

Linear models for classification

We've talked about regressing on \mathbb{R} ... what about $\{-1, +1\}$? Consider our old example, augmented with some new features.

Length ≥ 155	Width ≥ 90	Cool Design	Type
1	0	1	Ski
1	0	0	Ski
1	1	0	Ski
1	0	0	Ski
1	0	1	Ski
1	1	0	Ski
1	0	0	Ski
0	0	0	Ski
1	1	1	Snowboard
1	1	1	Snowboard
0	1	1	Snowboard
0	1	1	Snowboard
0	0	0	Snowboard
0	1	0	Snowboard
0	0	1	Snowboard
0	1	1	Snowboard

Idea: learn a predictor that is largely positive for one class and largely negative for another.

Sigmoid function:

$$\begin{aligned}\sigma(x) &= \frac{\exp[x/2]}{\exp[x/2] + \exp[-x/2]} \\ &= \frac{1}{1 + \exp[-x]}\end{aligned}$$

Inverse is the logistic:

$$\sigma^{-1}(y) = \log\left(\frac{y}{1-y}\right)$$

Specify a probability distribution on y conditioned on x :

$$p_{\theta}(y | x) = \sigma(y\theta^{\top}x) = \frac{1}{1 + \exp[-y\theta^{\top}x]}$$

This is the *logistic regression* model.

No closed-form solution exists; compute log likelihood:

$$\ell(\theta) = \log \prod_n p_{\theta}(y_n | x_n)$$

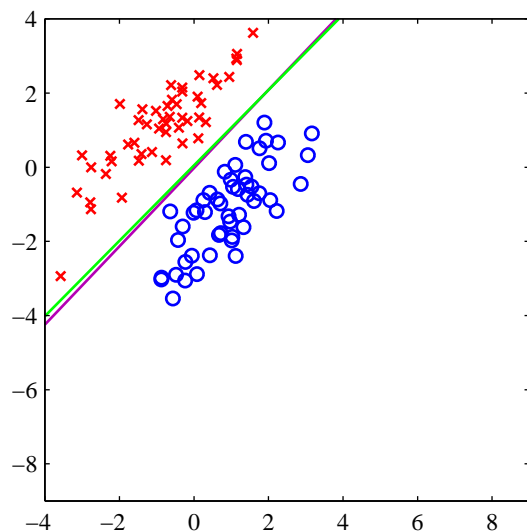
$$\begin{aligned}
&= \sum_n \log \frac{1}{1 + \exp[-y_n \theta^\top x_n]} \\
&= - \sum_n \log (1 + \exp[-y_n \theta^\top x_n])
\end{aligned}$$

Now, compute the gradient.

$$\begin{aligned}
\nabla_{\theta} \ell &= - \sum_n \frac{1}{1 + \exp[-y_n \theta^\top x_n]} \exp[-y_n \theta^\top x_n] (-y_n x_n) \\
&= \sum_n \frac{1}{1 + \exp[y_n \theta^\top x_n]} y_n x_n
\end{aligned}$$

And we can do gradient descent.

This is a *linear model* because the output (± 1) is a (thresholded) linear function of the input. For linear functions, we can draw the decision boundary as a linear function:



Find set of x for which $\theta^\top x = 0$ and this is the decision boundary, which will be linear.

For *multiclass* problems, replace logistic with *soft-max*. For each of the K classes, let $\theta_{k,:}$ be a weight vector. Then, set:

$$p_{\theta}(y | x) = \frac{\exp[\theta_y^\top x]}{\sum_k \exp[\theta_k^\top x]}$$

For a two class problem, this reduces to the logistic (with a small adjustment of parameters). Usually fix $\theta_{0,:} = 0$ for identifiability.

Compute log-likelihood and gradient:

$$\begin{aligned}\ell(\theta) &= \log \prod_n p_\theta(y_n | x_n) \\ &= \sum_n \left\{ \theta_{y_n}^\top x_n - \log \sum_k \exp[\theta_k^\top x_n] \right\} \\ \nabla_\theta \ell &= \sum_n \left\{ x_n - \frac{1}{\sum_k \exp[\theta_k^\top x_n]} \sum_k x_n \exp[\theta_k^\top x_n] \right\}\end{aligned}$$

Alternative formulation: maximum entropy.

(Conditional) Entropy of a distribution p is $H(p) = - \int dx \int dy p(x) \log p(x)$. High entropy means “random.”

Want distribution p that is maximally entropic, except that it agrees with the data with respect to (conditional) feature expectations; let \tilde{p} be the empirical distribution. Leads to identical model.

These model only the conditional probability of labels $p(y|x)$ and not the full data $p(x,y)$. We can factorize the latter as $p(y)p(x|y)$ in a *generative* fashion.

$$p(\text{data}) = \prod_n p(y_n)p(x_n | y_n)$$

For binary, $p(y)$ can be bernoulli; for multiclass, multinomial. In either case, parameterize by π . $p(x|y)$ is often more complicated. Suppose there are F -many features; then:

$$p(x | y) = p(x_1 | y)p(x_2 | y, x_1) \cdots p(x_F | y, x_{1:F-1}) = \prod_f p(x_f | y, x_{1:f-1})$$

We approximate this with the *naive Bayes* assumption of feature independence (conditional on y):

$$p(x | y) = \prod_f p(x_f | y, x_{1:f-1}) \approx \prod_f p(x_f | y)$$

When features are binary, make each $p(x_f | y)$ another bernoulli:

$$p(x_f | y; \theta) = \theta_f^{x_f} (1 - \theta_f)^{1-x_f}$$

Putting this together:

$$p(\text{data}) = \prod_n \pi^{y_n} (1 - \pi)^{1-y_n} \prod_f \theta_{y_n, f}^{x_{n,f}} (1 - \theta_{y_n, f})^{1-x_{n,f}}$$

Exercise: verify that there exist parameters π, θ for the naive Bayes model that mimick the logistic regression behavior (and vice-versa).

Change of pace. . .

Making things work:

1. Scale/translation of features interacts with regularization. Can “whiten” (or bound) data to fix this.
2. Output of regression can also be centered. (Reduces need for “bias feature.”)
3. Some interactions are exponential: take log of features.

Doing good experiments:

1. Often need to choose “hyperparameters.” Can use k -fold cross-validation, LOO xval or use a development/validation set. Typically $k = 10$.
2. Performance on development data better indicator of performance on test.
3. Never look at your development/test data. I mean it: never.
4. When breaking data into train/dev/test, be careful.