MAXIMUM ENTROPY

David Kauchak
CS457, Spring 2011

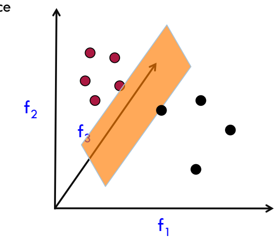
Some material derived from Jason Eisner

Linear classifier

- A linear classifier predicts the label based on a weighted, linear combination of the features

$$prediction = w_0 + w_1 f_1 + w_2 f_2 + \ldots + w_m f_m$$

- For two classes, a linear classifier can be viewed as a plane (hyperplane) in the feature space



Linear regression

Predict the response based on a weighted, linear combination of the features

$$h(\bar{f}) = w_0 + w_1 f_1 + w_2 f_2 + \ldots + w_m f_m$$

real value

weights

Learn weights by minimizing the square error on the training data

$$error(h) = \sum_{i=1}^{n} (y_i - (w_0 + w_1 f_1 + w_2 f_2 + \ldots + w_m f_m))^2$$

Logistic regression

$$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m$$

$$\frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = e^{w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m}$$

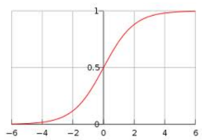
$$P(1 \mid x_1, x_2, \ldots, x_m) = (1 - P(1 \mid x_1, x_2, \ldots, x_m)) e^{w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m}$$

$$\ldots$$

$$P(1 \mid x_1, x_2, \ldots, x_m) = \frac{1}{1 + e^{-(w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m)}}$$

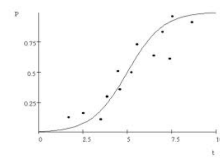
Logistic function

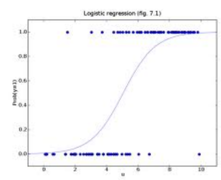
$$\text{logistic} = \frac{1}{1+e^{-x}}$$



Logistic regression

- Find the best fit of the data based on a logistic





Logistic regression

- How would we classify examples once we had a trained model?

$$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m$$

- If the sum > 0 then p(1)/p(0) > 1, so positive
- if the sum < 0 then p(1)/p(0) < 1, so negative
- Still a *linear* classifier (decision boundary is a line)

3 views of logistic regression

$$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m$$

linear classifier

…

$$P(1 \mid x_1, x_2, \ldots, x_m) = \frac{e^{w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m}}{1 + e^{w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m}}$$

exponential model
(log-linear model)

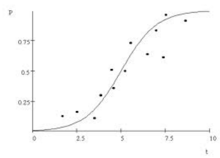
…

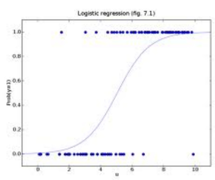
$$P(1 \mid x_1, x_2, \ldots, x_m) = \frac{1}{1 + e^{-(w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m)}}$$

logistic

Logistic regression

- Find the best fit of the data based on a logistic function





Training logistic regression models

- How should we learn the parameters for logistic regression (i.e. the w's)?

$$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m$$

parameters

$$P(1 \mid x_1, x_2, \ldots, x_m) = \frac{1}{1 + e^{-(w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m)}}$$

Training logistic regression models

- Idea 1: minimize the squared error (like linear regression)
  - Any problems?

$$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m$$

$$error(h) = \sum_{i=1}^{n} (y_i - h(f_i))^2$$

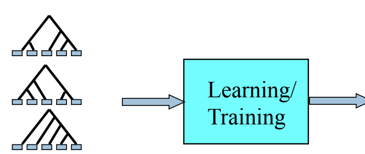
  - We don't know what the actual probability values are!

A digression

Why is this called Maximum Likelihood Estimation (MLE)?

