Lecture 7

Logistic Regression

Luigi Freda

ALCOR Lab
DIAG
University of Rome "La Sapienza"

January 28, 2018

Outline

1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

Outline

1 Intro
- Logistic Regression
- Decision Boundary

2 Maximum Likelihood Estimation
- Negative Log-Likelihood

3 Optimization Algorithms
- Gradient Descent
- Newton's Method
- Iteratively Reweighted Least Squares (IRLS)

4 Regularized Logistic Regression
- Concept

Linear Regression

linear regression

- $y \in \mathbb{R}$, $\mathbf{x} \in \mathbb{R}^D$ and $\mathbf{w} \in \mathbb{R}^D$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$

$$y(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon$$

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{w}^T \mathbf{x}, \sigma^2)$$

polynomial regression

- we replace $\mathbf{x}$ by a non-linear function $\phi(\mathbf{x}) \in \mathbb{R}^{d+1}$

$$y(x) = \mathbf{w}^T \phi(\mathbf{x}) + \epsilon$$

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{w}^T \phi(\mathbf{x}), \sigma^2)$$

- $\mu(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$ **(basis function expansion)**
- $\phi(\mathbf{x}) = [1, x, x^2, ..., x^d]$ is the vector of **polynomial basis functions**

N.B.: in both cases $\boldsymbol{\theta} = (\mathbf{w}, \sigma^2)$ are the **model parameters**

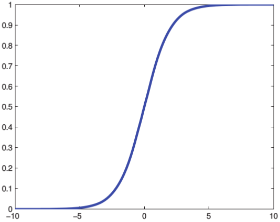
?can we generalize linear regression ($y \in \mathbb{R}$) to **binary classification** ($y \in \{0, 1\}$)?

we can follow two steps:

1. replace $y \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(x))$ with $y \sim \mathrm{Ber}(y|\mu(\mathbf{x}))$ (we want $y \in \{0, 1\}$)
2. replace $\mu(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with $\mu(\mathbf{x}) = \mathrm{sigm}(\mathbf{w}^T\mathbf{x})$ (we want $0 \leq \mu(\mathbf{x}) \leq 1$)

where

- $\mathrm{Ber}(y|\mu(\mathbf{x})) = \mu(\mathbf{x})^{\mathbb{I}(y=1)}(1 - \mu(\mathbf{x}))^{\mathbb{I}(y=0)}$ is the **Bernoulli distribution**
- $\mathbb{I}(e) = 1$ if e is true, $\mathbb{I}(e) = 0$ otherwise (**indicator function**)
- $\mathrm{sigm}(\eta) = \frac{\exp(\eta)}{1+\exp(\eta)} = \frac{1}{1+\exp(-\eta)}$ is **the sigmoid function** (aka logistic function)



following the two steps:

1. replace $y \sim \mathcal{N}(\mu(\mathbf{x}), \sigma^2(x))$ with $y \sim \mathrm{Ber}(y|\mu(\mathbf{x}))$ (we want $y \in \{0, 1\}$)
2. replace $\mu(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with $\mu(\mathbf{x}) = \mathrm{sigm}(\mathbf{w}^T\mathbf{x})$ (we want $0 \leq \mu(\mathbf{x}) \leq 1$)

we start from a linear regression

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2) \qquad \text{where } y \in \mathbb{R}$$

to obtain a logistic regression

$$p(y|\mathbf{x}, \mathbf{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^T\mathbf{x})) \qquad \text{where } y \in \{0, 1\}$$

Outline

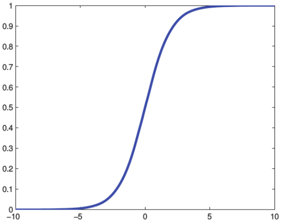
1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

$$p(y|\mathbf{x},\mathbf{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^T\mathbf{x})) \qquad \text{where } y \in \{0,1\}$$

