LOGISTIC REGRESSION

David Kauchak
CS451 – Fall 2013

Admin

Assignment 7

CS Lunch on Thursday

Priors

Coin1 data: 3 Heads and 1 Tail
Coin2 data: 30 Heads and 10 tails
Coin3 data: 2 Tails
Coin4 data: 497 Heads and 503 tails

If someone asked you what the probability of heads was for each of these coins, what would you say?

Training revisited

From a probability standpoint, what we're really doing when we're training the model is selecting the Θ that maximizes:

$$p(\theta \mid data)$$

i.e.

$$\text{argmax}_\theta \, p(\theta \mid data)$$

That we pick the most likely model parameters given the data

Estimating revisited

We can incorporate a prior belief in what the probabilities might be

To do this, we need to break down our probability

$$p(\theta \mid data) = ?$$

(Hint: Bayes rule)

Estimating revisited

What are each of these probabilities?

$$p(\theta \mid data) = \frac{p(data \mid \theta) p(\theta)}{p(data)}$$

Priors

likelihood of the data under the model

probability of different parameters, call the prior

$$p(\theta \mid data) = \frac{p(data \mid \theta) p(\theta)}{p(data)}$$

probability of seeing the data (regardless of model)

Priors

$$\theta = \text{argmax}_\theta \frac{p(data \mid \theta) p(\theta)}{p(data)}$$

Does p(data) matter for the argmax?

Priors

likelihood of the data under the model

probability of different parameters, call the prior

$$\theta = \text{argmax}_\theta\, p(data \mid \theta) p(\theta)$$

What does MLE assume for a prior on the model parameters?

Priors

likelihood of the data under the model

probability of different parameters, call the prior

$$\theta = \text{argmax}_\theta\, p(data \mid \theta) p(\theta)$$

- Assumes a uniform prior, i.e. all Θ are equally likely!
- Relies solely on the likelihood

A better approach

$$\theta = \text{argmax}_\theta\, p(data \mid \theta) p(\theta)$$

$$likelihood(data) = \prod_{i=1}^{n} p_\theta(x_i)$$

We can use any distribution we'd like
This allows us to impart addition bias into the model

Another view on the prior

Remember, the max is the same if we take the log:

$$\theta = \text{argmax}_\theta \log(p(data \mid \theta)) + \log(p(\theta))$$

$$\log - likelihood = \sum_{i=1}^{n} \log(p(x_i))$$

We can use any distribution we'd like
This allows us to impart addition bias into the model

Does this look like something we've seen before?

Regularization vs prior

$$\theta = \text{argmax}_\theta \log(p(data \mid \theta)) + \log(p(\theta))$$

fit: likelihood based on the data / loss function based on the data

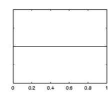
model bias: prior / regularizer

$$\text{argmin}_{w,b} \sum_{i=1}^{n} loss(yy') + \lambda regularizer(w)$$

Prior for NB

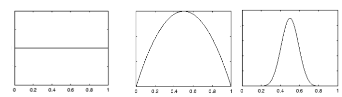
$$\theta = \text{argmax}_\theta \log(p(data \mid \theta)) + \log(p(\theta))$$

Uniform prior



$$p(x_i \mid y) = \frac{count(x_i, y)}{count(y)}$$

Dirichlet prior



$\lambda = 0$ → increasing

$$p(x_i \mid y) = \frac{count(x_i, y) + \lambda}{count(y) + possible\_values\_of\_x_i * \lambda}$$

Prior: another view

$$p(x_1, x_2, \ldots, x_m, y) = p(y) \prod_{j=1}^{m} p(x_i \mid y)$$

MLE: $p(x_i \mid y) = \dfrac{count(x_i, y)}{count(y)}$

What happens to our likelihood if, for one of the labels, we never saw a particular feature?

Goes to 0!

Prior: another view

$$p(x_i \mid y) = \frac{count(x_i, y)}{count(y)}$$

$$p(x_i \mid y) = \frac{count(x_i, y) + \lambda}{count(y) + possible\_values\_of\_x_i * \lambda}$$

Adding a prior avoids this!

