COMS 4721: Machine Learning for Data Science

Lecture 9, 2/16/2017

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

LOGISTIC REGRESSION

Linear classifiers

Given: Data $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$

A **linear classifier** takes a vector $w \in \mathbb{R}^d$ and scalar $w_0 \in \mathbb{R}$ and predicts

$$y_i = f(x_i; w, w_0) = \text{sign}(x_i^T w + w_0).$$

We discussed two methods last time:

- Least squares: Sensitive to outliers
- Perceptron: Convergence issues, assumes linear separability

Can we combine the separating hyperplane idea with probability to fix this?

In this lecture, we're going to see how we can extend the ideas from the perceptron combining it with probability to fix some of the issues from the perceptron

Linear discriminant analysis

We saw an example of a linear classification rule using a Bayes classifier.

For the model $y \sim \text{Bern}(\pi)$ and $x \,|\, y \sim N(\mu_y, \Sigma)$, declare $y = 1$ given x if

$$\ln \frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)} > 0\,.$$

In this case, the *log odds* is equal to

$$\ln \frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)} = \underbrace{\ln \frac{\pi_1}{\pi_0} - \frac{1}{2}(\mu_1+\mu_0)^T\Sigma^{-1}(\mu_1-\mu_0)}_{\text{a constant } w_0} + x^T \underbrace{\Sigma^{-1}(\mu_1-\mu_0)}_{\text{a vector } w}$$

Original formulation

we want to do define a probability on a label for a particular observation

Recall that originally we wanted to declare $y = 1$ given x if

$$\ln \frac{p(y=1|x)}{p(y=0|x)} > 0$$

We didn't have a way to define $p(y|x)$, so we used Bayes rule:

- Use $p(y|x) = \frac{p(x|y)p(y)}{p(x)}$ and let the $p(x)$ cancel each other in the fraction
- Define $p(y)$ to be a Bernoulli distribution (coin flip distribution)
- Define $p(x|y)$ however we want (e.g., a single Gaussian)

Now, we want to directly define $p(y|x)$. We'll use the log odds to do this.

We're going to start from the log odds. And rather than define these probability distributions on a label conditioned on the covariate, we're going to start by defining log odds directly.

* To see how this equivalent to the Bayes classifier using the log odds:

From Bayes classifier: The probability of label given the covariates is equal to the probability of covariates given the label times the probability of the label divided by the marginal probability of the covariates.

In *, if we were to calculate the ratio for $y=1/y=0$, this * denominator cancels, and we have exactly what we have for log odds of the bayes's classifier.

Log odds and Bayes classification

[Motivate how we might directly define the log odds and from that definition, then be able to define a prob. distribution

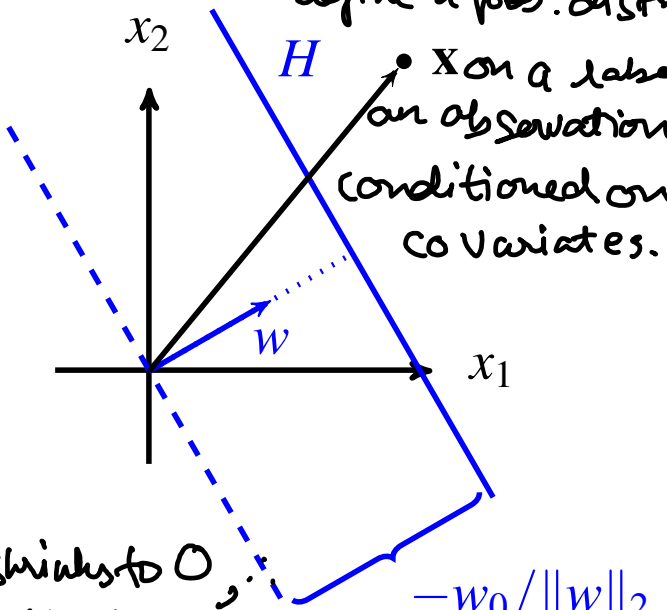
Log odds and hyperplanes

Classifying x based on the log odds

$$L = \ln \frac{p(y=+1|x)}{p(y=-1|x)},$$

we notice that

1. $L \gg 0$: more confident $y = +1$,
2. $L \ll 0$: more confident $y = -1$,
3. $L = 0$: can go either way



x on a label for an observation conditioned on its covariates.

