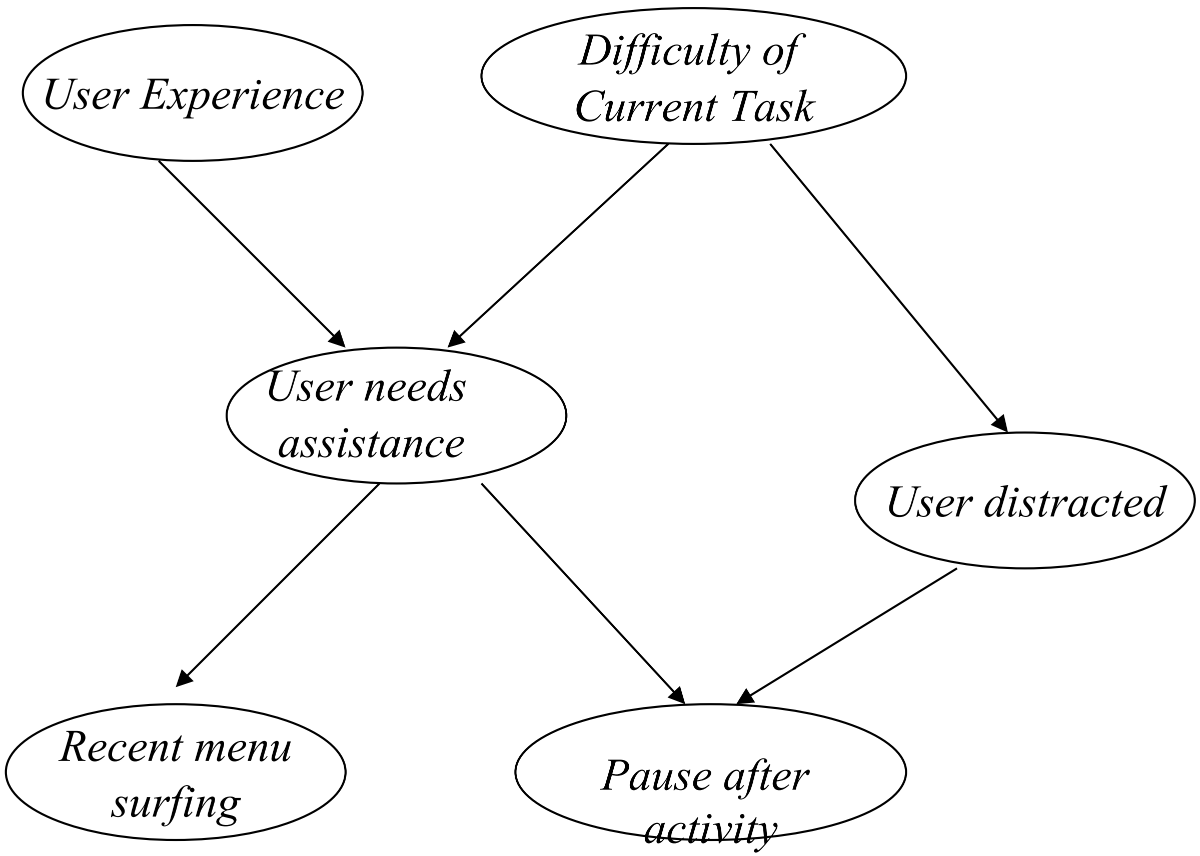
e.g. P(S=1 | W=1, R=1) = 0.1945, i.e. lower chance of sprinkler being ON if one also knows that it rained!

(“explaining away”,“Berkson's paradox”, or "selection bias")



Microsoft Office Assistant

- Only part of Bayesian network shown (Horvitz et al, Lumiere Project)



More Classification Methods

Directly getting to the Posterior: Logistic Regression, Neural Networks

Misc: SVMs

Logistic Regression (intro)