- $p(y=1|\mathbf{x},\mathbf{w}) = \mathrm{sigm}(\mathbf{w}^T\mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x})}{1+\exp(\mathbf{w}^T\mathbf{x})} = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$
- $p(y=0|\mathbf{x},\mathbf{w}) = 1 - p(y=1|\mathbf{x},\mathbf{w}) = 1 - \mathrm{sigm}(\mathbf{w}^T\mathbf{x}) = \mathrm{sigm}(-\mathbf{w}^T\mathbf{x})$
- $p(y=1|\mathbf{x},\mathbf{w}) = p(y=0|\mathbf{x},\mathbf{w}) = 0.5$ entails

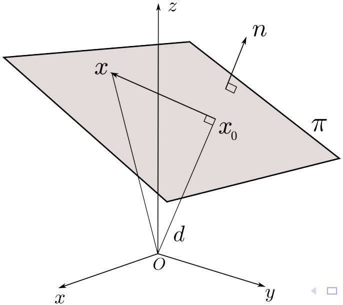
$$\mathrm{sigm}(\mathbf{w}^T\mathbf{x}) = 0.5 \implies \mathbf{w}^T\mathbf{x} = 0$$

- hence we have a **linear decision boundary** $\mathbf{w}^T\mathbf{x} = 0$



- **linear decision boundary** $\mathbf{w}^T\mathbf{x} = 0$ (hyperplane passing through the origin)
- indeed, as in the linear regression case $\mathbf{w}^T\mathbf{x} = [w_0, \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}]^T$ where $\mathbf{x} = [1, \tilde{\mathbf{x}}]^T$ and $\tilde{\mathbf{x}}_i$ are the *actual data samples*
- as a matter of fact, our linear decision boundary has the form $\mathbf{w}^T\tilde{\mathbf{x}} + w_0 = 0$

- **hyperplane** $\mathbf{a}^T\mathbf{x} + b = 0$ equivalent to $\mathbf{n}^T\mathbf{x} - d = 0$ where $\mathbf{n}$ is the normal unit vector (i.e. $\|\mathbf{n}\| = 1$) and $d \in \mathbb{R}$ is the distance origin-hyperplane
- one can define $\mathbf{x}_0 \triangleq \mathbf{n}d$ and rewrite the plane equation as $\mathbf{n}^T(\mathbf{x} - \mathbf{x}_0) = 0$



- we can replace $\mathbf{x}$ by a non-linear function $\phi(\mathbf{x})$ and obtain a

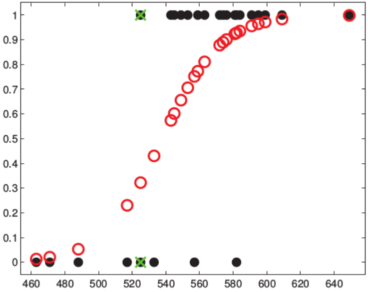
$$p(y|\mathbf{x}, \mathbf{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^T\phi(\mathbf{x})))$$

- if $x \in \mathbb{R}$ we can use $\phi(x) = [1, x, x^2, ..., x^d]$ which is the vector of **polynomial basis functions**
- in principle, if $\mathbf{x} \in \mathbb{R}^D$ we could use a **multivariate polynomial expansion** $\mathbf{w}^T\phi(\mathbf{x}) = \sum w_{i_1 i_2 \dots i_D} \prod_{j=1}^{D} x_j^{i_j}$ up to a certain degree d
- $p(y = 1|\mathbf{x}, \mathbf{w}) = \mathrm{sigm}(\mathbf{w}^T\phi(\mathbf{x}))$
- $p(y = 0|\mathbf{x}, \mathbf{w}) = \mathrm{sigm}(-\mathbf{w}^T\phi(\mathbf{x}))$
- $p(y = 1|\mathbf{x}, \mathbf{w}) = p(y = 0|\mathbf{x}, \mathbf{w}) = 0.5$ entails

$$\mathrm{sigm}(\mathbf{w}^T\phi(\mathbf{x})) = 0.5 \implies \mathbf{w}^T\phi(\mathbf{x}) = 0$$

- hence we have a **non-linear decision boundary** $\mathbf{w}^T\phi(\mathbf{x}) = 0$