[The ratio shrinks to 0, the log of that becomes more and more negative.]

prob of label +1 = prob. of label -1 given x. $\ln(1) = 0$

The linear function $x^T w + w_0$ captures these three objectives:

- The distance of x to a hyperplane H defined by (w, w_0) is $|\frac{x^T w}{\|w\|_2} + \frac{w_0}{\|w\|_2}|$.
- The sign of the function captures which side x is on.
- As x moves away/towards H, we become more/less confident.

↳ more +ve (dir^n of $\vec{w}$)
↳ more -ve (oppp. of $\vec{w}$)

LOG ODDS AND HYPERPLANES

So we already have a function that has the properties that we desire. And we start from there when defining log odds.

Logistic link function

We can directly plug in the hyperplane representation for the log odds:

$$\ln \frac{p(y=+1|x)}{p(y=-1|x)} = x^T w + w_0$$

Question: What is different from the previous Bayes classifier?

Answer: There was a formula for calculating w and w_0 based on the prior model and data x. Now, we put no restrictions on these values.

? How do we get estimates then?

Setting $p(y=-1|x) = 1 - p(y=+1|x)$, solve for $p(y=+1|x)$ to find

$$p(y=+1|x) = \frac{\exp\{x^T w + w_0\}}{1+\exp\{x^T w + w_0\}} = \sigma(x^T w + w_0)$$

(*)

↑ we've come up with a way to directly define the prob. that x is in class 1 given the parameters, $\vec{w}$ & w_0

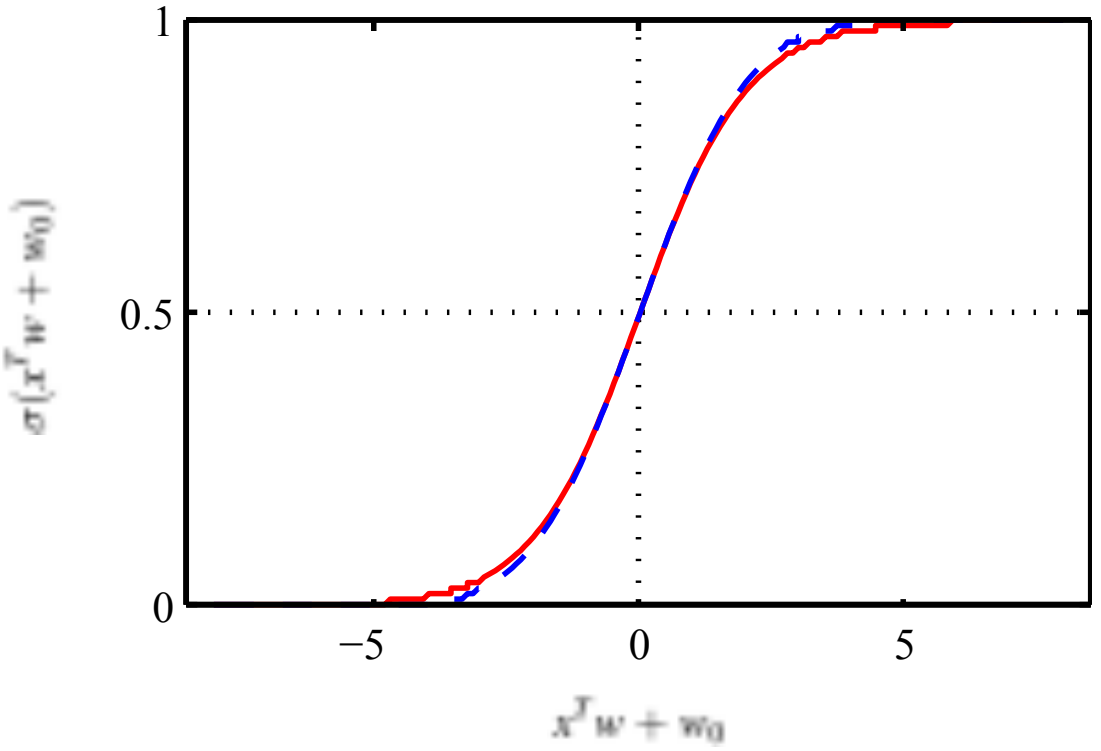
- This is called the *sigmoid function*.
- We have chosen $x^T w + w_0$ as the *link function* for the log odds.