Logistic Regression

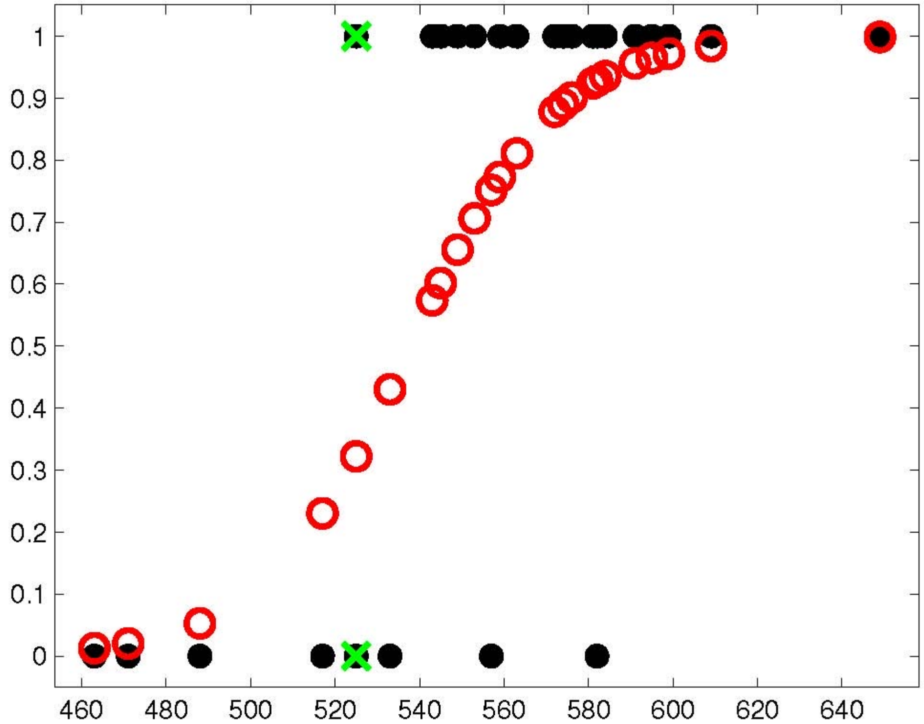
- Models a categorical variable (e.g. class label) as a function of predictors
- Studied extensively and have well-developed theory (variable selection methods, model diagnostic checks, extensions for dealing with correlated data)

Let $y(\mathbf{x})$ = 0/1 Target variable for binary classification (C_0 vs. C_1)

- **Then** $\mu = E(y \mid \mathbf{x}) = P(C_1 | \mathbf{x})$
  Model: $\ln [\mu / (1 - \mu)] = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$
  $$= \beta . \mathbf{x} \qquad \text{(I)}$$
- i.e. model "log-odds ratio" (aka **logit**) as a linear function of predictors

Formulation

- Equivalently: model $\mu = 1/(1+\exp(-\beta.\ \mathbf{x})) = \sigma\ (\beta.\ \mathbf{x})$
  - σ is called **the logistic function**, which is the inverse of logit
  - Rewrite model as
  - $\mu = \exp(\beta.\ \mathbf{x})/(\exp(\beta.\ \mathbf{x}) + 1)$ and note that $1 = \exp(0)$ to get a hint of how to generalize for more than 2 classes.



SAT data
From KM pg 259.

Training

Minimize Negative Log-Likelihood (NLL) of a suitable probability model

Implied Stat Model: y' s are i.i.d. Bernoulli

- $p(y|\mathbf{x}, \beta) = \text{Bernoulli}(y \mid \boldsymbol{\sigma}(\boldsymbol{\beta} . \mathbf{x}))$
- So $\text{NLL}(\beta) = - \Sigma_i [\, y_i \log \mu_i + (1-y_i) \log(1-\mu_i) \,]$

(cross-entropy error function)

If we use $\tilde{y}_i \in \{-1,1\}$ *then*

$\text{NLL}(\beta) = \Sigma_i \log(1+ \exp(- \tilde{y}_i \,\boldsymbol{\beta} . \mathbf{x}_i))$

> Takeaway: a non-linear max-likelihood problem needs to be solved iteratively.

The unknown parameters (β) are estimated by maximum likelihood. (gradient descent or iterative solutions, e.g. Newton-Raphson, or iterative reweighted least squares see KM 8.3)

SGD for Logistic Regression

- Loss for *i*th data point (when target is encoded as 0/1)

$$= \quad -[y_i \log \sigma(\boldsymbol{\beta} \cdot x_i + b) + (1 - y_i) \log(1 - \sigma(\boldsymbol{\beta} \cdot x_i + b))]$$

- So gradient is: $\dfrac{\partial L(\boldsymbol{\beta}, b)}{\partial \boldsymbol{\beta}_j} = [\sigma(\boldsymbol{\beta} \cdot x_i + b) - y]x_{ij}$

  – Where x_{ij} refers to the *j*th feature of x_i.

