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:

\[ p(θ|data) \]

i.e.

\[ \arg\max_θ p(θ|data) \]

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:

\[ p(\theta | \text{data}) = ? \]

(Hint: Bayes rule)

Priors

\[ p(\theta | \text{data}) = \frac{p(\text{data} | \theta)p(\theta)}{p(\text{data})} \]

likelihood of the data under the model

\[ p(\theta | \text{data}) = \frac{p(\text{data} | \theta)p(\theta)}{p(\text{data})} \]

probability of different parameters, call the prior

\[ \theta = \arg\max_\theta \frac{p(\text{data} | \theta)p(\theta)}{p(\text{data})} \]

probability of seeing the data (regardless of model)

Does \( p(\text{data}) \) matter for the \( \arg\max \)?
What does MLE assume for a prior on the model parameters?

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

A better approach

\[ \theta = \arg\max_{\theta} p(\text{data} | \theta) p(\theta) \]

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

\[ \theta = \arg\max_{\theta} \log(p(\text{data} | \theta)) + \log(p(\theta)) \]

Another view on the prior

\[ \text{log-likelihood} = \sum x_i \log(p(x_i)) \]

Does this look like something we’ve seen before?
Regularization vs prior

\[ \theta = \arg \max_{\theta} \log(p(data|\theta)) + \log(p(\theta)) \]

Prior for NB

\[ \theta = \arg \max_{\theta} \log(p(data|\theta)) + \log(p(\theta)) \]

Prior: another view

\[ p(x_1, x_2, ..., x_n, y) = p(y) \prod_{i=1}^{n} p(x_i|y) \]

MLE: \[ p(x_i|y) = \frac{\text{count}(x_i, y)}{\text{count}(y)} \]

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

Goes to 0!

Adding a prior avoids this!
Smoothing

for each label, pretend like we’ve seen each feature value occur in \( \lambda \) additional examples.

\[
p(x_i \mid y) = \frac{\text{count}(x_i, y)}{\text{count}(y)}
\]

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

Which model do we use, i.e. how do we calculate \( p(\text{feature}, \text{label}) \)?

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

How do we deal with overfitting?

Probabilistic models

Step 1: pick a model

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

Step 3 (optional): deal with overfitting

Joint models vs conditional models

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

\[
p(x_1, x_2, \ldots, x_n, y)
\]

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

\[
p(y \mid x_1, x_2, \ldots, x_n)
\]

A first try: linear

\[
p(y \mid x_1, x_2, \ldots, x_n) = x_1w_1 + x_2w_2 + \ldots + x_nw_n + b
\]

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

We like linear models, can we transform the probability into a function that ranges over all values?

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?

\[
P(1|x_1, x_2, ..., x_m) = \frac{P(1|x_1, x_2, ..., x_m)}{1 - P(1|x_1, x_2, ..., x_m)} = x_1w_1 + x_2w_2 + ... + w_mx_m + b
\]
Log odds (logit function)

\[ x_1 w_1 + x_2 w_2 + \ldots + x_n w_n + b \]

Linear regression

Log odds (logit function)

\[ \log \frac{P(1 | x_1, x_2, \ldots, x_n)}{1 - P(1 | x_1, x_2, \ldots, x_n)} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b \]

Odds ratio

Logistic function

\[ \text{logistic} = \frac{1}{1 + e^{-z}} \]

Logistic regression

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

\[ \log \frac{P(1 | x_1, x_2, \ldots, x_n)}{1 - P(1 | x_1, x_2, \ldots, x_n)} = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b \]

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)?

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

MLE logistic regression

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

\[
\log\text{-likelihood} = \sum_i \log(p(x_i)) - \sum_i \log(1 + e^{-y_i (w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})
\]

MLE logistic regression

\[
\text{MLE (data)} = \text{argmax}_w \log\text{-likelihood (data)}
\]

\[
= \text{argmax}_w \sum_i \log(1 + e^{-y_i (w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})
\]

\[
= \text{argmin}_w \sum_i \log(1 + e^{-y_i (w_1 x_2 + w_2 x_2 + \ldots + w_m x_m + b)})
\]

Logistic surrogate loss functions:

- Zero/one: \(\ell^0(y, \hat{y}) = 1[y \neq \hat{y}]\)
- Hinge: \(\ell^\text{hinge}(y, \hat{y}) = \max(0, 1 - y \hat{y})\)
- Logistic: \(\ell^\text{log}(y, \hat{y}) = \frac{1}{\log 2} \log(1 + e^{-y \hat{y}})\)
- Exponential: \(\ell^\text{exp}(y, \hat{y}) = \exp[-y \hat{y}]\)
- Squared: \(\ell^\text{sqr}(y, \hat{y}) = (y - \hat{y})^2\)
logistic regression: three views

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

linear classifier

conditional model

logistic

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

Solution?

Regularization/prior

\[
\text{Regularization/prior: } \argmin_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) - \log(p(w,b))
\]

What are some of the regularizers we know?

Regularization/prior

L2 regularization:

\[
\text{L2 regularization: } \argmin_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|^2
\]

Gaussian prior:

\[
p(w,b) \
\]

Overfitting
Regularization/prior

L2 regularization:
\[
\arg\min_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|^2
\]

Gaussian prior:
\[
\arg\min_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \frac{1}{2\sigma^2} \|w\|^2
\]

Does the \(\lambda\) make sense?

L1 regularization:
\[
\arg\min_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|_1
\]

Laplacian prior:
\[
 p(w,b) \sim \exp(-\frac{1}{\sigma} \|w\|_1)
\]

L2 regularization:
\[
\arg\min_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|^2
\]

Gaussian prior:
\[
\arg\min_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \frac{1}{2\sigma^2} \|w\|^2
\]

\[\lambda = \frac{1}{2\sigma^2}\]
L1 vs. L2

L1 = Laplacian prior
L2 = Gaussian prior

Logistic regression

Why is it called logistic regression?

A digression:
regression vs. classification

Raw data | Label | features: \( f_1, f_2, f_3, \ldots, f_n \) | Label |
--- | --- | --- | --- |
| | | 0 | classification: discrete (some finite set of labels) |
| | | 0 | regression: real value |
| | | 1 | |
| | | 1 | |
| | | 0 | | 

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

\[
h(y) = w_1 x_1 + b
\]
Learn a line \( h \) that minimizes some loss/error function:

\[
error(h) = \sum |y_i - h(x_i)|
\]

Sum of the individual errors:

\[
error(h) = \sum_{i=1}^{n} |y_i - h(x_i)|
\]

0/1 loss!

How do we find the minimum of an equation?

\[
error(h) = \sum |y_i - h(x_i)|
\]

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

Any problems here?

Ideas?

Squared error is convex!
Linear regression

We'd like to minimize the error.

Find \( w_1 \) and \( w_0 \) such that the error is minimized:

\[
\text{error}(h) = \sum_{i=1}^{n} \left( y_i - (w_1 f_i + w_0) \right)^2
\]

We can solve this in closed form.

Multiple linear regression

If we have \( m \) features, then we have a line in \( m \) dimensions:

\[
h(f) = w_0 + w_1 f_1 + w_2 f_2 + \ldots + w_m f_m
\]

We can still calculate the squared error like before:

\[
h(f) = w_0 + w_1 f_1 + w_2 f_2 + \ldots + w_m f_m
\]

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

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