* ① more +ve as we're confident, that's it's on +1 side,
numerator → ∞, denominator → same no. +1
ratio → 1

② more -ve,
other side, numerator → 0, denominator → 1

ratio → 0

LOGISTIC SIGMOID FUNCTION



more +ve, more confident +1 label.

more −ve, more confident −1 label.

- Red line: Sigmoid function $\sigma(x^T w + w_0)$, which maps x to $p(y = +1|x)$.
- The function $\sigma(\cdot)$ captures our desire to be more confident as we move away from the separating hyperplane, defined by the x-axis.
- (Blue dashed line: On a later slide.)

LOGISTIC REGRESSION

As with regression, absorb the offset: $w \leftarrow \begin{bmatrix} w_0 \\ w \end{bmatrix}$ and $x \leftarrow \begin{bmatrix} 1 \\ x \end{bmatrix}$.

Definition

Let $(x_1, y_1), \ldots, (x_n, y_n)$ be a set of binary labeled data with $y \in \{-1, +1\}$.

Logistic regression models each y_i as independently generated, with

$$P(y_i = +1 | x_i, w) = \sigma(x_i^T w), \quad \sigma(x_i; w) = \frac{e^{x_i^T w}}{1 + e^{x_i^T w}}.$$

Discriminative vs Generative classifiers

- This is a *discriminative* classifier because x is not directly modeled.
- Bayes classifiers are known as *generative* because x is modeled.

Discriminative: $p(y|x)$ Generative: $p(x|y)p(y)$.

* We have no distribution on x anymore. We've only defined a distribution on y. The Bayes classifier, as we've discussed previously are generative classifiers, because we 1st define a distribution on y, and then we define a class specific distribution on x given y.

So here we are back to having no distribution on x. We only define the distribution on y. So in that sense, the logistic regression model can be viewed as a discriminative classifier.

LOGISTIC REGRESSION LIKELIHOOD

We've defined probability distributions on our data that implicitly define a joint likelihood of the data.

Data likelihood

Define $\sigma_i(w) = \sigma(x_i^T w)$. The joint likelihood of $y_1, \ldots, y_n$ is

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n, w) = \prod_{i=1}^{n} p(y_i | x_i, w)$$

independence assumption.

is a discrete prob. on 2 events.

$$= \prod_{i=1}^{n} \sigma_i(w)^{\mathbb{1}(y_i=+1)} \left(1 - \sigma_i(w)\right)^{\mathbb{1}(y_i=-1)}$$

indicator

is picks out the correct event for the true value

For true value, we have indicator equal to 1, for wrong value, it will be equal to 0.

we've now defined a joint likelihood on all of the labels given all of the covariates and regression coefficient vector w.

- Notice that each x_i modifies the probability of success for its y_i.
- Predicting new data is the same:
  - If $x^T w > 0$, then $\sigma(x^T w) > 1/2$ and predict $y = +1$, and vice versa.
  - We now get a confidence in our prediction via the probability $\sigma(x^T w)$.

Build in the label into the sigmoid function itself.

More notation changes

* → feed as the argument the label times the dot product.