Parsed sentences

Grammar



⋮

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

English

$$P(\alpha \to \beta \mid \alpha) = \frac{\text{count}(\alpha \to \beta)}{\text{count}(\alpha)}$$

MLE

- Maximum likelihood estimation picks the values for the model parameters that maximize the likelihood of the training data

parameters

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

parameter values

model (Θ)

MLE

- Maximum likelihood estimation picks the values for the model parameters that maximize the likelihood of the training data

parameters parameter values

S → NP VP	0.9
S → VP	0.1
NP → Det A N	0.5
NP → NP PP	0.3
NP → PropN	0.2
A → ε	0.6
A → Adj A	0.4
PP → Prep NP	1.0
VP → V NP	0.7
VP → VP PP	0.3

model (Θ)

$$MLE(data) = \arg\max_{\theta} p_{\theta}(data)$$

$$= \arg\max_{\theta} \prod_i p_{\theta}(data_i)$$

$$= \arg\max_{\theta} \log(\sum_i p_{\theta}(data_i))$$

If this is what you want to optimize, you can do NO BETTER than MLE!

MLE example

- You flip a coin 100 times. 60 times you get heads.
- What is the MLE for heads?
  - p(head) = 0.60
- What is the likelihood of the data under this model (each coin flip is a data point)?

$$likelihood(data) = \prod_i p_{\theta}(data_i)$$

$\log(0.60^{60} * 0.40^{40}) = -67.3$

MLE Example

- Can we do any better?

$$likelihood(data) = \prod_i p_{\theta}(data_i)$$

- p(heads) = 0.5
  - $\log(0.50^{60} * 0.50^{40}) = -69.3$
- p(heads) = 0.7
  - $\log(0.70^{60} * 0.30^{40}) = -69.5$

Training logistic regression models

- Idea 1: minimize the squared error (like linear regression)

$$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m$$

$$error(h) = \sum_{i=1}^{n} (y_i - h(f_i))^2$$

  - We don't know what the actual probability values are!

Ideas?

Training logistic regression models

- Idea 2: maximum likelihood training

$$MLE(data) = \underset{\theta}{\arg\max}\, p_\theta(data)$$

$$= \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} p_{\bar{w}}(label_i \mid \bar{f}_i)$$

$$= \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} \log p_{\bar{w}}(label_i \mid \bar{f}_i)$$

...

How do we solve this?

Training logistic regression models

- Idea 2: maximum likelihood training

$$MLE(data) = \underset{\theta}{\arg\max}\, p_\theta(data)$$

$$= \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} p_{\bar{w}}(label_i \mid \bar{f}_i)$$

$$= \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} \log p_{\bar{w}}(label_i \mid \bar{f}_i)$$

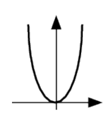
...

1. plug in our logistic equation
2. take partial derivatives and solve

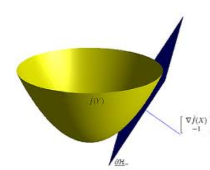
Unfortunately, no closed form solution.

Convex functions

- Convex functions look something like:



convex function



- What are some nice properties about convex functions?
- How can we find the minimum/maximum of a convex function?

Finding the minimum

You're blindfolded, but you can see out of the bottom of the blindfold to the ground right by your feet. I drop you off somewhere and tell you that you're in a convex shaped valley and escape is at the bottom/minimum. How do you get out?

One approach: gradient descent

- Partial derivatives give us the slope in that dimension
- Approach:
  - pick a starting point (w)
  - repeat until likelihood can't increase in any dimension:
    - pick a dimension
    - move a small amount in that dimension towards increasing likelihood (using the derivative)

Gradient descent

- pick a starting point (w)
- repeat until loss doesn't decrease in all dimensions:
  - pick a dimension
  - move a small amount in that dimension towards decreasing loss (using the derivative)

$$w_i = w_i - \alpha \frac{d}{dw_i} error(w)$$

learning rate (how much we want to move in the error direction)

Solving convex functions

- Gradient descent is just one approach
- A whole field called convex optimization
  - http://www.stanford.edu/~boyd/cvxbook/
- Lots of well known methods
  - Conjugate gradient
  - Generalized Iterative Scaling (GIS)
  - Improved Iterative Scaling (IIS)
  - Limited-memory quasi-Newton (L-BFGS)
- The key: if we get an error function that is convex, we can minimize/maximize it (eventually)

Another thought experiment

What is a 100,000-dimensional space like?

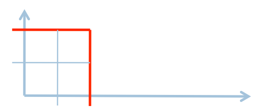
You're a 1-D creature, and you decide to buy a 2-unit apartment

2 rooms (very, skinny rooms)

Another thought experiment

What is a 100,000-dimensional space like?