Logistic Regression
A 1D Example

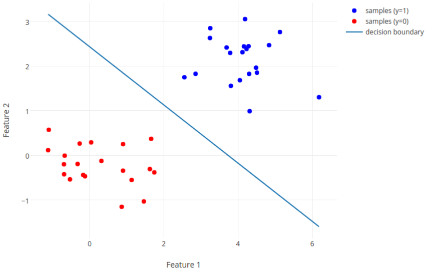


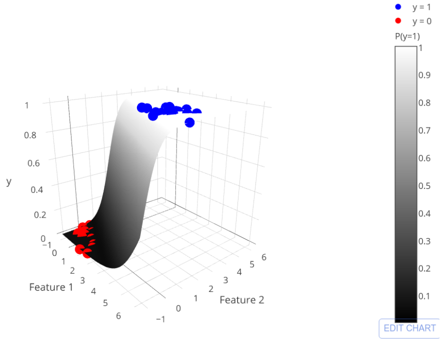
- solid black dots are data (x_i, y_i)
- open red circles are **predicted probabilities**: $p(y = 1|x, \mathbf{w}) = \text{sigm}(w_0 + w_1 x)$
- in this case data is **not** linearly separable
- the linear decision boundary is $w_0 + w_1 x = 0$ which entails $x = -w_0/w_1$

in general, when data is not linearly separable, we can try to use the basis function expansion as a further step

Logistic Regression

A 2D Example

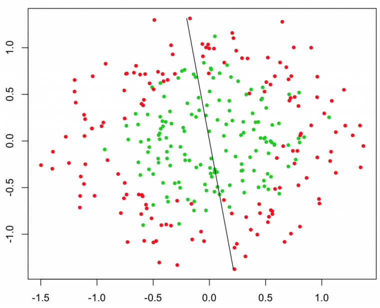


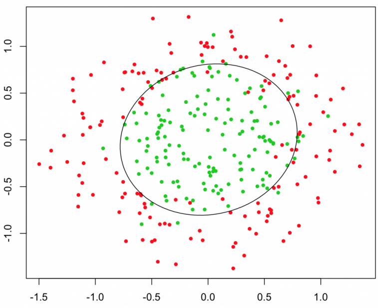


- *left*: a linear decision boundary on the "feature plane" (x_1, x_2)
- *right*: a 3D plot of $p(y = 1|\mathbf{x}, \mathbf{w}) = \text{sigm}(w_0 + w_1 x_2 + w_2 x_2)$

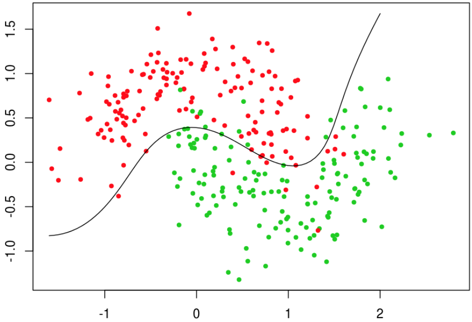
Logistic Regression

Examples





- *left*: non-linearly separable data with a linear decision boundary
- *right*: the same dataset fit with a quadratic model (and quadratic decision boundary)



another example of non-linearly separable data which is fit by using a polynomial model

Outline

1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

- the likelihood for the logistic regression is given by

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_i p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) = \prod_i \mathrm{Ber}(y_i|\mu_i) = \prod_i \mu_i^{\mathbb{I}(y_i=1)}(1-\mu_i)^{\mathbb{I}(y_i=0)}$$

  where $\mu_i \triangleq \mathrm{sigm}(\mathbf{w}^T\mathbf{x}_i)$

- the Negative Log-Likelihood (NLL) is given by

$$NLL = -\log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_i \Big[\mathbb{I}(y_i = 1)\log\mu_i + \mathbb{I}(y_i = 0)\log(1-\mu_i)\Big] =$$

$$= \sum_i \Big[y_i \log\mu_i + (1-y_i)\log(1-\mu_i)\Big]$$

Negative Log-Likelihood

Gradient and Hessian

- we have

$$NLL = \sum_i \left[y_i \log \mu_i + (1 - y_i) \log(1 - \mu_i) \right]$$

where $\mu_i \triangleq \text{sigm}(\mathbf{w}^T \mathbf{x}_i)$

- in order to find the MLE we have to minimize the NLL and impose $\frac{\partial \text{NLL}}{\partial w_i} = 0$
- given $\sigma(a) \triangleq \text{sigm}(a) = \frac{1}{1+e^{-a}}$ it is possible to show (**homework** ex 8.3) that