Use the following fact to condense the notation:

$$\underbrace{\frac{e^{y_i x_i^T w}}{1+e^{y_i x_i^T w}}}_{\sigma_i(y_i \cdot w)^*} = \left(\underbrace{\frac{e^{x_i^T w}}{1+e^{x_i^T w}}}_{\sigma_i(w)}\right)^{\mathbb{1}(y_i=+1)} \left(\underbrace{1-\frac{e^{x_i^T w}}{1+e^{x_i^T w}}}_{1-\sigma_i(w)}\right)^{\mathbb{1}(y_i=-1)}$$

$y_i = +1$ $\quad$ $y_i = -1$

therefore, the data likelihood can be written compactly as

joint likelihood = product of sigmoid functions, where we also multiply the argument of the sigmoid function by label, +1 / -1.

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n, w) = \prod_{i=1}^{n} \sigma_i(y_i \cdot w)$$

We want to maximize this over w.

We've defined the model, likelihood terms of data, given the parameters of the model.

Maximum likelihood

The maximum likelihood solution for w can be written

$$\begin{aligned} w_{\text{ML}} &= \arg\max_w \sum_{i=1}^{n} \ln \sigma_i(y_i \cdot w) \\ &= \arg\max_w \mathcal{L} \end{aligned}$$

why?

As with the Perceptron, we can't directly set $\nabla_w \mathcal{L} = 0$, so we need an iterative algorithm. At step t, we can update

Gradient descent: dir = to move to update our objective fn.

$$w^{(t+1)} = w^{(t)} + \eta \nabla_w \mathcal{L},$$

Gradient of objective function evaluated at the current value

$$\nabla_w \mathcal{L} = \sum_{i=1}^{n} (1 - \sigma_i(y_i \cdot w))\, y_i x_i.$$

btw 0 & 1 ; +1/-1

We will see that this results in an algorithm similar to the Perceptron.

We're summing each of these data specific vectors to get a gradient.

So this simply flipping direction of x_i, which is a vector.

LOGISTIC REGRESSION ALGORITHM (STEEPEST ASCENT)

Input: Training data $(x_1, y_i), \ldots, (x_n, y_n)$ and step size $\eta > 0$

1. **Set** $w^{(1)} = \vec{0}$
2. **For step** $t = 1, 2, \ldots$ **do**
   - Update $w^{(t+1)} = w^{(t)} + \eta \sum_{i=1}^{n} \left(1 - \sigma_i(y_i \cdot w^{(t)})\right) y_i x_i$ *

Perceptron: Search for misclassified (x_i, y_i), update $w^{(t+1)} = w^{(t)} + \eta y_i x_i$.

Logistic regression: Something similar except we sum over all data.

- Recall that $\sigma_i(y_i \cdot w)$ picks out the probability assigned to the observed y_i.
- Therefore $1 - \sigma_i(y_i \cdot w)$ is the probability assigned to the *wrong* value.
- Perceptron is "all-or-nothing." Either it's correctly or incorrectly classified.
- Logistic regression has a probability "fudge-factor."

what is the probabilistic interpretation?

So if our data labeled instance is +1 but
our model given the current vector W is very confident in predicting -1.
Then this is going to be very close to 0,
in which case we'll have something very close to 1 here.
On the other side, if our labeled instance is, say +1 and
our model is very confident in predicting +1 for that instance.
This will be very close to 1 and
then we'll have something that's essentially equal to 0.
So this value here that's pre-multiplying y times X
is like a measure of how confident we are in our prediction.

BAYESIAN LOGISTIC REGRESSION

Problem: If a hyperplane can separate all training data, then $\|w_{\text{ML}}\|_2 \to \infty$. This drives $\sigma_i(y_i \cdot w) \to 1$ for each (x_i, y_i).

