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?

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 \( p(\text{feature}, \text{label}) \)?

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

How do we deal with overfitting?
Training revisited

What we’re really doing during training is selecting the $\Theta$ that maximizes:

$$p(\theta|\text{data})$$

i.e.

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

That is, we pick the most likely model parameters given the data.

Estimating revisited

We want to incorporate a prior belief of what the probabilities might be.

To do this, we need to break down our probability

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

(Hint: Bayes rule)

Estimating revisited

What are each of these probabilities?

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

Priors

- likelihood of the data under the model
- probability of different parameters, call the prior

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

- probability of seeing the data (regardless of model)
Priors

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

Does \( p(\text{data}) \) matter for the \( \arg\max \)?

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

A better approach

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

- Assumes a uniform prior, i.e. all \( \theta \) are equally likely!
- Relies solely on the likelihood

We can use any distribution we’d like. This allows us to impart additional bias into the model.
Another view on the prior

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

$$\theta = \arg\max_\theta \log(p(data|\theta)) + \log(p(\theta))$$

log-likelihood = \sum log(p(x_i))

We can use any distribution we’d like. This allows us to impart additional bias into the model.

Does this look like something we’ve seen before?

Regularization vs prior

$$\theta = \arg\max_\theta \log(p(data|\theta)) + \log(p(\theta))$$

$\lambda$ regularizer

loss function based on the data

 società prior

argmin_{w,} \sum_{i=1} \text{loss}(y^i) + \lambda \text{regularizer}(w)$

Prior for NB

$$\theta = \arg\max_\theta \log(p(data|\theta)) + \log(p(\theta))$$

Uniform prior

Dirichlet prior

$p(x_{i},y) = \frac{\text{count}(x_i,y)}{\text{count}(y)}$

$p(x_{i},y) = \frac{\text{count}(x_i,y) + \lambda}{\text{count}(y) + \text{possible values of } x_i + \lambda}$

Prior: another view

$$p(x_1, x_2, ..., x_m, y) = p(y) \propto \prod_{i=1} p(x_i | y)$$

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!
Prior: another view

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

$$p(x_i | y) = \frac{\text{count}(x_i, y) + \lambda}{\text{count}(y) + \text{possible_values_of} \_x_i + \lambda}$$

Adding a prior avoids this!

Smoothing

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

$$p(x_i | y) = \frac{\text{count}(x_i, y) + \lambda}{\text{count}(y) + \text{possible_values_of} \_x_i + \lambda}$$

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

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

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

$$p(x_i | y) = \frac{\text{count}(x_i, y) + \lambda}{\text{count}(y) + \text{possible_values_of} \_x_i + \lambda}$$

Does this do the right thing in these cases?

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 $p(\text{feature}, \text{label})$?

How do we train the model, i.e., how to 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):

\[ p(x_1, x_2, ..., x_m, y) \]

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

\[ p(y | x_1, x_2, ..., x_m) \]

A first try: linear

\[ p(y | x_1, x_2, ..., x_m) = x_1 w_1 + w_2 x_2 + ... + w_m x_m + b \]

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

\[ x_1 w_1 + w_2 x_2 + ... + w_m x_m + b \]

Linear model

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?

\[ \frac{P(1 | x_1, x_2, ..., x_m)}{P(0 | x_1, x_2, ..., x_m)} = \frac{P(1 | x_1, x_2, ..., x_m)}{1 - P(1 | x_1, x_2, ..., x_m)} = x_1 w_1 + w_2 x_2 + ... + w_m x_m + b \]
Odds ratio

Where is the dividing line between class 1 and class 0 being selected?

Log odds (logit function)

How do we get the probability of an example?
Log odds (logit function)

\[
\begin{align*}
\log \frac{P(1|x_1, x_2, \ldots, x_n)}{1-P(1|x_1, x_2, \ldots, x_n)} &= w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b \\
\frac{P(1|x_1, x_2, \ldots, x_n)}{1-P(1|x_1, x_2, \ldots, x_n)} &= e^{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b} \\
P(1|x_1, x_2, \ldots, x_n) &= (1-P(1|x_1, x_2, \ldots, x_n))e^{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b} \\
\cdots \\
P(1|x_1, x_2, \ldots, x_n) &= \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b)}}
\end{align*}
\]

Logistic function

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

Logistic regression

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

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

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 and b)?

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

\[
P(1|x_1, x_2, \ldots, x_n) = \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b)}}
\]
MLE logistic regression

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

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

$$= \sum_{i=1}^{n} \log \left( \frac{1}{1 + e^{-(x_i^T w + b)}} \right)$$

$$= \sum_{i=1}^{n} -\log(1 + e^{-(x_i^T w + b)})$$

We want to maximize, i.e.