$$\frac{d\sigma(a)}{da} = \sigma(a)(1 - \sigma(a))$$

- using the previous equation and the chain rule for calculus we can compute the gradient $\mathbf{g}$

$$\mathbf{g} \triangleq \frac{d}{d\mathbf{w}} NLL(\mathbf{w}) = \sum_i \frac{\partial \text{NLL}}{\partial \mu_i} \frac{d\mu_i}{da_i} \frac{da_i}{d\mathbf{w}} = \sum_i (\mu_i - y_i)\mathbf{x}_i$$

where $\mu_i = \sigma(a_i)$ and $a_i \triangleq \mathbf{w}^T \mathbf{x}_i$

Negative Log-Likelihood
Gradient and Hessian

- the gradient can be rewritten as

$$\mathbf{g} = \sum_i (\mu_i - y_i)\mathbf{x}_i = \mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y})$$

where $\mathbf{X}$ is the design matrix, $\boldsymbol{\mu} \triangleq [\mu_1, ..., \mu_N]^T$, $\mathbf{y} \triangleq [y_1, ..., y_N]^T$ and $\mu_i \triangleq \text{sigm}(\mathbf{w}^T\mathbf{x}_i)$

- the Hessian is

$$\mathbf{H} \triangleq \frac{d}{d\mathbf{w}} g(\mathbf{w})^T = \sum_i \left(\frac{d\mu_i}{da_i}\frac{da_i}{d\mathbf{w}}\right)\mathbf{x}_i^T = \sum_i \mu_i(1-\mu_i)\mathbf{x}_i\mathbf{x}_i^T = \mathbf{X}^T\mathbf{S}\mathbf{X}$$

where $\mathbf{S} \triangleq \text{diag}(\mu_i(1-\mu_i))$

- it is easy to see that $\mathbf{H} > 0$ $\quad (\mathbf{v}^T\mathbf{H}\mathbf{v} = (\mathbf{v}^T\mathbf{X}^T)\mathbf{S}(\mathbf{X}\mathbf{v}) = \mathbf{z}^T\mathbf{S}\mathbf{z} > 0\ )$
- given that $\mathbf{H} > 0$ we have that the NLL is **convex** and has a **unique global minimum**
- unlike linear regression, there is no closed form for the MLE (since the gradient contains non-linear functions)
- we need to use an **optimization algorithm** to compute the MLE

Outline

1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

Gradient Descent

The Gradient

- given a continuously differentiable function $f(\boldsymbol{\theta}) \in \mathbb{R}$ we can use **first order** Taylor's expansion an approximate

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}^*) + \mathbf{g}(\boldsymbol{\theta}^*)^T(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$$

where the gradient $\mathbf{g}$ is defined as

$$\mathbf{g}(\boldsymbol{\theta}) \triangleq \frac{\partial f}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial f}{\partial \theta_1} \\ \vdots \\ \frac{\partial f}{\partial \theta_m} \end{bmatrix}$$

- hence, in a neighbourhood of $\boldsymbol{\theta}^*$ one has

$$\Delta f \approx \mathbf{g}^T \Delta\boldsymbol{\theta}$$

- it is easy to see that with $\|\Delta\boldsymbol{\theta}\| = \eta$ $\quad (\|\mathbf{v}\| \triangleq \sqrt{\mathbf{v}^T\mathbf{v}})$
  1. Δf is max when $\Delta\boldsymbol{\theta} = +\eta\frac{\mathbf{g}}{\|\mathbf{g}\|}$
  2. Δf is min when $\Delta\boldsymbol{\theta} = -\eta\frac{\mathbf{g}}{\|\mathbf{g}\|}$ (**steepest descent**)

where $\hat{\mathbf{g}} \triangleq \frac{\mathbf{g}}{\|\mathbf{g}\|}$ is the unit vector in the gradient direction

