CSC 411: Lecture 04: Logistic Regression

Class based on Raquel Urtasun & Rich Zemel's lectures

Sanja Fidler

University of Toronto

Jan 20, 2015

Today

- Key Concepts:
  - Logistic Regression
  - Regularization
  - Cross validation

(note: we are still talking about binary classification)

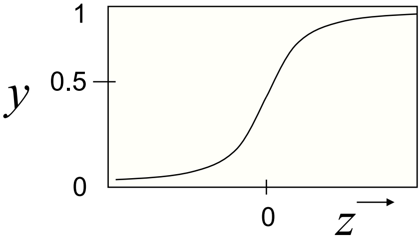
Logistic Regression

- An alternative: replace the $sign(\cdot)$ with the sigmoid or logistic function
- We assumed a particular functional form: sigmoid applied to a linear function of the data

$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

$$\sigma(z) = \frac{1}{1+\exp(-z)}$$



- The output is a smooth function of the inputs and the weights. It can be seen as a smoothed and differentiable alternative to $sign(\cdot)$

Logistic Regression

- We assumed a particular functional form: sigmoid applied to a linear function of the data

$$y(\mathbf{x}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right)$$

where the sigmoid is defined as

$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$

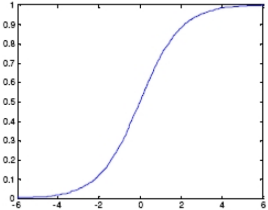
  - One parameter per data dimension (feature) and the bias
  - Features can be discrete or continuous
  - Output of the model: value $y \in [0, 1]$
  - Allows for gradient-based learning of the parameters

Shape of the Logistic Function

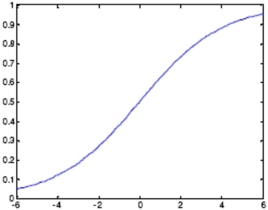
- Let's look at how modifying **w** changes the shape of the function
- 1D example:

$$y = \sigma\left(w_1 x + w_0\right)$$

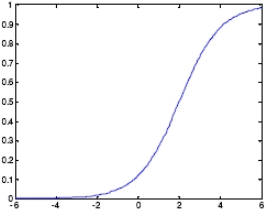
$w_0 = 0, w_1 = 1$



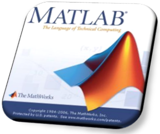
$w_0 = 0, w_1 = 0.5$



$w_0 = -2, w_1 = 1$



- Demo



Probabilistic Interpretation

- If we have a value between 0 and 1, let's use it to model class probability

$$p(C=0|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0) \quad \text{with} \quad \sigma(z) = \frac{1}{1+\exp(-z)}$$

- Substituting we have

$$p(C=0|\mathbf{x}) = \frac{1}{1+\exp\left(-\mathbf{w}^T\mathbf{x} - w_0\right)}$$

- Suppose we have two classes, how can I compute $p(C=1|\mathbf{x})$?
- Use the marginalization property of probability

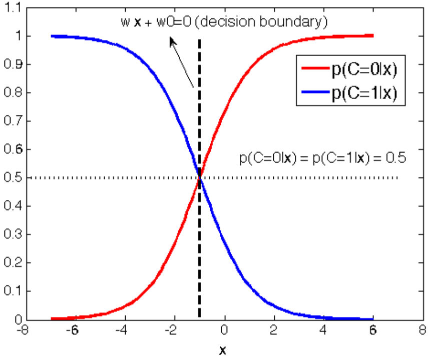
$$p(C=1|\mathbf{x}) + p(C=0|\mathbf{x}) = 1$$

- Thus (show matlab)

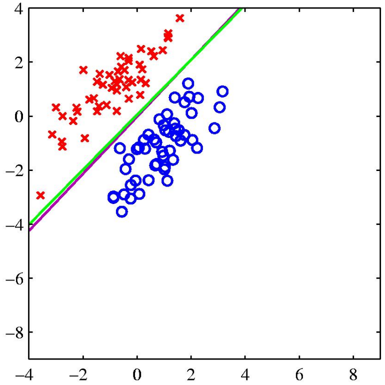
$$p(C=1|\mathbf{x}) = 1 - \frac{1}{1+\exp\left(-\mathbf{w}^T\mathbf{x} - w_0\right)} = \frac{\exp(-\mathbf{w}^T\mathbf{x} - w_0)}{1+\exp\left(-\mathbf{w}^T\mathbf{x} - w_0\right)}$$