Smoothing

$$p(x_i \mid y) = \frac{count(x_i, y)}{count(y)}$$

$$p(x_i \mid y) = \frac{count(x_i, y) + \lambda}{count(y) + possible\_values\_of\_x_i * \lambda}$$

for each label, pretend like we've seen each feature value occur in λ additional examples

Sometimes this is also called smoothing because it is seen as smoothing or interpolating between the MLE and some other distribution

Basic steps for probabilistic modeling

Step 1: pick a model

Step 2: figure out how to estimate the probabilities for the model

Step 3 (optional): deal with overfitting

Probabilistic models

Which model do we use, i.e. how do we calculate p(*feature, label*)?

How do train the model, i.e. how to we we estimate the probabilities for the model?

How do we deal with overfitting?

Joint models vs conditional models

We've been trying to model the joint distribution (i.e. the data generating distribution):

$$p(x_1, x_2, \ldots, x_m, y)$$

However, if all we're interested in is classification, why not directly model the conditional distribution:

$$p(y \mid x_1, x_2, \ldots, x_m)$$

A first try: linear

$$p(y \mid x_1, x_2, \ldots, x_m) = x_1 w_1 + w_2 x_2 + \ldots + w_m x_m + b$$

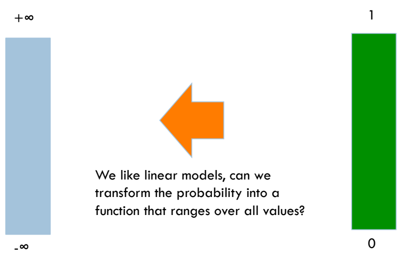
Any problems with this?

- Nothing constrains it to be a probability
- Could still have combination of features and weight that exceeds 1 or is below 0

The challenge

$x_1w_1 + w_2x_2 + \ldots + w_mx_m + b$ — Linear model

$p(y \mid x_1, x_2, \ldots, x_m)$ — probability



Odds ratio

Rather than predict the probability, we can predict the ratio of 1/0 (positive/negative)

Predict the **odds** that it is 1 (true): How much more likely is 1 than 0.

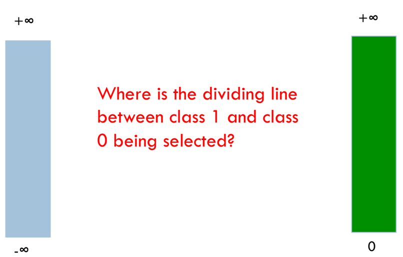
Does this help us?

$$\frac{P(1 \mid x_1, x_2, \ldots, x_m)}{P(0 \mid x_1, x_2, \ldots, x_m)} = \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)} = x_1w_1 + w_2x_2 + \ldots + w_mx_m + b$$

Odds ratio

$x_1w_1 + w_2x_2 + \ldots + w_mx_m + b$ — Linear model

$\frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)}$ — odds ratio



Odds ratio

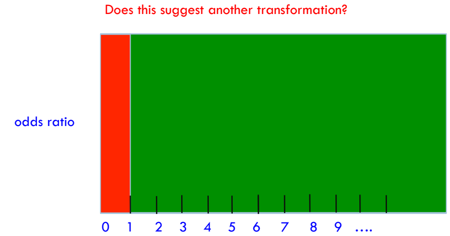
$P(1 \mid x_1, x_2, \ldots, x_m) > P(0 \mid x_1, x_2, \ldots, x_m)$

$P(1 \mid x_1, x_2, \ldots, x_m) > 1 - P(1 \mid x_1, x_2, \ldots, x_m)$

$\frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)}$

We're trying to find some transformation that transforms the odds ratio to a number that is -∞ to +∞

Does this suggest another transformation?



Log odds (logit function)

$x_1 w_1 + w_2 x_2 + \ldots + w_m x_m + b$

Linear regression

$\log \frac{P(1 \mid x_1, x_2, \ldots, x_m)}{1 - P(1 \mid x_1, x_2, \ldots, x_m)}$

odds ratio

How do we get the probability of an example?

Log odds (logit function)

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

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

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

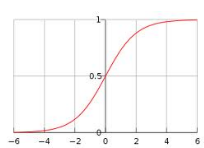
…

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

anyone recognize this?