- The overall gradient: $\nabla L(f(x; \boldsymbol{\beta}), y)) = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\beta}_1} L(f(x; \boldsymbol{\beta}), y) \\ \frac{\partial}{\partial \boldsymbol{\beta}_2} L(f(x; \boldsymbol{\beta}), y) \\ \vdots \\ \frac{\partial}{\partial \boldsymbol{\beta}_n} L(f(x; \boldsymbol{\beta}), y) \end{bmatrix}$

- SGD Update: $\boldsymbol{\beta}_{new} = \boldsymbol{\beta}_{old} - \eta \nabla L\left(f\left(x; \boldsymbol{\beta}_{old}\right), y\right)$

SGD for Logistic Regression

- Loss for *i*th data point (when target is encoded as -1/1)

$= \quad \log\left(1 + e^{-y_i \boldsymbol{\beta} \cdot \mathbf{x}_i}\right)$

- So gradient is: $\dfrac{\partial L(\boldsymbol{\beta}, b)}{\partial \boldsymbol{\beta}_j} = \left(\dfrac{-y_i \mathbf{x}_{ij}}{1 + e^{y_i \boldsymbol{\beta} \cdot \mathbf{x}_i}}\right)$

  – Where x_{ij} refers to the *j*th feature of x_i .

- The overall gradient: $\nabla L(f(x;\boldsymbol{\beta}), y)) = \begin{bmatrix} \frac{\partial}{\partial \boldsymbol{\beta}_1} L(f(x;\boldsymbol{\beta}), y) \\ \frac{\partial}{\partial \boldsymbol{\beta}_2} L(f(x;\boldsymbol{\beta}), y) \\ \vdots \\ \frac{\partial}{\partial \boldsymbol{\beta}_n} L(f(x;\boldsymbol{\beta}), y) \end{bmatrix}$

- SGD Update: $\boldsymbol{\beta}_{new} = \boldsymbol{\beta}_{old} - \eta \nabla L\left(f\left(x; \boldsymbol{\beta}_{old}\right), y\right)$

Properties

- Logistic regression is an example of a generalized linear model (GLM), with canonical link function = logit, corresponding to Bernoulli (see glmnet in R)

- Disadvantages:
  - Solution not simple closed form, but still reasonably fast
- Advantages:
  - Have parameters with useful interpretations
    - the effect of a unit change in x_i is to increase the odds of a response multiplicatively by the factor $\exp(\beta_i)$
  - Quite robust, well developed

Multiclass Logistic Regression

- Extension to K classes: use K-1 models
  - one each for $\ln [P(C_i|x)/P(C_k|x)]$
  - Set all coefficients for class K to 0 (to make the system **identifiable;** this choice is arbitrary)

- Put them together to get posteriors.
  - $P(C_i|\mathbf{x}) = \exp(\beta_{i0} + \beta_{i1}x_1 \,..) / (1 + \Sigma_j \exp(\beta_{j0} + \beta_{j1}x_1 \,..)$, $i \neq k$
  - $P(C_K|\mathbf{x}) = \ldots.$

Visit to SAS


Y
H
C
B
$\Pr(Y=1) = \frac{1}{1+\exp\{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\beta_3 x_3+\beta_4 x_4)\}}$
G
O
A
X
K
P
L
-Y
B
c
x

Multilayered Feedforward Networks for Classification

- choose sufficiently large network (no. of hidden units)
- trained by "1 of M" desired output values
  - ("one-hot coding")
- use validation set to determine when to stop training
- try several initializations, training regimens
- + powerful, nonlinear, flexible
- - interpretation? (Needs extra effort); slow ?

MLPs as Approximate Bayes Classifiers

- Output of “universal” Feedforward neural nets (MLP, RBF, deep nets) trained by "1 of M" desired output values, estimate Bayesian *a posteriori* probabilities if the cost function is

  mean square error OR cross-entropy

Nowadays some packages use softmax, but often not needed.

