Logistic Regression

David Helmbold

University of California, Santa Cruz
dph@soe.ucsc.edu

S'15. modified F'15, W'15

Logistic Regression learns parameter vector θ (or $\mathbf{w}$)

Idea (inductive bias) behind 2-class Logistic Regression:

- two labels, so $y \in \{0, 1\}$ (binary classification, not regression)
- discriminative models give $p(y = 1 \mid \mathbf{x}; \theta)$
- labels softly separated by hyperplane
- maximum confusion at hyperplane:
  when $\theta^\intercal \mathbf{x} = 0$ then $p(y = 1 \mid \mathbf{x}; \theta) = 1/2$
- use "add a dimension" trick to "shift" hyperplane off 0
- assume $p(y = 1 \mid \mathbf{x}; \theta)$ is some $g(\theta \cdot \mathbf{x})$

What properties of $g(\theta \cdot \mathbf{x}) = g(\theta^\intercal \mathbf{x}) = p(y = 1 \mid \mathbf{x}; \theta)$ do we want?

- $g(-\infty) = 0$

What properties of $g(\theta \cdot \mathbf{x}) = g(\theta^\intercal \mathbf{x}) = p(y = 1 \mid \mathbf{x}; \theta)$ do we want?

- $g(-\infty) = 0$
- $g(\infty) = 1$

What properties of $g(\theta \cdot \mathbf{x}) = g(\theta^\intercal \mathbf{x}) = p(y = 1 \mid \mathbf{x}; \theta)$ do we want?

- $g(-\infty) = 0$
- $g(\infty) = 1$
- $g(0) = 1/2$

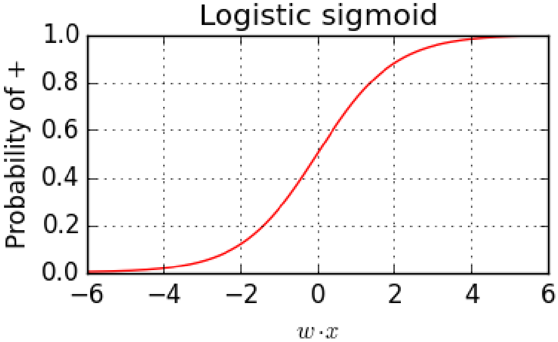
What properties of $g(\theta \cdot \mathbf{x}) = g(\theta^\intercal \mathbf{x}) = p(y = 1 \mid \mathbf{x}; \theta)$ do we want?

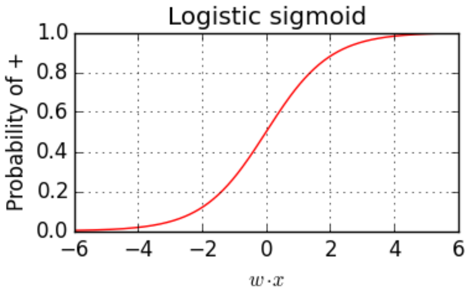
- $g(-\infty) = 0$
- $g(\infty) = 1$
- $g(0) = 1/2$
- confidence of label increases as move away from boundary, so $g(\theta^\intercal \mathbf{x})$ is monotonically increasing

What properties of $g(\theta \cdot \mathbf{x}) = g(\theta^\intercal \mathbf{x}) = p(y = 1 \mid \mathbf{x}; \theta)$ do we want?

- $g(-\infty) = 0$
- $g(\infty) = 1$
- $g(0) = 1/2$
- confidence of label increases as move away from boundary, so $g(\theta^\intercal \mathbf{x})$ is monotonically increasing
- $g(-a) = 1 - g(a)$ (symmetry, implies $g(0) = 1/2$)

- $g(-\infty) = 0$
- $g(\infty) = 1$
- $g(-a) = 1 - g(a)$ (symmetry, implies $g(0) = 1/2$)





$$\overbrace{h_\theta(\mathbf{x})}^{y \text{ in Bishop}} = g(\theta \cdot \mathbf{x}) = \frac{1}{1 + \exp(-\theta \cdot \mathbf{x})} = \frac{\exp(\theta \cdot \mathbf{x})}{1 + \exp(\theta \cdot \mathbf{x})}$$

$$P(0 \mid \mathbf{x}; \theta) = 1 - h_\theta(\mathbf{x}) = \frac{\exp(-\theta \cdot \mathbf{x})}{1 + \exp(-\theta \cdot \mathbf{x})} = \frac{1}{1 + \exp(\theta \cdot \mathbf{x})}$$