Logistic function

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



Logistic regression

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

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

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)

Training logistic regression models

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

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

parameters

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

MLE logistic regression

Find the parameters that maximize the likelihood (or log-likelihood) of the data:

$$\log - likelihood = \sum_{i=1}^{n} \log(p(x_i))$$

$$= \sum_{i=1}^{n} \log\left(\frac{1}{1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}}\right)$$

assume labels 1, -1

$$= \sum_{i=1}^{n} -\log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})$$

MLE logistic regression

$$\log - likelihood = \sum_{i=1}^{n} -\log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})$$

We want to maximize, i.e.

$$MLE(data) = \text{argmax}_{w,b} \log - likelihood(data)$$

$$= \text{argmax}_{w,b} \sum_{i=1}^{n} -\log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})$$

$$= \arg\min_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})$$

Look familiar? Hint: anybody read the book?

MLE logistic regression

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})$$

Surrogate loss functions:

Zero/one:	$\ell^{(0/1)}(y, \hat{y}) = \mathbf{1}[y\hat{y} \le 0]$
Hinge:	$\ell^{(hin)}(y, \hat{y}) = \max\{0, 1 - y\hat{y}\}$
Logistic:	$\ell^{(log)}(y, \hat{y}) = \frac{1}{\log 2} \log(1 + \exp[-y\hat{y}])$
Exponential:	$\ell^{(exp)}(y, \hat{y}) = \exp[-y\hat{y}]$
Squared:	$\ell^{(sqr)}(y, \hat{y}) = (y - \hat{y})^2$

logistic regression: three views

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

linear classifier

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

conditional model
logistic

$$\mathrm{argmin}_{w,b}\sum_{i=1}^{n}\log(1+e^{-y_i(w_1x_2+w_2x_2+\ldots+w_mx_m+b)})$$

linear model
minimizing logistic loss

Overfitting

$$\mathrm{argmin}_{w,b}\sum_{i=1}^{n}\log(1+e^{-y_i(w_1x_2+w_2x_2+\ldots+w_mx_m+b)})$$

If we minimize this loss function, in practice, the results aren't great and we tend to overfit

Solution?

Regularization/prior

$$\mathrm{argmin}_{w,b}\sum_{i=1}^{n}\log(1+e^{-y_i(w_1x_2+w_2x_2+\ldots+w_mx_m+b)}) + \lambda\, regularizer(w,b)$$

or

$$\mathrm{argmin}_{w,b}\sum_{i=1}^{n}\log(1+e^{-y_i(w_1x_2+w_2x_2+\ldots+w_mx_m+b)}) - \log(p(w,b))$$

What are some of the regularizers we know?

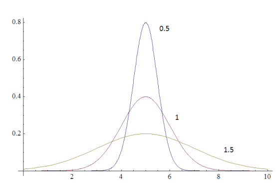
Regularization/prior

L2 regularization:

$$\mathrm{argmin}_{w,b}\sum_{i=1}^{n}\log(1+e^{-y_i(w_1x_2+w_2x_2+\ldots+w_mx_m+b)}) + \lambda\|w\|^2$$

Gaussian prior:

p(w,b) ~



Regularization/prior

L2 regularization:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \lambda \|w\|^2$$

Gaussian prior:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \frac{1}{2\sigma^2} \|w\|^2$$

Does the λ make sense?

$$\lambda = \frac{1}{2\sigma^2}$$

Regularization/prior

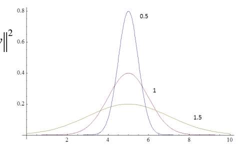
L2 regularization:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \lambda \|w\|^2$$

Gaussian prior:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \frac{1}{2\sigma^2} \|w\|^2$$

$$\lambda = \frac{1}{2\sigma^2}$$



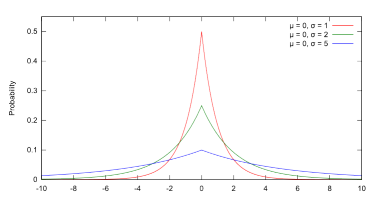
Regularization/prior

L1 regularization:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \lambda \|w\|$$