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

$$= \arg \max_{w, b} \sum_{i=1}^{n} -\log(1 + e^{-(x_i^T w + b)})$$

$$= \arg \min_{w, b} \sum_{i=1}^{n} \log(1 + e^{-(x_i^T w + b)})$$

Look familiar? Hint: anybody read the book?

MLE logistic regression

MLE logistic regression

log-likelihood = \( \sum_{i=1}^{n} -\log(1 + e^{-(x_i^T w + b)}) \)

We want to maximize, i.e.

\( \text{MLE (data)} = \arg \max_{w, b} \text{log-likelihood (data)} \)

\( = \arg \max_{w, b} \sum_{i=1}^{n} -\log(1 + e^{-(x_i^T w + b)}) \)

\( = \arg \min_{w, b} \sum_{i=1}^{n} \log(1 + e^{-(x_i^T w + b)}) \)

MLE logistic regression

MLE logistic regression

MLE logistic regression

logistic regression: three views

$$P(1 | x_1, x_2, \ldots, x_n) = \frac{1}{1 + e^{-(x_1 w_1 + x_2 w_2 + \ldots + x_n w_n)}}$$

$$= \frac{1}{1 + e^{-\sum_{i=1}^{n} x_i w_i}}$$

Conditional model logistic

$$\text{linear classifier}$$

$$\text{logistic}$$

$$\text{linear model minimizing logistic loss}$$

$$\text{argmin}_{w, b} \sum_{i=1}^{n} \log(1 + e^{-(x_i^T w + b)})$$

$$\text{argmin}_{w, b} \sum_{i=1}^{n} \log(1 + e^{-(x_i^T w + b)})$$

Surrogate loss functions:

- **Zero/one**: \( \ell^{(0/1)}(y, \hat{y}) = 1[y \neq \hat{y}] \)
- **Hinge**: \( \ell^{(\text{hinge})}(y, \hat{y}) = \max(0, 1 - y \hat{y}) \)
- **Logistic**: \( \ell^{(\text{logistic})}(y, \hat{y}) = -\frac{1}{\log 2} \log(1 + \exp(-y \hat{y})) \)
- **Exponential**: \( \ell^{(\text{exponential})}(y, \hat{y}) = \exp(-y \hat{y}) \)
- **Squared**: \( \ell^{(\text{squared})}(y, \hat{y}) = (y - \hat{y})^2 \)
Overfitting

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

Solution?

Regularization/prior

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

Solution?

Regularization/prior

L2 regularization:

$\text{arg min}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|_2^2$

Gaussian prior:

Gaussians are defined by a mean $\mu$ and a variance $\sigma^2$

$p(w,b) \sim \mathcal{N}(\mu, \sigma^2)$

Regularization/prior

L2 regularization:

$\text{arg min}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|_2^2$

Gaussian prior:

$p(w,b) \sim \mathcal{N}(\mu, \sigma^2)$

Does the $\lambda$ make sense? $\lambda = \frac{1}{2\sigma^2}$
### L2 Regularization

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

**Gaussian prior:**

\[
\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \frac{1}{2\sigma^2} \|w\|^2
\]

\[\lambda = \frac{1}{2\sigma^2}\]

---

### L1 Regularization

\[
\text{argmin}_{w,b} \sum_{i=1}^{n} \log(1 + e^{-y_i(w^T x_i + b)}) + \lambda \|w\|
\]

**Laplacian prior:**

\[p(w|b) \sim \frac{1}{\sigma} e^{-\frac{|w|}{\sigma}}\]

---

### L1 vs. L2

\[
\text{L1 = Laplacian prior} \quad \text{L2 = Gaussian prior}
\]
Logistic regression

Why is it called logistic regression? It is a classifier??

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

A digression: regression vs. classification

Raw data

Label

features

Label

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?

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

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

Any problems here?

Ideas?

Linear regression

Learn a line \( h \) that minimizes an error function:

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

in the case of a 2d line:

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

Linear regression

We’d like to minimize the error

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

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

We can solve this in closed form
Multiple linear regression

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

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

weights

We can still calculate the squared error like before

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

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

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
Basic steps for probabilistic modeling

<table>
<thead>
<tr>
<th>Step 1: pick a model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2: figure out how to estimate the probabilities for the model</td>
</tr>
<tr>
<td>Step 3 (optional): deal with overfitting</td>
</tr>
</tbody>
</table>

Probabilistic models

- Which model do we use, i.e. how do we calculate \( p(\text{feature, label}) \)?
- How do we train the model, i.e. how do we estimate the probabilities for the model?
- How do we deal with overfitting?

Probabilistic models summarized

Two classification models:
- Naïve Bayes (models joint distribution)
- Logistic Regression (models conditional distribution)
  - In practice this tends to work better if all you want to do is classify

Priors/smoothing/regularization
- Important for both models
- In theory: allow us to impart some prior knowledge
- In practice: avoids overfitting and often tune on development data