# Logistic Regression

3 Jun 2019

Vineeth N Balasubramanian

आई आई टी हैदराबाद
IIT Hyderabad

# Classification Methods

- k-Nearest Neighbors
- Decision Trees
- Naïve Bayes
- Support Vector Machines
- Logistic Regression
- Neural Networks
- Ensemble Methods (Boosting, Random Forests)

# Regression Formulation

Given $x$, want to predict an estimate $\hat{y}$ of $y$, which minizes the discrepancy ($L$) between $\hat{y}$ and $y$.

$$L(\hat{y}; y) := |\hat{y} - y| \qquad \textit{Absolute error}$$

$$:= (\hat{y} - y)^2 \qquad \textit{Squared error}$$

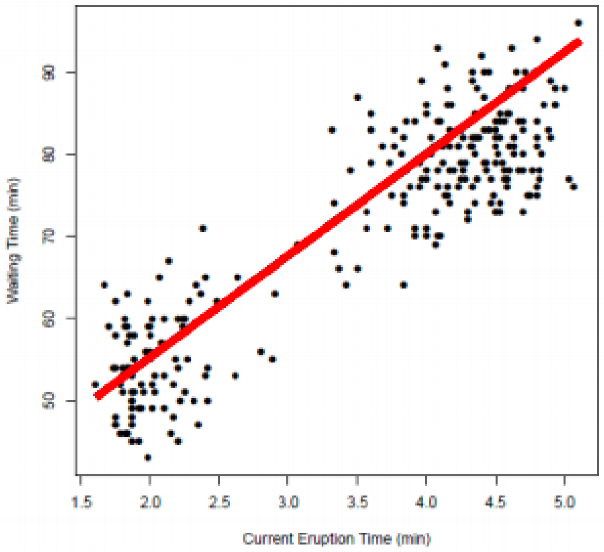
Loss

A **linear predictor** $f$, can be defined by the slope $w$ and the intercept $w_0$ :

$$\hat{f}(\vec{x}) := \vec{w} \cdot \vec{x} + w_0$$

which minimizes the prediction loss.

$$\min_{w, w_0} \mathbb{E}_{\vec{x}, y} \left[ L(\hat{f}(\vec{x}), y) \right]$$



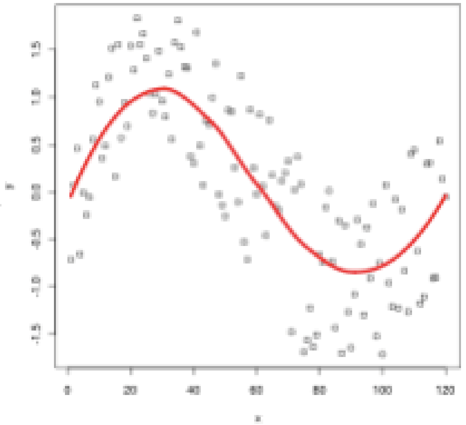

Slide Credit: Nakul Verma, Columbia Univ

# Parametric (vs) Non-parametric Regression

If we assume a particular form of the regressor:

*Parametric regression*

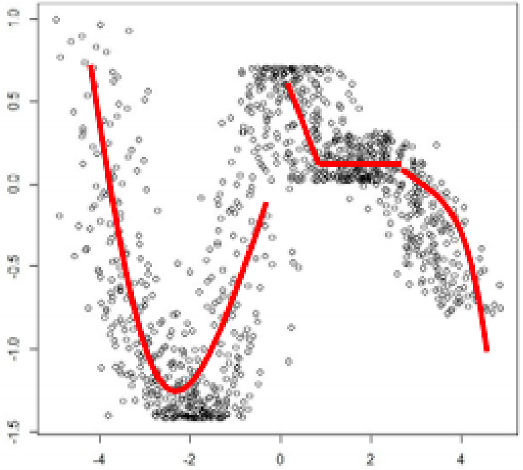
*Goal: to learn the parameters which yield the minimum error/loss*



If no specific form of regressor is assumed:

*Non-parametric regression*

*Goal: to learn the predictor directly from the input data that yields the minimum error/loss*