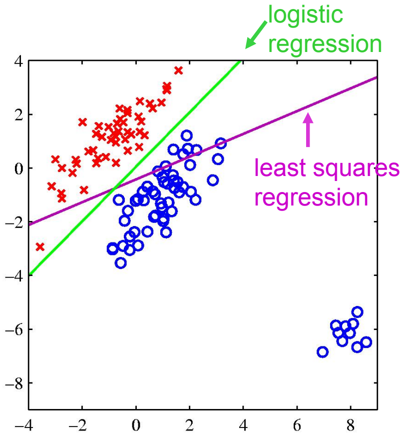
Decision Boundary for Logistic Regression

- What is the decision boundary for logistic regression?
- $p(C=1|\mathbf{x},\mathbf{w}) = p(C=0|\mathbf{x},\mathbf{w}) = 0.5$
- $p(C=0|\mathbf{x},\mathbf{w}) = \sigma\left(\mathbf{w}^T\mathbf{x} + w_0\right) = 0.5$, where $\sigma(z) = \frac{1}{1+\exp(-z)}$
- Decision boundary: $\mathbf{w}^T\mathbf{x} + w_0 = 0$
- Logistic regression has a **linear decision boundary**



Logistic Regression vs Least Squares Regression





If the right answer is 1 and the model says 1.5, it loses, so it changes the boundary to avoid being "too correct" (tilts away from outliers)

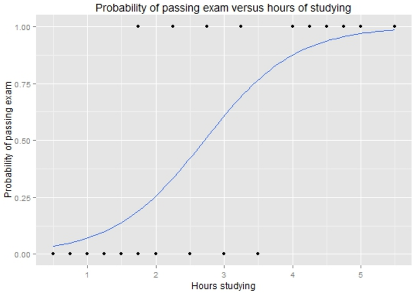
33

Example

- **Problem**: Given the number of hours a student spent learning, will (s)he pass the exam?
- Training data (top row: $\mathbf{x}^{(i)}$, bottom row: $t^{(i)}$)

Hours	0.50	0.75	1.00	1.25	1.50	1.75	1.75	2.00	2.25	2.50	2.75	3.00	3.25	3.50	4.00	4.25	4.50	4.75	5.00	5.50
Pass	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1	1	1	1

- Learn $\mathbf{w}$ for our model, i.e. logistic regression (coming up)
- Make predictions:



Hours of study	Probability of passing exam
1	0.07
2	0.26
3	0.61
4	0.87
5	0.97

Learning?

- How should we learn the weights $\mathbf{w}, w_0$?
- We have a probabilistic model
- Let's use maximum likelihood

(simplify notation: we will write $\mathbf{w}$ to represent both $\mathbf{w}$ and w_0)

Conditional Likelihood

- Assume $t \in \{0, 1\}$, we can write the probability distribution of each of our training points $p(t^{(1)}, \cdots, t^{(N)} | \mathbf{x}^{(1)}, \cdots \mathbf{x}^{(N)}; \mathbf{w})$
- Assuming that the training examples are sampled IID: independent and identically distributed, we can write the *likelihood function*:

$$L(\mathbf{w}) = p(t^{(1)}, \cdots, t^{(N)} | \mathbf{x}^{(1)}, \cdots \mathbf{x}^{(N)}; \mathbf{w}) = \prod_{i=1}^{N} p(t^{(i)} | \mathbf{x}^{(i)}; \mathbf{w})$$

- We can write each probability as (will be useful later):

$$\begin{aligned} p(t^{(i)} | \mathbf{x}^{(i)}; \mathbf{w}) &= p(C = 1 | \mathbf{x}^{(i)}; \mathbf{w})^{t^{(i)}} p(C = 0 | \mathbf{x}^{(i)}; \mathbf{w})^{1 - t^{(i)}} \\ &= \left(1 - p(C = 0 | \mathbf{x}^{(i)}; \mathbf{w})\right)^{t^{(i)}} p(C = 0 | \mathbf{x}^{(i)}; \mathbf{w})^{1 - t^{(i)}} \end{aligned}$$