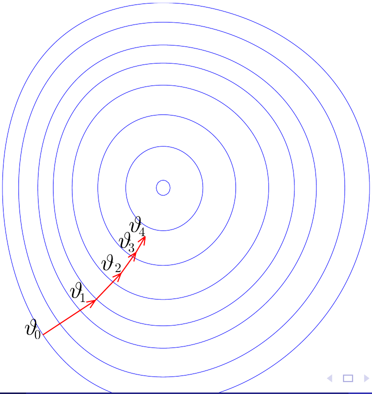
Gradient Descent

- the simplest algorithm for unconstrained optimization is **gradient descent** (aka steepest descent)

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \mathbf{g}_k$$

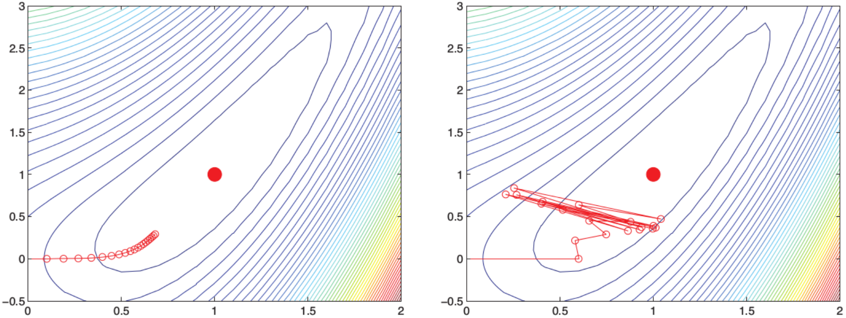
where $\eta \in \mathbb{R}^+$ is the **step size** (or **learning rate**) and $\mathbf{g}_k \triangleq \mathbf{g}(\boldsymbol{\theta}_k)$

- starting from an initial guess $\boldsymbol{\theta}_0$, at each step k we move towards the negative gradient direction $-\mathbf{g}_k$



Gradient Descent

problem: how to choose the step size η?



- *left*: using a fixed step size $\eta = 0.1$
- *right*: using a fixed step size $\eta = 0.6$
- if we use constant step size and we make it **too small**, convergence will be very slow, but if we make it **too large**, the method can fail to **convergence** at all

Gradient Descent

Line Search

- **convergence to the global optimum**: the method is guaranteed to converge to the **global optimum** $\boldsymbol{\theta}^*$ no matter where we start
- **global convergence**: the method is guaranteed to converge to a **local optimum** no matter where we start
- let's develop a more stable method for picking *eta* so as to have global convergence
- consider a general update

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta \mathbf{d}_k$$

  where $\eta > 0$ and $\mathbf{d}_k$ are respectively our **step size** and selected **descent direction**
- by Taylor's theorem, we have

$$f(\boldsymbol{\theta}_k + \eta \mathbf{d}_k) \approx f(\boldsymbol{\theta}_k) + \eta \mathbf{g}_k^T \mathbf{d}_k$$

- if η is chosen small enough and $\mathbf{d}_k = -\mathbf{g}_k$, then $f(\boldsymbol{\theta}_k + \eta \mathbf{d}_k) < f(\boldsymbol{\theta}_k)$ (since $\Delta f \approx -\eta \mathbf{g}^T \mathbf{g} < 0$)
- but we don't want to choose the step size η too small, or we will move very slowly and may not reach the minimum
- **line minimization** of **line search**: pick η so as to minimize

$$\phi(\eta) \triangleq f(\boldsymbol{\theta}_k + \eta \mathbf{d}_k)$$

- in order to minimize

$$\phi(\eta) \triangleq f(\boldsymbol{\theta}_k + \eta \mathbf{d}_k)$$

we must impose

$$\frac{d\phi}{d\eta} = \frac{\partial f}{\partial \boldsymbol{\theta}}^T \bigg|_{\boldsymbol{\theta}_k + \eta \mathbf{d}_k} \mathbf{d}_k = \mathbf{g}(\boldsymbol{\theta}_k + \eta \mathbf{d}_k)^T \mathbf{d}_k = 0$$