Derivative of $g()$ simple: $g'(a) = g(a)(1 - g(a))$

Logistic Regression finds a maximum likelihood estimator for the data.
Likelihood of model θ is probability of getting the m training labels.

$$L(\theta) = p(\vec{y} \mid X; \theta)$$

Logistic Regression finds a maximum likelihood estimator for the data.
Likelihood of model θ is probability of getting the m training labels.

$$\begin{aligned} L(\theta) &= p(\vec{y} \mid X; \theta) \\ &= \prod_{i=1}^{m} p(y_i \mid \mathbf{x}_i; \theta) \end{aligned}$$

Logistic Regression finds a maximum likelihood estimator for the data. Likelihood of model θ is probability of getting the m training labels.

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X; \theta) \\
&= \prod_{i=1}^{m} p(y_i \mid \mathbf{x}_i; \theta) \\
&= \prod_{i=1}^{m} \underbrace{p(y = 1 \mid \mathbf{x}_i; \theta)^{y_i} \; p(y = 0 \mid \mathbf{x}_i; \theta)^{1-y_i}}_{\text{encodes if-test on } y}
\end{aligned}
$$

Logistic Regression finds a maximum likelihood estimator for the data.
Likelihood of model θ is probability of getting the m training labels.

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X; \theta) \\
&= \prod_{i=1}^{m} p(y_i \mid \mathbf{x}_i; \theta) \\
&= \prod_{i=1}^{m} \underbrace{p(y=1 \mid \mathbf{x}_i; \theta)^{y_i} \; p(y=0 \mid \mathbf{x}_i; \theta)^{1-y_i}}_{\text{encodes if-test on } y} \\
&= \prod_{i=1}^{m} h_\theta(\mathbf{x}_i)^{y_i} (1 - h_\theta(\mathbf{x}_i))^{1-y_i}
\end{aligned}
$$

As before, log-likelihood easier:

$$\begin{aligned}\ell(\theta) &= \log(L(\theta)) \\ &= \sum_{i=1}^{m} y_i \log(h_\theta(\mathbf{x}_i)) + (1-y_i)\log(1-h_\theta(\mathbf{x}_i))\end{aligned}$$

Take derivatives for just one $(\mathbf{x}_i, y_i)$ (some algebra, uses $g'(a) = g(a)(1-g(a))$) :

$$\frac{\partial}{\partial\theta_j}\ell(\theta) = \underbrace{(y_i - h_\theta(\mathbf{x}_i))}_{\text{prediction error}} x_{i,j}$$

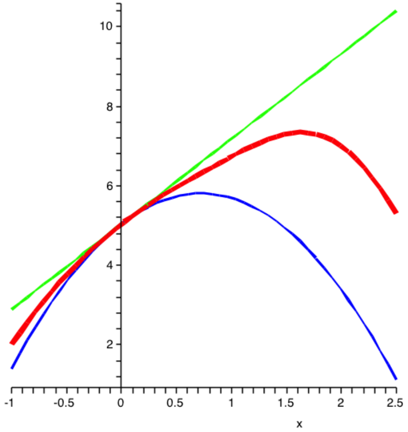
Stochastic Gradient update on $(\mathbf{x}_i, y_i)$:

$$\theta_j := \theta_j + \alpha(y_i - h_\theta(\mathbf{x}_i))x_{i,j} \qquad \text{(component } j \text{ update)}$$

$$\theta := \theta + \alpha(y_i - h_\theta(\mathbf{x}_i))\mathbf{x}_i \qquad \text{(all components)}$$

Looks similar to LMS update, but $h_\theta()$ is different.

Second Order Methods



Red - function to maximize;
Green - 1st order (linear) approx.; Blue - quadratic approximation

Second order Newton methods:

- Want to find maximum of some function $F(z)$ (e.g. log-likelihood)
- Start with initial z_0
- Use quadratic approximation:

$$F(z_0 + \delta) \approx F(z_0) + \delta F'(z_0) + \delta^2 F''(z_0)/2$$

- Maximize approximation (set derivative w.r.t. δ to 0 and solve):

$$\delta = -F'(z_0)/F''(z_0)$$

- Newton-Raphson for multiple dimensions, also called Fisher scoring, or iteratively-reweighted-least-squares (for logistic regression)
- May converge faster, but stochastic gradient descent may learn quicker (Bottou)

SoftMax for Multi-class Logistic Regression

- Learn weights θ_k for each class $k \in \{1, 2, ..., K\}$
- Class-k-ness of instance $\mathbf{x}$ is estimated by $\theta_k \cdot \mathbf{x}$
- Estimate $p(\text{Class} = k \mid \mathbf{x}; \theta_1, \ldots, \theta_K)$ for instance $\mathbf{x}$ with SoftMax function:

$$h_k(\mathbf{x}; \theta_1, \ldots, \theta_K) = \frac{\exp(\theta_k \cdot \mathbf{x})}{\sum_{r=1}^{K} \exp(\theta_r \cdot \mathbf{x})}$$

- Want weights that maximize likelihood of the sample.