- We can learn the model by maximizing the likelihood

$$\max_{\mathbf{w}} L(\mathbf{w}) = \max_{\mathbf{w}} \prod_{i=1}^{N} p(t^{(i)} | \mathbf{x}^{(i)}; \mathbf{w})$$

- Easier to maximize the log likelihood $\log L(\mathbf{w})$

Loss Function

$$
\begin{aligned}
L(\mathbf{w}) &= \prod_{i=1}^{N} p(t^{(i)}|\mathbf{x}^{(i)}) \qquad \text{(likelihood)} \\
&= \prod_{i=1}^{N} \left(1 - p(C=0|\mathbf{x}^{(i)})\right)^{t^{(i)}} p(C=0|\mathbf{x}^{(i)})^{1-t^{(i)}}
\end{aligned}
$$

- We can convert the maximization problem into minimization so that we can write the loss function:

$$
\begin{aligned}
\ell_{log}(\mathbf{w}) &= -\log L(\mathbf{w}) \\
&= -\sum_{i=1}^{N} \log p(t^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) \\
&= -\sum_{i=1}^{N} t^{(i)} \log(1 - p(C=0|\mathbf{x}^{(i)}, \mathbf{w})) - \sum_{i=1}^{N} (1 - t^{(i)}) \log p(C=0|\mathbf{x}^{(i)}; \mathbf{w})
\end{aligned}
$$

- Is there a closed form solution?
- It's a convex function of **w**. Can we get the global optimum?

Gradient Descent

$$\min_{\mathbf{w}} \ell(\mathbf{w}) = \min_{\mathbf{w}} \left\{ -\sum_{i=1}^{N} t^{(i)} \log(1 - p(C = 0|\mathbf{x}^{(i)}, \mathbf{w})) - \sum_{i=1}^{N} (1 - t^{(i)}) \log p(C = 0|\mathbf{x}^{(i)}, \mathbf{w}) \right\}$$

- Gradient descent: iterate and at each iteration compute steepest direction towards optimum, move in that direction, step-size λ

$$w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda \frac{\partial \ell(\mathbf{w})}{\partial w_j}$$

- You can write this in vector form

$$\nabla \ell(\mathbf{w}) = \left[\frac{\partial \ell(\mathbf{w})}{\partial w_0}, \cdots, \frac{\partial \ell(\mathbf{w})}{\partial w_k} \right]^T, \qquad \text{and} \qquad \triangle(\mathbf{w}) = -\lambda \nabla \ell(\mathbf{w})$$

- But where is $\mathbf{w}$?

$$p(C = 0|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x} - w_0)}, \quad p(C = 1|\mathbf{x}) = \frac{\exp(-\mathbf{w}^T\mathbf{x} - w_0)}{1 + \exp(-\mathbf{w}^T\mathbf{x} - w_0)}$$

Let's Compute the Updates

- The loss is

$$\ell_{log-loss}(\mathbf{w}) = -\sum_{i=1}^{N} t^{(i)} \log p(C=1|\mathbf{x}^{(i)}, \mathbf{w}) - \sum_{i=1}^{N} (1-t^{(i)}) \log p(C=0|\mathbf{x}^{(i)}, \mathbf{w})$$

  where the probabilities are

$$p(C=0|\mathbf{x},\mathbf{w}) = \frac{1}{1+\exp(-z)} \qquad p(C=1|\mathbf{x},\mathbf{w}) = \frac{\exp(-z)}{1+\exp(-z)}$$

  and $z = \mathbf{w}^T\mathbf{x} + w_0$

- We can simplify

$$\begin{aligned}
\ell(\mathbf{w})_{log-loss} &= \sum_i t^{(i)} \log(1+\exp(-z^{(i)})) + \sum_i t^{(i)} z^{(i)} + \sum_i (1-t^{(i)}) \log(1+\exp(-z^{(i)})) \\
&= \sum_i \log(1+\exp(-z^{(i)})) + \sum_i t^{(i)} z^{(i)}
\end{aligned}$$