Your job's going well and you're making good money. You upgrade to a 2-D apartment with 2-units per dimension

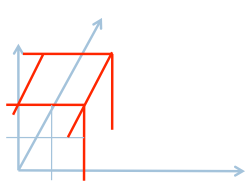


4 rooms (very, flat rooms)

Another thought experiment

What is a 100,000-dimensional space like?

You get promoted again and start having kids and decide to upgrade to another dimension.



8 rooms (very, normal rooms)

Each time you add a dimension, the amount of space you have to work with goes up exponentially

Another thought experiment

What is a 100,000-dimensional space like?

Larry Page steps down as CEO of google and they ask you if you'd like the job. You decide to upgrade to a 100,000 dimensional apartment.

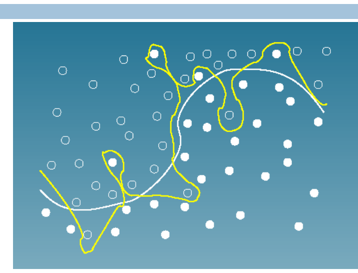
How much room do you have?
Can you have a big party?

$2^{100,000}$ rooms (it's very quiet and lonely…) = ~10^{30} rooms per person if you invited everyone on the planet

The challenge

- Because logistic regression has fewer constraints (than, say NB) it has a lot more options
- We're trying to find 100,000 *w* values (or a point in a 100,000 dimensional space)
- It's easy for logistic regression to fit to nuances in the data: overfitting

Overfitting



Given these points as **training** data, which is a better line to learn to separate the points?

Preventing overfitting

$$\log \frac{P(1 \mid x_1,x_2,\ldots,x_m)}{1-P(1 \mid x_1,x_2,\ldots,x_m)} = w_0 + w_1x_2 + w_2x_2 + \ldots + w_mx_m$$

We want to avoid any single feature from having too much weight

$$MLE(data) = \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} \log p_{\bar{w}}(y \mid \bar{f})$$ normal MLE

ideas?

Preventing overfitting

$$\log \frac{P(1 \mid x_1,x_2,\ldots,x_m)}{1-P(1 \mid x_1,x_2,\ldots,x_m)} = w_0 + w_1x_2 + w_2x_2 + \ldots + w_mx_m$$

We want to avoid any single feature from having too much weight

$$MLE(data) = \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} \log p_{\bar{w}}(y \mid \bar{f})$$ normal MLE

$$MLE(data) = \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} \log p_{\bar{w}}(y \mid \bar{f}) \boxed{- \alpha \sum_{j=1}^{m} w_j^2}$$ regularized MLE

Preventing overfitting: regularization

$$MLE(data) = \underset{\bar{w}}{\arg\max} \sum_{i=1}^{n} \log p_{\bar{w}}(y \mid \bar{f}) \boxed{- \alpha \sum_{j=1}^{m} w_j^2}$$

regularized MLE

What affect will this have on the learned weights assuming a positive α?

penalize large weights
encourage smaller weights

- still a convex problem!
- equivalent to assuming your w_j are distributed from a Gaussian with mean 0 (called a prior)

NB vs. Logistic regression

- NB and logistic regression look very similar
  - both are probabilistic models
  - both are linear
  - both learn parameters that maximize the log-likelihood of the training data
- How are they different?

NB vs. Logistic regression

NB

$$f_1 \log(P(f_1 \mid l)) + \bar{f}_1 \log(1 - P(f_1 \mid l)) + \ldots + \log(P(l))$$

Estimates the weights under the strict assumption that the features are independent

Naïve bayes is called a *generative* model; it models the joint distribution p(features, labels)

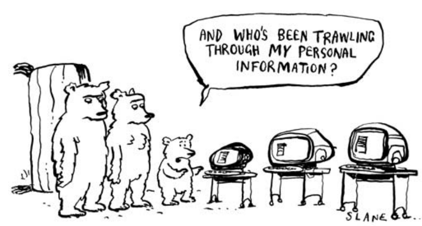
Logistic regression

$$\frac{e^{w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m}}{1 + e^{w_0 + w_1 x_2 + w_2 x_2 + \ldots + w_m x_m}}$$

If NB assumption doesn't hold, we can adjust the weights to compensate for this

Logistic regression is called a *discriminative* model; it models the conditional distribution directly p(labels | features)

Some historical perspective



http://www.reputation.com/blog/2010/02/17/privacy-a-historical-perspective/

Estimating the best chess state

Write a function that takes as input a “state” representation of tic tac toe and scores how good it is for you if you’re X. How would you do it?