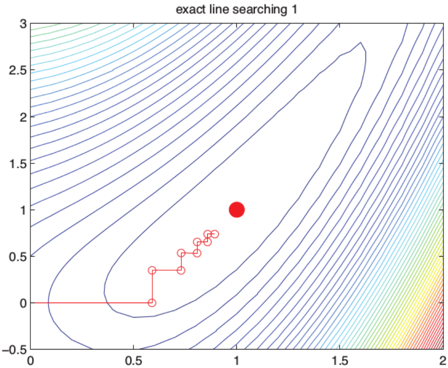
- since in the gradient descent method we have $\mathbf{d}_k = \mathbf{g}_k$, the following condition must be satisfied

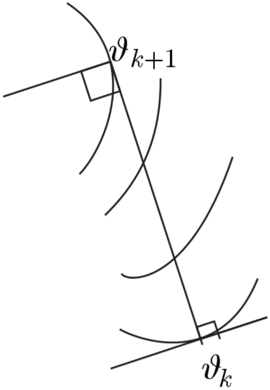
$$\mathbf{g}(\boldsymbol{\theta}_k + \eta \mathbf{d}_k)^T \mathbf{g}_k = 0$$

- from the following condition

$$\mathbf{g}(\boldsymbol{\theta}_k + \eta \mathbf{d}_k)^T \mathbf{g}_k = 0$$

we have that consecutive descent directions are **orthogonal** and we have a **zig-zag** behaviour





Outline

1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

- given a twice-continuously differentiable function $f(\boldsymbol{\theta}) \in \mathbb{R}$ we can use a **second order** Taylor's expansion to approximate

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}^*) + \mathbf{g}(\boldsymbol{\theta}^*)^T(\boldsymbol{\theta} - \boldsymbol{\theta}^*) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H}(\boldsymbol{\theta}^*)(\boldsymbol{\theta} - \boldsymbol{\theta}^*)$$

- the **Hessian matrix H** $= \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}^2}$ of a function $f(\boldsymbol{\theta}) \in \mathbb{R}$ is defined as follows (element-wise)

$$\mathbf{H}_{ij} = \frac{\partial^2 f(\boldsymbol{\theta})}{\partial \theta_i \partial \theta_j}$$

Newton's Method

- hence if we consider an optimization algorithm, at step k we have

$$f(\boldsymbol{\theta}) \approx f_{quad}(\boldsymbol{\theta}) \triangleq f(\boldsymbol{\theta}_k) + \mathbf{g}_k^T(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^T \mathbf{H}_k(\boldsymbol{\theta} - \boldsymbol{\theta}_k)$$

- in order to find $\boldsymbol{\theta}_{k+1}$ we can then minimize $f_{quad}(\boldsymbol{\theta})$

$$f_{quad}(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{A}\boldsymbol{\theta} + \mathbf{b}^T\boldsymbol{\theta} + c$$

  where

$$\mathbf{A} = \frac{1}{2}\mathbf{H}_k, \quad \mathbf{b} = \mathbf{g}_k - \mathbf{H}_k\boldsymbol{\theta}_k, \quad c = f_k - \mathbf{g}_k^T\boldsymbol{\theta}_k + \frac{1}{2}\boldsymbol{\theta}_k^T\mathbf{H}_k\boldsymbol{\theta}_k$$

- we can then impose

$$\frac{\partial f_{quad}}{\partial \boldsymbol{\theta}} = 0 \implies 2\mathbf{A}\boldsymbol{\theta} + \mathbf{b} = 0 \implies \mathbf{H}_k\boldsymbol{\theta} + \mathbf{g}_k - \mathbf{H}_k\boldsymbol{\theta}_k = 0$$

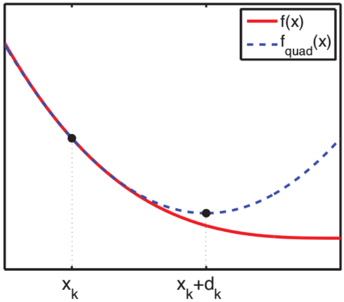
- the minimum of f_{quad} is then

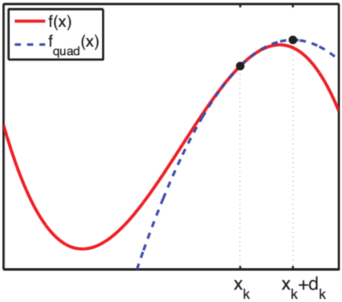
$$\boldsymbol{\theta} = \boldsymbol{\theta}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$$