- Now it's easy to take derivatives

Updates

$$\ell(\mathbf{w}) = \sum_i t^{(i)} z^{(i)} + \sum_i \log(1 + \exp(-z^{(i)}))$$

- Now it's easy to take derivatives
- Remember $z = \mathbf{w}^T \mathbf{x} + w_0$

$$\frac{\partial \ell}{\partial w_j} = \sum_i \left(t^{(i)} x_j^{(i)} - x_j^{(i)} \cdot \frac{\exp(-z^{(i)})}{1 + \exp(-z^{(i)})} \right)$$

- What's $x_j^{(i)}$? The $j-$th dimension of the $i-$th training example $\mathbf{x}^{(i)}$
- And simplifying

$$\frac{\partial \ell}{\partial w_j} = \sum_i x_j^{(i)} \left(t^{(i)} - p(C = 1 | \mathbf{x}^{(i)}; \mathbf{w}) \right)$$

- Don't get confused with indices: j for the weight that we are updating and i for the training example

Gradient Descent

- Putting it all together (plugging the update into gradient descent):

Gradient descent for logistic regression:

$$w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda \sum_i x_j^{(i)} \left(t^{(i)} - p(C = 1|\mathbf{x}^{(i)}; \mathbf{w}) \right)$$

where:

$$p(C = 1|\mathbf{x}^{(i)}; \mathbf{w}) = \frac{\exp(-\mathbf{w}^T\mathbf{x} - w_0)}{1 + \exp(-\mathbf{w}^T\mathbf{x} - w_0)} = \frac{1}{1 + \exp(\mathbf{w}^T\mathbf{x} + w_0)}$$

- This is all there is to learning in logistic regression. Simple, huh?

Regularization

- We can also look at

$$p(\mathbf{w}|\{t\},\{\mathbf{x}\}) \propto p(\{t\}|\{\mathbf{x}\},\mathbf{w})\, p(\mathbf{w})$$

with $\{t\} = (t^{(1)}, \cdots, t^{(N)})$, and $\{\mathbf{x}\} = (\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)})$

- We can define priors on parameters $\mathbf{w}$
- This is a form of regularization
- Helps avoid large weights and overfitting

$$\max_{\mathbf{w}} \log \left[p(\mathbf{w}) \prod_i p(t^{(i)}|\mathbf{x}^{(i)},\mathbf{w}) \right]$$

- What's $p(\mathbf{w})$?

Regularized Logistic Regression

- For example, define prior: normal distribution, zero mean and identity covariance $p(\mathbf{w}) = \mathcal{N}(0, \alpha^{-1}\mathbf{I})$
- This prior pushes parameters towards zero
- Including this prior the new gradient is

$$w_j^{(t+1)} \leftarrow w_j^{(t)} - \lambda \frac{\partial \ell(\mathbf{w})}{\partial w_j} - \lambda \alpha w_j^{(t)}$$

  where t here refers to iteration of the gradient descent

- The parameter α is the importance of the regularization, and it's a hyper-parameter
- How do we decide the best value of α (or a hyper-parameter in general)?

Use of Validation Set

Tuning hyper-parameters:

- **Never use test data for tuning the hyper-parameters**
- We can divide the set of training examples into two disjoint sets: **training** and **validation**
- Use the first set (i.e., training) to estimate the weights $\mathbf{w}$ for different values of α
- Use the second set (i.e., validation) to estimate the best α, by evaluating how well the classifier does on this second set
- This tests how well it generalizes to unseen data

Cross-Validation

- Leave-p-out cross-validation:
  - We use p observations as the validation set and the remaining observations as the training set.
  - This is repeated on all ways to cut the original training set.
  - It requires $\mathcal{C}_n^p$ for a set of n examples
- Leave-1-out cross-validation: When $p = 1$, does not have this problem
- k-fold cross-validation:
  - The training set is randomly partitioned into k equal size subsamples.
  - Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k - 1$ subsamples are used as training data.
  - The cross-validation process is then repeated k times (the folds).
  - The k results from the folds can then be averaged (or otherwise combined) to produce a single estimate

Logistic Regression wrap-up

Advantages:

- Easily extended to multiple classes (thoughts?)
- Natural probabilistic view of class predictions
- Quick to train
- Fast at classification
- Good accuracy for many simple data sets
- Resistant to overfitting
- Can interpret model coefficients as indicators of feature importance

Less good:

- Linear decision boundary (too simple for more complex problems?)

[Slide by: Jeff Howbert]