(Called a *state evaluation function*)

Old school optimization

- Possible parses (or whatever) have scores
- Pick the one with the best score
- How do you define the score?
  - Completely ad hoc!
  - **Throw anything you want into the mix**
  - Add a bonus for this, a penalty for that, etc.
  - State evaluation function for chess…

Old school optimization

- “Learning”
  - adjust bonuses and penalties by hand to improve performance. ☺
- Total kludge, but totally flexible too …
  - Can throw in **any** intuitions you might have
- But we’re purists… we only use probabilities!

New “revolution”?

- Probabilities!

New "revolution"?

- Prob

Exposé at 9

Probabilistic Revolution Not Really a Revolution, Critics Say

Probabilities no more than scores in disguise

"We're just adding stuff up like the old corrupt regime did," admits spokesperson

83% of

Probabilists Rally Behind ^ Paradigm

".2, .4, .6, .8! We're not gonna take your bait!"

1. Can estimate our parameters *automatically*
   - e.g., p(t7 | t5, t6)   (trigram probability)
   - from supervised or unsupervised data
2. Our results are more meaningful
   - Can use probabilities to place bets, quantify risk
   - e.g., how sure are we that this is the correct parse?
3. Our results can be meaningfully combined ⇒ modularity!
   - Multiply indep. conditional probs – normalized, unlike scores
   - p(English text) * p(English phonemes | English text) * p(Jap. phonemes | English phonemes) * p(Jap. text | Jap. phonemes)
   - p(semantics) * p(syntax | semantics) * p(morphology | syntax) * p (phonology | morphology) * p(sounds | phonology)

Probabilists Regret Being Bound by Principle

- Ad-hoc approach does have one advantage
- Consider e.g. Naïve Bayes for spam categorization:
  - `Buy this supercalifragilistic Ginsu knife set for only $39 today …`
- Some useful features:
  - Contains `Buy`
  - Contains `supercalifragilistic`
  - Contains a dollar amount under $100
  - Contains an imperative sentence
  - Reading level = 8th grade
  - Mentions money (use word classes and/or regexp to detect this)

Any problem with these features for NB?

Probabilists Regret Being Bound by Principle

`Buy this supercalifragilistic Ginsu knife set for only $39 today …`

- Naïve Bayes
  - Contains a dollar amount under $100
  - Mentions money (use word classes and/or regexp to detect this)

	Spam	not-Spam
< $100	0.5	0.02
Money amount	0.9	0.1

How likely is it to see both features in either class using NB? Is this right?

Probabilists Regret Being Bound by Principle

```
Buy this supercalifragilistic Ginsu knife set for
only $39 today …
```

- Naïve Bayes
  - Contains a dollar amount under $100
  - Mentions money (use word classes and/or regexp to detect this)

	Spam	not-Spam
< $100	0.5	0.02
Money amount	0.9	0.1

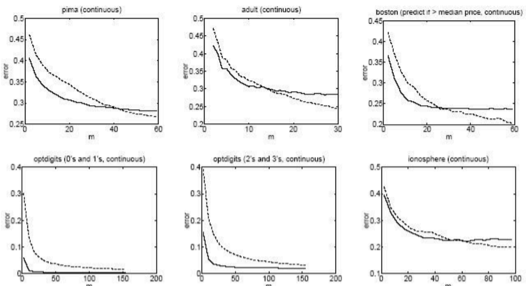
0.5*0.9=0.45 0.02*0.1=0.002

Overestimates! The problem is that the features are not independent

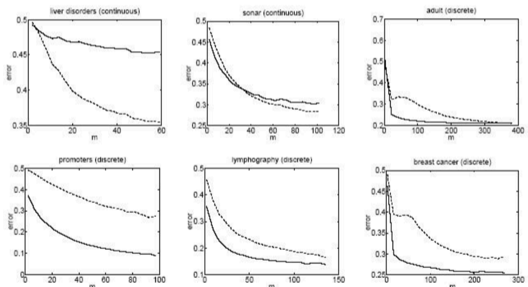
NB vs. Logistic regression

- Logistic regression allows us to put in features that overlap and adjust the probabilities accordingly
- Which to use?
  - NB is better for small data sets: strong model assumptions keep the model from overfitting
  - Logistic regression is better for larger data sets: can exploit the fact that NB assumption is rarely true