- Significance:
  - interpretation of outputs; quality of results
  - setting of thresholds for acceptance/rejection
  - can combine outputs of multiple networks

Softmax

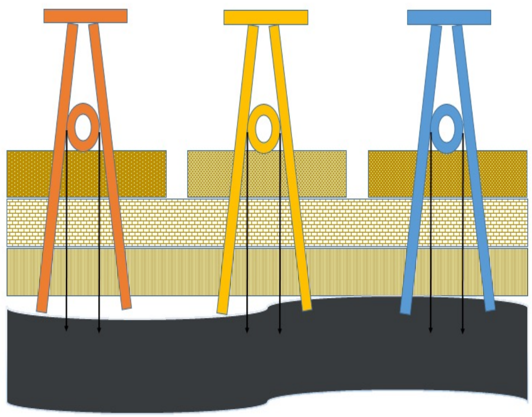
- Monotonic transformation of a set of numbers: x → t(x)
  - All t(x) are non-negative
  - Sum to one
  - Potential for interpretation as discrete probabilities.

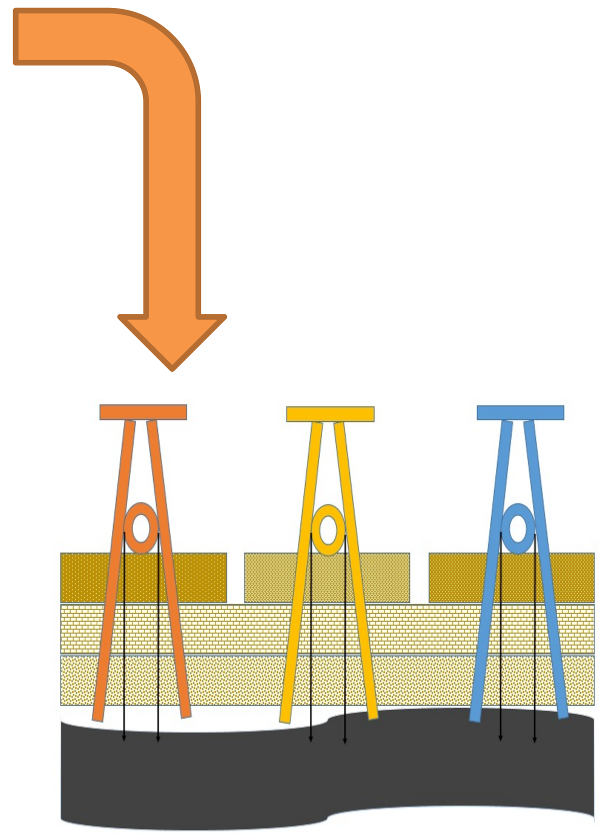
Exponentiate and normalize.

*Multi-Task & Transfer Learning**

[*Transfer Learning*]()

make use of the knowledge gained while solving one problem and applying it to a different but related problem.





[*Multitask Learning*]()

Simultaneously learning multiple (related) tasks

[*Code leaders*]()

[Active Learning](): *incrementally recruiting labelled points based on analysis of model on existing training set. Usually evaluated using a "banana" plot.*

Extras

Naïve Bayes Example

Step 1: Estimating Univariate Probabilities for each variable-class combination

Discrete Probabilities: (Binary/Categorical)

See tax evasion example in main slides.

Continuous variables:

- Discretize the range into bins
  - one ordinal attribute per bin
  - violates independence assumption
- Binarize: (may lose substantial info)
- Probability density estimation:
  - (usually assuming Normal distribution)

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:

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

  – One for each (x_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) $\times$ P(Married| Class=No) $\times$ P(Income=120K| Class=No) = 4/7 $\times$ 4/7 $\times$ 0.0072 = 0.0024

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

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

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

=> Class = No

Smoothing Naïve Bayes

- Avoids zero probablity due to one attribute-value/class combo being absent in training data.
  - Zeroes entire product term
- Probability estimation:

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

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

c: number of classes

p: prior probability

m: weight of prior (i.e. # of virtual samples)

e.g. in text analysis, add a “virtual document that has one instance of every word in the vocabulary (Laplace smoothing): $(N_{ic} + 1) / (Nc + |vocab|)$