- in the Newton's method one selects $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$

Newton's Method

- in the Newton's method one selects $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$
- the step $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ is what should be added to $\boldsymbol{\theta}_k$ to minimize the second order approximation of f around $\boldsymbol{\theta}_k$
- in its simplest form, Newton's method requires that $\mathbf{H}_k > 0$ (the function is strictly convex)
- if not, the objective function is not convex, then $\mathbf{H}_k$ may not be positive definite, so $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ may not be a descent direction





Newton's Method

Algorithm 8.1: Newton's method for minimizing a strictly convex function

Initialize $\boldsymbol{\theta}_0$;
for $k = 1, 2, \ldots$ *until convergence* **do**
    Evaluate $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k)$;
    Evaluate $\mathbf{H}_k = \nabla^2 f(\boldsymbol{\theta}_k)$;
    Solve $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$;
    Use line search to find stepsize η_k along $\mathbf{d}_k$;
    $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{d}_k$;

Outline

1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

- let us now apply Newton's algorithm to find the MLE for binary logistic regression
- the Newton update at iteration $k+1$ for this model is as follows (using $\eta_k = 1$, since the Hessian is exact)

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$$

- since

$$\mathbf{g}_k = \mathbf{X}^T(\boldsymbol{\mu}_k - \mathbf{y}), \quad \mathbf{H}_k = \mathbf{X}^T\mathbf{S}_k\mathbf{X}$$

we have

$$\mathbf{w}_{k+1} = \mathbf{w}_k + (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_k) =$$

$$= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}[(\mathbf{X}^T\mathbf{S}_k\mathbf{X})\mathbf{w}_k + \mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_k)] = (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{S}_k\mathbf{X}\mathbf{w}_k + \mathbf{y} - \boldsymbol{\mu}_k)$$

- then we have

$$\mathbf{w}_{k+1} = (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T\mathbf{S}_k\mathbf{z}_k$$

where $\mathbf{z}_k \triangleq \mathbf{X}\mathbf{w}_k + \mathbf{S}_k^{-1}(\mathbf{y} - \boldsymbol{\mu}_k)$

- the following equation

$$\mathbf{w}_{k+1} = (\mathbf{X}^T \mathbf{S}_k \mathbf{X})^{-1} \mathbf{X}^T \mathbf{S}_k \mathbf{z}_k$$

  with $\mathbf{z}_k \triangleq \mathbf{X}\mathbf{w}_k + \mathbf{S}_k^{-1}(\mathbf{y} - \boldsymbol{\mu}_k)$ is an example of **weighted least squares problem**, which is a minimizer of

$$J = \sum_{i=1}^{N} s_{ki}(z_{ki} - \mathbf{w}^T \mathbf{x}_i)^2 = \|\mathbf{z}_k - \mathbf{X}\mathbf{w}_k\|_{\mathbf{S}_k^{-1}}$$

  where $\mathbf{S}_k = \text{diag}(s_{ki})$, $\mathbf{z}_k = [z_{k1}, ..., z_{kN}]^T$

- since $\mathbf{S}_k$ is a diagonal matrix we can write the element-wise update

$$z_{ki} = \mathbf{w}_k^T \mathbf{x}_i + \frac{y_i - \mu_{ki}}{\mu_{ki}(1 - \mu_{ki})}$$

  where $\boldsymbol{\mu}_k = [\mu_{k1}, ..., \mu_{kN}]^T$

- this algorithm is called **iteratively reweighted least squares** (IRLS)

Iteratively Reweighted Least Squares

IRLS

Algorithm 8.2: Iteratively reweighted least squares (IRLS)

$\mathbf{w} = \mathbf{0}_D$;

$w_0 = \log(\overline{y}/(1-\overline{y}))$;

repeat

$\quad \eta_i = w_0 + \mathbf{w}^T\mathbf{x}_i$;