NB — vs. Logistic regression ----



NB — vs. Logistic regression ----



Logistic regression with more classes

- NB works on multiple classes
- Logistic regression only works on two classes
- Idea: something like logistic regression, but with more classes
  - Like NB, one model per each class
  - The model is a weight vector

$$P(class_1 \mid x_1, x_2, \ldots, x_m) = e^{w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m}$$

$$P(class_2 \mid x_1, x_2, \ldots, x_m) = e^{w_{2,0} + w_{2,1}x_2 + w_{2,2}x_2 + \ldots + w_{2,m}x_m}$$

$$P(class_3 \mid x_1, x_2, \ldots, x_m) = e^{w_{3,0} + w_{3,1}x_2 + w_{3,2}x_2 + \ldots + w_{3,m}x_m}$$

…

anything wrong with this?

Challenge: probabilistic modeling

$$P(class_1 \mid x_1, x_2, \ldots, x_m) = e^{w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m}$$

$$P(class_2 \mid x_1, x_2, \ldots, x_m) = e^{w_{2,0} + w_{2,1}x_2 + w_{2,2}x_2 + \ldots + w_{2,m}x_m}$$

$$P(class_3 \mid x_1, x_2, \ldots, x_m) = e^{w_{3,0} + w_{3,1}x_2 + w_{3,2}x_2 + \ldots + w_{3,m}x_m}$$

…

These are supposed to be probabilities!

$$P(class_1 \mid x_1, x_2, \ldots, x_m) + P(class_2 \mid x_1, x_2, \ldots, x_m) + P(class_3 \mid x_1, x_2, \ldots, x_m) + \ldots \neq 1$$

Ideas?

Maximum Entropy Modeling aka Multinomial Logistic Regression

Normalize each class probability by the sum over all the classes

$$P(class_1 \mid x_1, x_2, \ldots, x_m) = \frac{e^{w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m}}{P(class_1 \mid x_1, x_2, \ldots, x_m) + P(class_2 \mid x_1, x_2, \ldots, x_m) + P(class_3 \mid x_1, x_2, \ldots, x_m) + \ldots}$$

$$P(class_1 \mid x_1, x_2, \ldots, x_m) = \frac{e^{w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m}}{\sum_{i=1}^{|C|} P(class_i \mid x_1, x_2, \ldots, x_m)}$$

$$= \frac{e^{w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m}}{\sum_{i=1}^{|C|} e^{w_{i,0} + w_{i,1}x_2 + w_{i,2}x_2 + \ldots + w_{i,m}x_m}}$$

normalizing constant

Log-linear model

$$P(class_1 \mid x_1, x_2, \ldots, x_m) = \frac{e^{w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m}}{\sum_{i=1}^{|C|} P(class_i \mid x_1, x_2, \ldots, x_m)}$$

$$\log P(class_1 \mid x_1, x_2, \ldots, x_m) = w_{1,0} + w_{1,1}x_2 + w_{1,2}x_2 + \ldots + w_{1,m}x_m - \log\left(\sum_{i=1}^{|C|} P(class_i \mid x_1, x_2, \ldots, x_m)\right)$$

- still just a linear combination of feature weightings
- class specific features

Training the model

- Can still use maximum likelihood training

$$MLE(data) = \underset{\theta}{\arg\max} \sum_{i=1}^{n} \log p(label_i \mid \bar{f}_i)$$

- Use regularization

$$MLE(data) = \underset{\theta}{\arg\max} \sum_{i=1}^{n} \log p(label_i \mid \bar{f}_i) - \alpha R(\theta)$$

- Plug into a convex optimization package
  - there are a few complications, but this is the basic idea

Maximum Entropy

- Suppose there are 10 classes, A through J.
- I don't give you any other information.
- Question: Given a new example m: what is your guess for p(C | m)?