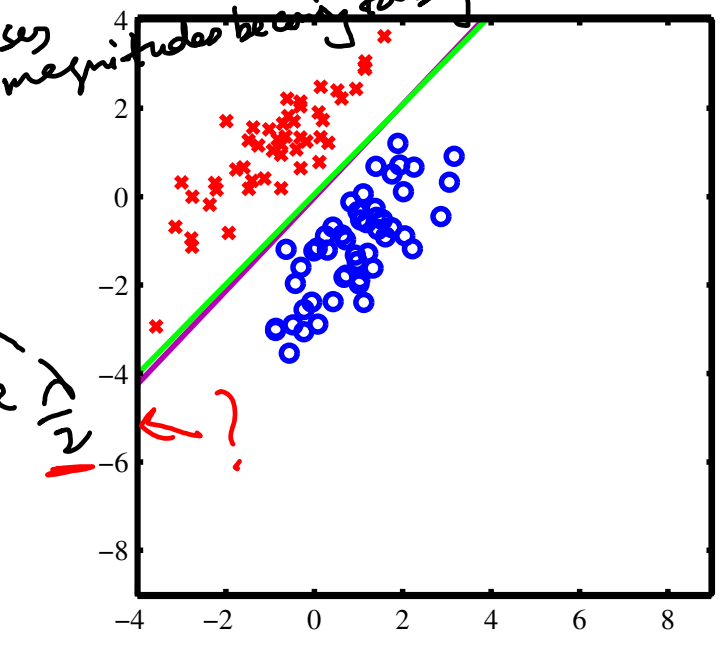
Even for nearly separable data it might get a few very wrong in order to be more confident about the rest. This is a case of "over-fitting."

A solution: Regularize w with $\lambda w^T w$:

$$w_{\text{MAP}} = \arg\max_w \sum_{i=1}^n \ln \sigma_i(y_i \cdot w) - \lambda w^T w$$

We've seen how this corresponds to a Gaussian prior distribution on w.

How about the posterior $p(w|x, y)$?



Because as W goes to infinity,
it's going to make all of the data on the plus side.
It's going to make the dot product between all the data
on the plus side with that W become positive infinity.
And it's going to meet the dot product of all the data on the negative side with
that vector W become minus infinity.
And in that sense we can maximize the likelihood, because we can
predict that all the data on the +1 side are gonna be +1 with probability 1 and
all of the data on the -1 side are going to be -1 with probability 1.
So there's no reason to have any uncertainty to
use the uncertainty that the model gives us if we can do everything perfectly.

We've defined a Gaussian prior on the coefficient vector w.
We can do map for that, but can we do Bayesian inference?
Can we define the posterior of w because we have defined
likelihood and a prior?
So the answer we'll see is no we can't calculate the posterior in
closed form, which is going to motivate something called Laplace
approximation.

LAPLACE APPROXIMATION

Posterior calculation

Define the prior distribution on w to be $w \sim N(0, \lambda^{-1} I)$. The posterior is

$$p(w|x, y) = \frac{p(w) \prod_{i=1}^{n} \sigma_i(y_i \cdot w)}{\int p(w) \prod_{i=1}^{n} \sigma_i(y_i \cdot w)\, dw}$$

This is not a "standard" distribution and we can't calculate the denominator.

Therefore we can't actually say what $p(w|x, y)$ is.

Can we approximate $p(w|x, y)$?

Laplace approximation

One strategy

Pick a distribution to approximate $p(w|x, y)$. We will say

$$p(w|x, y) \approx \text{Normal}(\mu, \Sigma).$$

Now we need a method for setting μ and Σ.

Laplace approximations

Using a condensed notation, notice from Bayes rule that

$$p(w|x, y) = \frac{e^{\ln p(y,w|x)}}{\int e^{\ln p(y,w|x)} dw}.$$

We will approximate $\ln p(y, w|x)$ in the numerator and denominator.

LAPLACE APPROXIMATION

Let's define $f(w) = \ln p(y, w|x)$.

Taylor expansions

We can approximate $f(w)$ with a **second order Taylor expansion**.