$\quad \mu_i = \text{sigm}(\eta_i)$;

$\quad s_i = \mu_i(1-\mu_i)$;

$\quad z_i = \eta_i + \frac{y_i - \mu_i}{s_i}$;

$\quad \mathbf{S} = \text{diag}(s_{1:N})$;

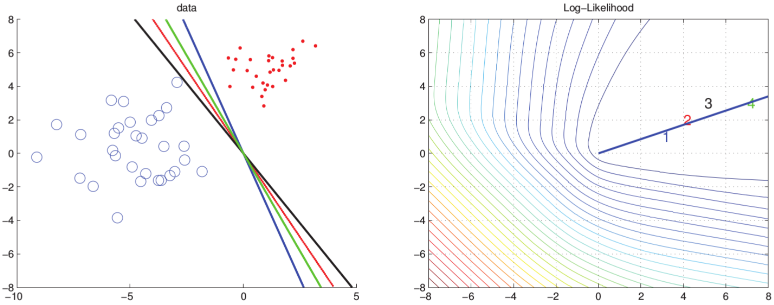
$\quad \mathbf{w} = (\mathbf{X}^T\mathbf{S}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{S}\mathbf{z}$;

until *converged*;

Outline

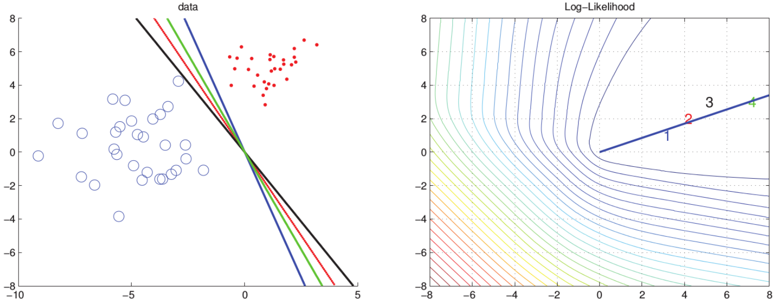
1. Intro
   - Logistic Regression
   - Decision Boundary
2. Maximum Likelihood Estimation
   - Negative Log-Likelihood
3. Optimization Algorithms
   - Gradient Descent
   - Newton's Method
   - Iteratively Reweighted Least Squares (IRLS)
4. Regularized Logistic Regression
   - Concept

Regularized Logistic Regression



- consider the **linearly separable** 2D data in the above figure
- there are different decision boundaries that can perfectly separate the training data (4 examples are shown in different colors)
- the likelihood surface is shown: it is unbounded as we move up and to the right in parameter space, along a ridge where $w_2/w_1 = 2.35$ (the indicated diagonal line)

Regularized Logistic Regression



- we can maximize the likelihood by driving $\|\mathbf{w}\|$ to infinity (subject to being on this line), since large regression weights make the sigmoid function very steep, turning it into an infinitely steep sigmoid function $\mathbb{I}(\mathbf{w}^T\mathbf{x} > w_0)$
- consequently the **MLE is not well defined** when the data is linearly separable

Regularized Logistic Regression

- to prevent this, we can move to **MAP estimation** and hence add a **regularization component** in the classification setting (as we did in the ridge regression)
- to regularize the problem we can simply add spherical prior at the origin $p(\mathbf{w}) = \mathcal{N}(\mathbf{x}|0, \lambda\mathbf{I})$ and then maximize the posterior $p(\mathbf{w}|\mathcal{D}) \propto p(\mathcal{D}|\mathbf{w})p(\mathbf{w})$
- as a consequence a simple l_2 regularization can be easily obtained by using the following new objective, gradient and Hessian

$$f'(\mathbf{w}) = \text{NLL}(\mathbf{w}) + \lambda\mathbf{w}^T\mathbf{w}$$
$$\mathbf{g}'(\mathbf{w}) = \mathbf{g}(\mathbf{w}) + 2\lambda\mathbf{w}$$
$$\mathbf{H}'(\mathbf{w}) = \mathbf{H}(\mathbf{w}) + 2\lambda\mathbf{I}$$

- these modified equations can be used into any of the presented optimizers

Credits

- Kevin Murphy's book