- Suppose I tell you that 55% of all examples are in class A.
- Question: Now what is your guess for p(C | m)?

- Suppose I also tell you that 10% of all examples contain `Buy` and 80% of these are in class A or C.
- Question: Now what is your guess for p(C | m), if m contains `Buy`?

Maximum Entropy

	A	B	C	D	E	F	G	H	I	J
prob	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Qualitatively

Maximum entropy principle: given the constraints, pick the probabilities as "equally as possible"

Quantitatively

Maximum entropy: given the constraints, pick the probabilities so as to maximize the entropy

$$Entropy(\text{model}) = \sum_{c} p(c) \log p(c)$$

Maximum Entropy

	A	B	C	D	E	F	G	H	I	J
prob	0.55	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Qualitatively

Maximum entropy principle: given the constraints, pick the probabilities as "equally as possible"

Quantitatively

Maximum entropy: given the constraints, pick the probabilities so as to maximize the entropy

$$Entropy(\text{model}) = \sum_{c} p(c) \log p(c)$$

Maximum Entropy

	A	B	C	D	E	F	G	H	I	J
Buy	.051	.0025	.029	.0025	.0025	.0025	.0025	.0025	.0025	.0025
Other	.499	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446

- Column A sums to 0.55 ("55% of all messages are in class A")

Maximum Entropy

	A	B	C	D	E	F	G	H	I	J
Buy	.051	.0025	.029	.0025	.0025	.0025	.0025	.0025	.0025	.0025
Other	.499	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446

- Column A sums to 0.55
- Row `Buy` sums to 0.1 ("10% of all messages contain `Buy`")

Maximum Entropy

	A	B	C	D	E	F	G	H	I	J
Buy	.051	.0025	.029	.0025	.0025	.0025	.0025	.0025	.0025	.0025
Other	.499	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446

- Column A sums to 0.55
- Row `Buy` sums to 0.1
- (`Buy`, A) and (`Buy`, C) cells sum to 0.08 ("80% of the 10%")

- Given these constraints, fill in cells "as equally as possible": maximize the entropy (related to cross-entropy, perplexity)

Entropy = -.051 log .051 - .0025 log .0025 - .029 log .029 - …

Largest if probabilities are evenly distributed

Maximum Entropy

	A	B	C	D	E	F	G	H	I	J
Buy	.051	.0025	.029	.0025	.0025	.0025	.0025	.0025	.0025	.0025
Other	.499	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446	.0446

- Column A sums to 0.55
- Row `Buy` sums to 0.1
- (`Buy`, A) and (`Buy`, C) cells sum to 0.08 ("80% of the 10%")

- Given these constraints, fill in cells "as equally as possible": maximize the entropy
- Now p(`Buy`, C) = .029 and p(C | `Buy`) = .29
- We got a compromise: p(C | `Buy`) < p(A | `Buy`) < .55

Generalizing to More Features

<$100

Other

	A	B	C	D	E	F	G	H	…
Buy	.051	.0025	.029	.0025	.0025	.0025	.0025	.0025	
Other	.499	.0446	.0446	.0446	.0446	.0446	.0446	.0446	

What we just did

- For each feature ("contains Buy"), see what fraction of training data has it
- Many distributions p(c,m) would predict these fractions
- Of these, pick distribution that has max entropy

- Amazing Theorem: The maximum entropy model is the same as the maximum likelihood model!
  - If we calculate the maximum likelihood parameters, we're also calculating the maximum entropy model

What to take home…

- Many learning approaches
  - Bayesian approaches (of which NB is just one)
  - Linear regression
  - Logistic regression
  - Maximum Entropy (multinomial logistic regression)
  - SVMs
  - Decision trees
  - …
- Different models have different strengths/weaknesses/uses
  - Understand what the model is doing
  - Understand what assumptions the model is making
  - Pick the model that makes the most sense for your problem/data
- Feature selection is important

Articles discussion

- http://gigaom.com/mobile/wilson-siri-call-911/