Recall that $w \in \mathbb{R}^{d+1}$. For any point $z \in \mathbb{R}^{d+1}$,

$$f(w) \approx f(z) + (w-z)^T \nabla f(z) + \frac{1}{2}(w-z)^T \left(\nabla^2 f(z)\right)(w-z)$$

The notation $\nabla f(z)$ is short for $\nabla_w f(w)|_z$, and similarly for the matrix of second derivatives. We just need to pick z.

The Laplace approximation defines $z = w_{\text{MAP}}$.

LAPLACE APPROXIMATION (SOLVING)

Recall $f(w) = \ln p(y, w|x)$ and $z = w_{\text{MAP}}$. From Bayes rule and the Laplace approximation we now have

$$\begin{aligned} p(w|x,y) &= \frac{\mathrm{e}^{f(w)}}{\int \mathrm{e}^{f(w)} dw} \\ &\approx \frac{\mathrm{e}^{f(z)+(w-z)^T \nabla f(z)+\frac{1}{2}(w-z)^T \left(\nabla^2 f(z)\right)(w-z)}}{\int \mathrm{e}^{f(z)+(w-z)^T \nabla f(z)+\frac{1}{2}(w-z)^T (\nabla^2 f(z))(w-z)} dw} \end{aligned}$$

This can be simplified in two ways,

1. The term $\mathrm{e}^{f(w_{\text{MAP}})}$ in the numerator and denominator can be viewed as a constant since it doesn't vary in w. It therefore cancels out.

2. By definition of how we find w_{MAP}, the vector $\nabla_w \ln p(y, w|x)|w_{\text{MAP}} = 0$.

LAPLACE APPROXIMATION (SOLVING)

We're therefore left with the approximation

$$p(w|x,y) \quad \approx \quad \frac{\mathrm{e}^{-\frac{1}{2}(w-w_{\mathrm{MAP}})^T\left(-\nabla^2 \ln p(y,w_{\mathrm{MAP}}|x)\right)(w-w_{\mathrm{MAP}})}}{\int \mathrm{e}^{-\frac{1}{2}(w-w_{\mathrm{MAP}})^T(-\nabla^2 \ln p(y,w_{\mathrm{MAP}}|x))(w-w_{\mathrm{MAP}})}dw}$$

The solution comes by observing that this is a multivariate normal,

$$p(w|x,y) \approx \mathrm{Normal}(\mu, \Sigma),$$

where

$$\mu = w_{\mathrm{MAP}}, \quad \Sigma = \left(-\nabla^2 \ln p(y, w_{\mathrm{MAP}}|x)\right)^{-1}$$

We can take the second derivative (Hessian) of the log joint likelihood to find

$$\nabla^2 \ln p(y, w_{\mathrm{MAP}}|x) = -\lambda I - \sum_{i=1}^{n} \sigma(y_i \cdot x_i^T w_{\mathrm{MAP}}) \left(1 - \sigma(y_i \cdot x_i^T w_{\mathrm{MAP}})\right) x_i x_i^T$$

Laplace approximation for logistic regression

Given labeled data $(x_1, y_1), \ldots, (x_n, y_n)$ and the model

$$P(y_i|x_i, w) = \sigma(y_i x_i^T w), \quad w \sim N(0, \lambda^{-1} I), \qquad \sigma(y_i x_i^T w) = \frac{e^{y_i x_i^T w}}{1 + e^{y_i x_i^T w}}$$

1. Find: $w_{\text{MAP}} = \arg\max_w \sum_{i=1}^{n} \ln \sigma(y_i x_i^T w_{\text{MAP}}) - \frac{\lambda}{2} w^T w$
2. Set: $-\Sigma^{-1} = -\lambda I - \sum_{i=1}^{n} \sigma(y_i x_i^T w_{\text{MAP}}) \left(1 - \sigma(y_i x_i^T w_{\text{MAP}})\right) x_i x_i^T$
3. Approximate: $p(w|x, y) = N(w_{\text{MAP}}, \Sigma)$.