- Use one-of-K encoding for labels:
  make each label $\mathbf{y}_i$ a K-vector with $y_{i,k} = 1$ if class = k
  (rest of y_i is 0).
- Likelihood of m labels in sample is:

$$
\begin{aligned}
L(\theta_1, \ldots, \theta_K) &= p(\mathbf{y}_i, \ldots, \mathbf{y}_N \mid X; \theta_1, \ldots, \theta_K) \\
&= \prod_{i=1}^{m} p(y_i \mid \mathbf{x}_i; \theta_1, \ldots, \theta_K) \\
&= \prod_{i=1}^{m} \underbrace{\prod_{k=1}^{K} h_k(\mathbf{x}_i)^{y_{i,k}}}_{p(\text{Class}|\mathbf{x}_i;\theta_1,\ldots,\theta_K)}
\end{aligned}
$$

- iterative methods maximize log likelihood (SGD, 2nd order)

θ_K is redundant! $p(\text{class} = K \mid \mathbf{x}) = 1 - \sum_{k=1}^{K-1} p(\text{class} = k \mid \mathbf{x})$.
In softmax:

$$
\begin{aligned}
h_k(\mathbf{x}; \theta_1, \ldots, \theta_K) &= \frac{\exp(\theta_k \cdot \mathbf{x})}{\sum_{r=1}^{K} \exp(\theta_r \cdot \mathbf{x})} \\
&= \frac{\exp(\theta_k \cdot \mathbf{x}) / \exp(\theta_K \cdot \mathbf{x})}{\sum_{r=1}^{K} \exp(\theta_r \cdot \mathbf{x}) / \exp(\theta_K \cdot \mathbf{x})} \\
&= \frac{\exp((\theta_k - \theta_K) \cdot \mathbf{x})}{\sum_{r=1}^{K} \exp((\theta_r - \theta_K) \cdot \mathbf{x})}
\end{aligned}
$$

so can learn $\tilde{\theta}_k = \theta_k - \theta_K$!

With modified $\tilde{\theta}_1, \ldots, \tilde{\theta}_{K-1}$:

$$h_k(\mathbf{x}; \tilde{\theta}_1, \ldots, \tilde{\theta}_{K-1}) = \begin{cases} \dfrac{\exp(\tilde{\theta}_k)}{1 + \sum_{r=1}^{K-1} \exp(\tilde{\theta}_r \cdot \mathbf{x})} & \text{if } k \leq K-1 \\[2ex] \dfrac{1}{1 + \sum_{r=1}^{K-1} \exp(\tilde{\theta}_r \cdot \mathbf{x})} & \text{if } k = K \end{cases}$$

Now looks more like 2-class case (see slide 5)

Can also learn $\tilde{\theta}_k$ directly.

Logistic Regression Summary

- Logistic regression learns weights for distribution on labels, $p(y = 1|\mathbf{x}, \theta)$
- Can use gradient descent to learn θ
- Can threshold at $\theta \cdot \mathbf{x} = 0$ to get predictions
- Extends to multi-class with soft-max
- Bonus:
  - can overfit if data linearly separable
  - $\theta \cdot \mathbf{x}$ is the log odds, $\log \frac{p(y=1|\mathbf{x};\theta)}{p(y=0|\mathbf{x};\theta)}$

Comparison

Too soon for 142 - ignore

	Fisher	LDA	Perceptron	Logistic regression	Naive Bayes
Model	LTU	$p(\mathbf{x} \mid y)$	LTU	$p(y \mid \mathbf{x})$	$p(\mathbf{x} \mid y)$
Data	numeric	numeric	numeric	numeric	mixed
Interpretable	yes	yes	yes	yes	somewhat
Missing vals?	?	no	no	no	yes
Outliers	bad	bad	fatal(*)	good	fair/poor
Irrelevant features	bad	bad	bad	bad	a little better
Monotone transform	no	no	no	no	rarely
Computation	good	good	good	good (-)	v. good