Slide Credit: Nakul Verma, Columbia Univ

## Linear Regression

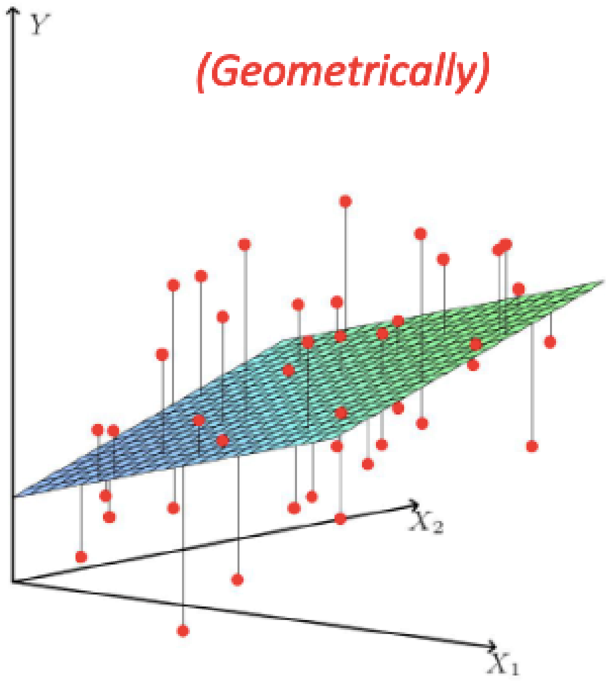
Want to find a **linear predictor** *f*, i.e., *w* (intercept $w_0$ absorbed via lifting):

$$\hat{f}(\vec{x}) := \vec{w} \cdot \vec{x}$$

which minimizes the prediction loss over the population.

$$\min_{\vec{w}} \mathbb{E}_{\vec{x},y} \left[ L(\hat{f}(\vec{x}), y) \right]$$

*(Geometrically)*




We estimate the parameters by minimizing the corresponding loss on the training data:

$$\arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} \left[ L(\vec{w} \cdot \vec{x}_i, y_i) \right]$$

$$= \arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} \left( \vec{w} \cdot \vec{x}_i - y_i \right)^2$$

*for squared error*

Slide Credit: Nakul Verma, Columbia Univ

Linear Regression

## Linear predictor with squared loss:

$$\arg\min_{w} \frac{1}{n} \sum_{i=1}^{n} \left(\vec{w} \cdot \vec{x}_i - y_i\right)^2$$

$$= \arg\min_{w} \left\| \begin{pmatrix} \ldots x_1 \ldots \\ \ldots x_i \ldots \\ \ldots x_n \ldots \end{pmatrix} \begin{pmatrix} w \end{pmatrix} - \begin{pmatrix} y_1 \\ y_i \\ y_n \end{pmatrix} \right\|^2$$

$$= \arg\min_{w} \left\| X\vec{w} - \vec{y} \right\|_2^2$$

*Unconstrained problem!*

*Can take the gradient and examine the stationary points!*

*Why need not check the second order conditions?*

Slide Credit: Nakul Verma, Columbia Univ

Linear Regression

## Best fitting *w*:

$$\frac{\partial}{\partial \vec{w}} \|X\vec{w} - \vec{y}\|^2 = 2X^\top(X\vec{w} - \vec{y})$$

$$X^\top X\vec{w} = X^\top \vec{y}$$

*At a stationary point*

$$\implies \vec{w}_{\text{ols}} = (X^\top X)^\dagger X^\top \vec{y}$$

Pseudo-inverse

***Also called the Ordinary Least Squares (OLS)***

*The solution is unique and stable when $X^T X$ is invertible*

Slide Credit: Nakul Verma, Columbia Univ

# Regularized Least-Squared Regression

- Complex models (lots of parameters) often prone to overfitting.
- Overfitting can be reduced by imposing a constraint on the overall magnitude of the parameters.
- Two common types of regularization in linear regression:
  - **$L_2$ regularization (a.k.a. ridge regression):** Find **w** which minimizes:

$$\sum_{j=1}^{N}\left(y_j - \sum_{i=0}^{d} w_i \cdot x_i\right)^2 + \lambda \sum_{i=1}^{d} w_i^{\,2}$$

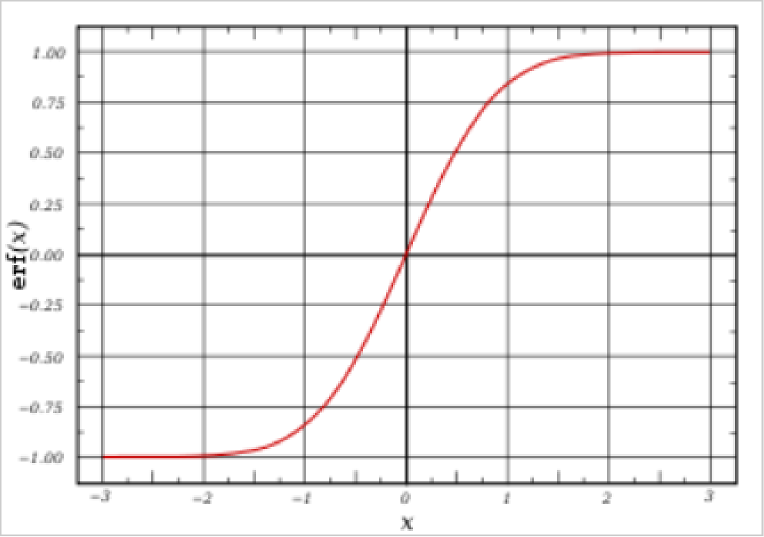
    - $\lambda$ is the regularization parameter: bigger $\lambda$ imposes more constraint
  - **$L_1$ regularization (a.k.a. lasso):** Find **w** which minimizes:

$$\sum_{j=1}^{N}\left(y_j - \sum_{i=0}^{d} w_i \cdot x_i\right)^2 + \lambda \sum_{i=1}^{d} |w_i|$$

# Logistic Regression

- To predict an outcome variable that is categorical from one or more categorical or continuous predictor variables.
- Used because having a categorical outcome variable violates the assumption of linearity in normal regression.

- Let X be the data instance, and Y be the class label: Learn P(Y|X) directly
  - Let W = ($W_1$, $W_2$, … $W_n$), X=($X_1$, $X_2$, …, $X_n$), **W.X** is the dot product
  - Sigmoid function: $P(Y=1\mid \mathbf{X}) = \dfrac{1}{1+e^{-\mathbf{wx}}}$



# Logistic Regression

- Generative or Discriminative?

# Logistic Regression

- Generative classifier, e.g., Naïve Bayes:
  - Assume some functional form for **P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X=x)
  - This is 'generative' model
    - Indirect computation of P(Y|X) through Bayes rule
    - But, can generate a sample of the data

- Discriminative classifier, e.g., Logistic Regression:
  - Assume some functional form for **P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the 'discriminative' model
    - Directly learn P(Y|X)

# Logistic Regression

- In logistic regression, we learn the conditional distribution P(y|x)
- Let $p_y(x;w)$ be our estimate of P(y|x), where w is a vector of adjustable parameters.
- Assume there are two classes, y = 0 and y = 1 and

$$p_1(\mathbf{x};\mathbf{w}) = \frac{1}{1+e^{-\mathbf{wx}}} \qquad p_0(\mathbf{x};\mathbf{w}) = 1 - \frac{1}{1+e^{-\mathbf{wx}}}$$

- This is equivalent to $\log \frac{p_1(\mathbf{x};\mathbf{w})}{p_0(\mathbf{x};\mathbf{w})} = \mathbf{wx}$

- That is, the log odds of class 1 is a linear function of x
- Q: How to find **W**?

# Logistic Regression

- Conditional data likelihood - Probability of observed Y values in the training data, conditioned on corresponding X values.
- We choose parameters w that satisfy

$$\mathbf{w} = \arg\max_{\mathbf{w}} \prod_{l} P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

- where
  - $\mathbf{w} = \langle w_0, w_1, \ldots, w_n \rangle$ is the vector of parameters to be estimated,
  - $y^l$ denotes the observed value of Y in the *l* th training example, and
  - $\mathbf{x}^l$ denotes the observed value of **X** in the *l* th training example

# Logistic Regression

- Equivalently, we can work with log of conditional likelihood:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_l \ln P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

- Conditional data log likelihood, l(W), can be written as

$$l(\mathbf{w}) = \sum_l y^l \ln P(y^l = 1 \mid \mathbf{x}^l, \mathbf{w}) + (1 - y^l) \ln P(y^l = 0 \mid \mathbf{x}^l, \mathbf{w})$$

- Note here that Y can take only values 0 or 1, so only one of the two terms in the expression will be non-zero for any given $y^l$

# Logistic Regression: Training

- We need to estimate:

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^l \mid \mathbf{x}^l, \mathbf{w})$$