Laplacian prior:

p(w,b) ~



Regularization/prior

L1 regularization:

$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \lambda \|w\|$$

Laplacian prior:

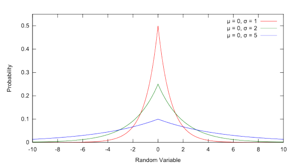
$$\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)}) + \frac{1}{\sigma} \|w\|$$

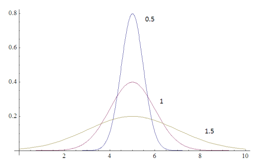
$$\lambda = \frac{1}{2\sigma^2}$$

L1 vs. L2

L1 = Laplacian prior

L2 = Gaussian prior

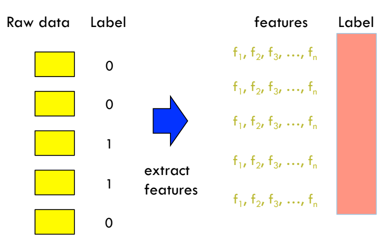




Logistic regression

Why is it called logistic regression?

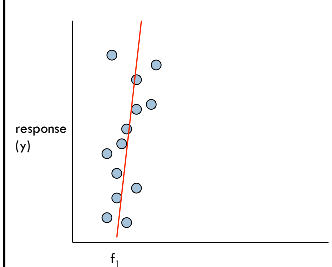
A digression: regression vs. classification



classification: discrete (some finite set of labels)

regression: real value

linear regression



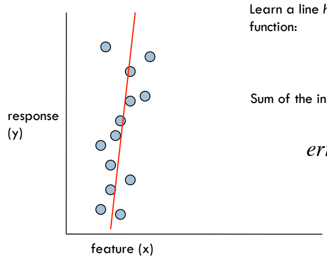
Given some points, find the ***line*** that best fits/explains the data

Our model is a line, i.e. we're assuming a linear relationship between the feature and the label value

$$h(y) = w_1 x_1 + b$$

How can we find this line?

Linear regression



Learn a line *h* that minimizes some loss/error function:

$$error(h) = ?$$

Sum of the individual errors:

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

0/1 loss!

Error minimization

How do we find the minimum of an equation?

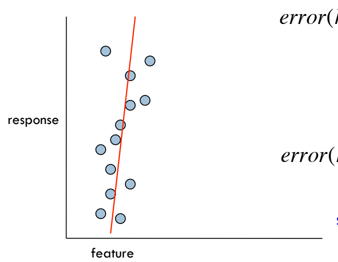
$$error(h) = \sum_{i=1}^{n} |y_i - h(f_i)|$$

Take the derivative, set to 0 and solve (going to be a min or a max)

Any problems here?

Ideas?

Linear regression

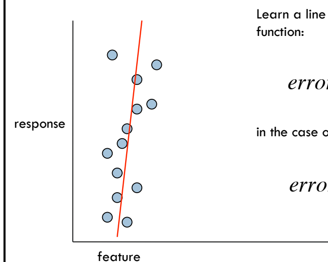


$$error(h) = \sum_{i=1}^{n} |y_i - h(f_i)|$$

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

squared error is convex!

Linear regression



Learn a line *h* that minimizes an error function:

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

in the case of a 2d line:

$$error(h) = \sum_{i=1}^{n} (y_i - (w_1 x_1 + w_0))^2$$

function for a line

Linear regression

We'd like to *minimize* the error

Find w_1 and w_0 such that the error is minimized

$$error(h) = \sum_{i=1}^{n} (y_i - (w_1 f_i + w_0))^2$$

We can solve this in closed form

Multiple linear regression

If we have m features, then we have a line in *m* dimensions

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

weights

Multiple linear regression

We can still calculate the squared error like before

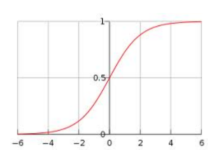
$$h(\bar{f}) = w_0 + w_1 f_1 + w_2 f_2 + ... + w_m f_m$$

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

Still can solve this exactly!

Logistic function

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



Logistic regression

Find the best fit of the data based on a logistic