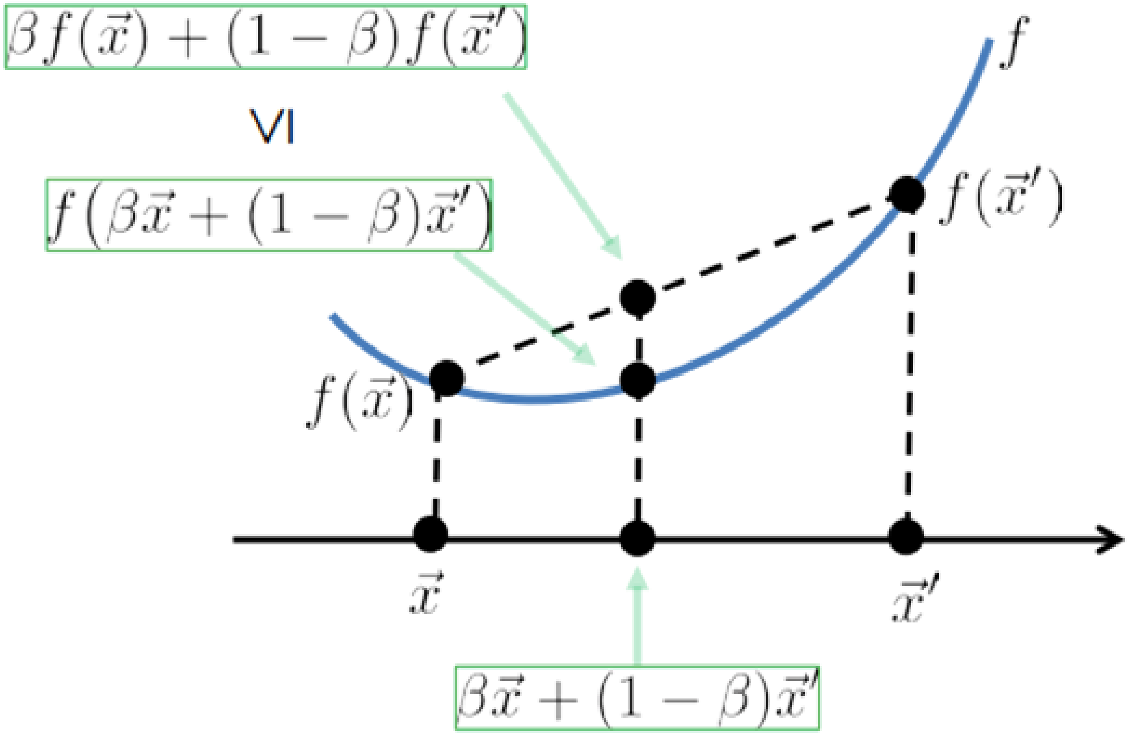
- Equivalently, we can minimize negative log likelihood

- This is convex – so, unique global minimum
- No closed-form solution though. Iterative method required.

## Convexity

A function $f$: $\mathbf{R}^d \rightarrow \mathbf{R}$ is called convex iff for any two points $x$, $x'$ and $\beta \in [0,1]$

$$f\left(\beta\vec{x} + (1-\beta)\vec{x}'\right) \leq \beta f(\vec{x}) + (1-\beta) f(\vec{x}')$$



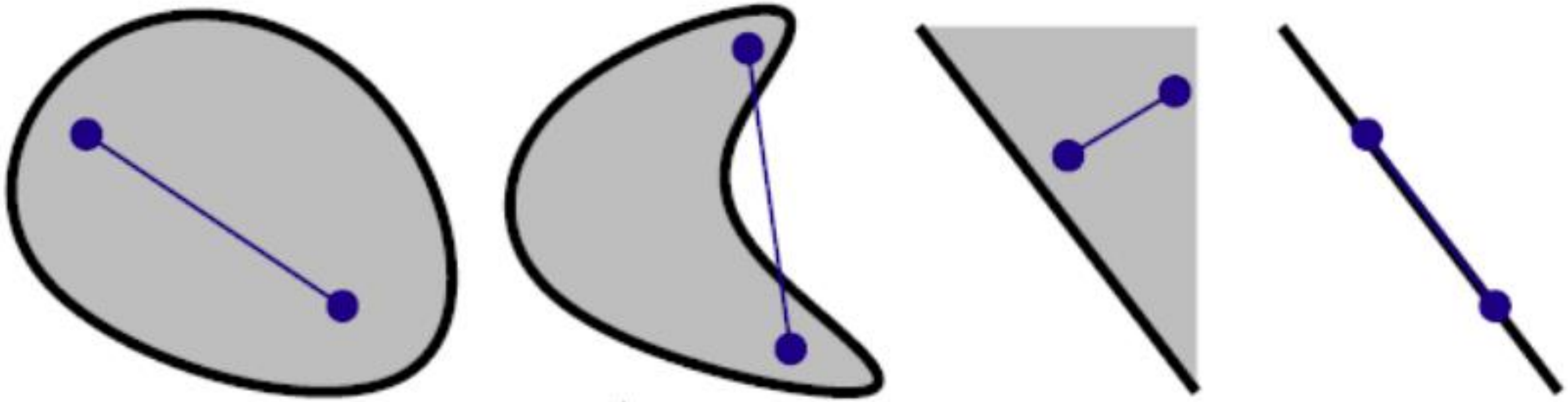

Slide credit: Nakul Verma, Columbia University

# Convexity

A set $S \subset \mathbf{R}^d$ is called convex iff for any two points $x, x' \in S$ and any $\beta \in [0,1]$

$$\beta \vec{x} + (1-\beta)\vec{x}' \in S$$

Examples:




Slide credit: Nakul Verma, Columbia University

## Convex Optimization

A constrained optimization

$$\underset{\vec{x} \in \mathbf{R}^d}{\text{minimize}} \quad f(\vec{x}) \qquad \textit{(objective)}$$

$$\text{subject to:} \quad g_i(\vec{x}) \leq 0 \quad \text{for } 1 \leq i \leq n \qquad \textit{(constraints)}$$

is called a convex optimization problem

If:

the objective function $f(\vec{x})$ is convex function, and

the feasible set induced by the constraints $g_i$ is a convex set

*Why do we care?*

*We* ***can*** *find the optimal solution for convex problems* ***efficiently****!*

Slide credit: Nakul Verma, Columbia University

# Classification Methods

- Every local optima is a global optima in a convex optimization problem.
- Example convex problems:
  - Linear programs, quadratic programs,
  - Conic programs, semi-definite program.
- Several solvers exist to find the optima:
  - CVX, SeDuMi, C-SALSA, …
- We can use a simple 'descent-type' algorithm for finding the minima!

Slide credit: Nakul Verma, Columbia University

# Gradient Descent

**Theorem (Gradient Descent):**

Given a smooth function $f : \mathbf{R}^d \rightarrow \mathbf{R}$

Then, for any $\vec{x} \in \mathbf{R}^d$ and $\vec{x}' := \vec{x} - \eta \nabla_x f(\vec{x})$

For sufficiently small $\eta > 0$, we have: $f(\vec{x}') \leq f(\vec{x})$

Can derive a simple algorithm (the projected Gradient Descent):

Initialize $\vec{x}^0$

for t = 1,2,...do

$\vec{x}'^t := \vec{x}^{t-1} - \eta \nabla_x f(\vec{x}^{t-1})$ *(step in the gradient direction)*

$\vec{x}^t := \Pi_{g_i}(\vec{x}'^t)$ *(project back onto the constraints)*

terminate when no progress can be made, ie, $|f(\vec{x}^t) - f(\vec{x}^{t-1})| \leq \epsilon$

Slide credit: Nakul Verma, Columbia University

# Logistic Regression: Training

- Use gradient ascent (descent) for the maximization (min) problem
- The i th component of the vector gradient has the form

$$\frac{\partial}{\partial w_i} l(\mathbf{w}) = \sum_l x_i^l (y^l - \boxed{\hat{P}(y^l = 1 \mid \mathbf{x}^l, \mathbf{w})})$$

Logistic Regression prediction

- Beginning with initial weights, we repeatedly update the weights in the direction of the gradient, changing the i th weight according to

$$w_i \leftarrow w_i + \eta \sum_l x_i^l (y^l - \hat{P}(y^l = 1 \mid \mathbf{x}^l, \mathbf{w}))$$

# Regularization in Logistic Regression

- Overfitting can arise especially when data has very high dimensions and is sparse.
- One approach -> modified "penalized log likelihood function," which penalizes large values of **w**, as before.

$$\mathbf{w} = \arg\max_{\mathbf{w}} \sum_{l} \ln P(y^l \mid \mathbf{x}^l, \mathbf{w}) - \frac{\lambda}{2} \|\mathbf{w}\|^2$$

- Derivative then becomes:

$$\frac{\partial}{\partial w_i} l(\mathbf{w}) = \sum_{l} x_i^l (y^l - \hat{P}(y^l = 1 \mid \mathbf{x}^l, \mathbf{w})) - \lambda w_i$$

# Logistic Regression

- In general, NB and LR make different assumptions
  - NB: Features independent given class -> assumption on P(X|Y)
  - LR: Functional form of P(Y|X), no assumption on P(X|Y)
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - Concave (convex) -> global optimum with gradient ascent (descent)
- Extending logistic regression to multiple classes
  - Use softmax for each class k! $p(y=k|x) = \frac{\exp(\theta_k^\top x)}{\sum_{i=1}^{K} \exp(\theta_i^\top x)}$

# Readings

- PRML Bishop, Chapter 4 (Sec 4.3)
- "Introduction to Machine Learning" by Ethem Alpaydin, Chapter 10