NYC DATA SCIENCE ACADEMY

Python Machine Learning

Logistic Regression And Gradient Descent

NYC Data Science Academy

Outline

- Limitation of Linear Regression
  - Logistic Regression
  - Gradient Descent Algorithm

Classification Problems

- *Categorical* (qualitative) variables: takes values in a finite set (usually unordered, i.e. nominal).
  - email: {spam, non-spam}
  - blood type: {A, B, AB, O}
  - tumor: {malignant, benign}
- *Classification*: given a feature (or a set of features), we want to predict categorical output.
- Sometimes people are also interested in estimating the probabilities that *an observation* belongs to each category.
- For example a business would like to estimate the probability for a customer to purchase each of its products.

A Classification Example

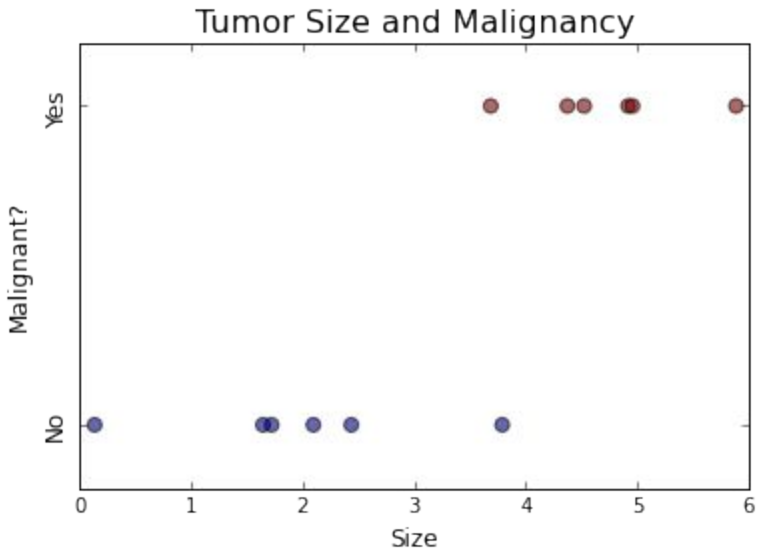
- Predict whether a tumor is malignant or benign based on the tumor size
- The output is binary:
  - 0: benign
  - 1: malignant
- Here is a simulated data set:

	Size	Malignant
0	3.788628	0
1	2.436510	0
2	2.096497	0
3	0.136507	0
4	1.722612	0
5	1.645241	0

	Size	Malignant
6	4.917259	1
7	4.372999	1
8	4.956182	1
9	4.522782	1
10	3.686135	1
11	5.884622	1

A Classification Example

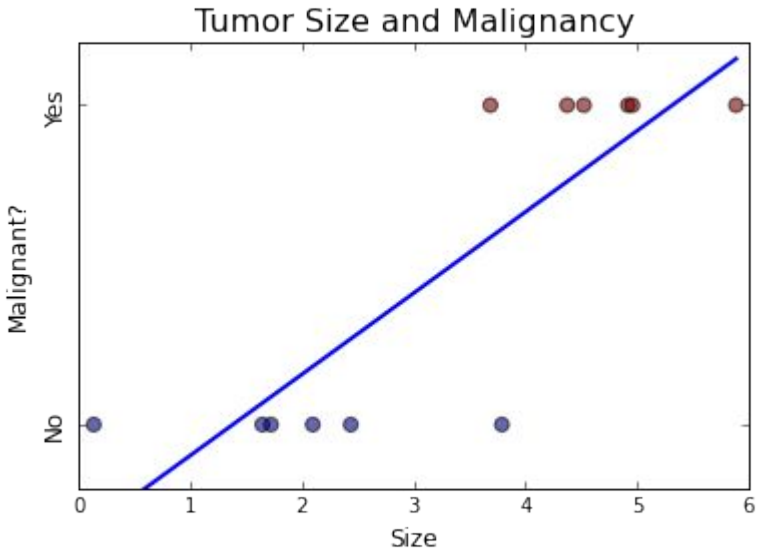
- By selecting the Size as feature and Malignant as output, we can visualize the data as:



- Question: Is linear regression suitable for the classification task?

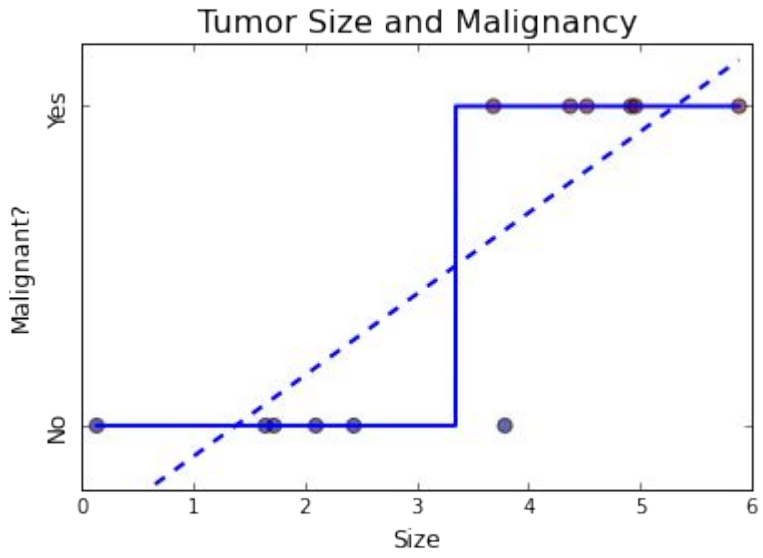
Can We Use Linear Regression as a Classifier?

- Let us fit a linear regression model with the **simulated** tumor data:



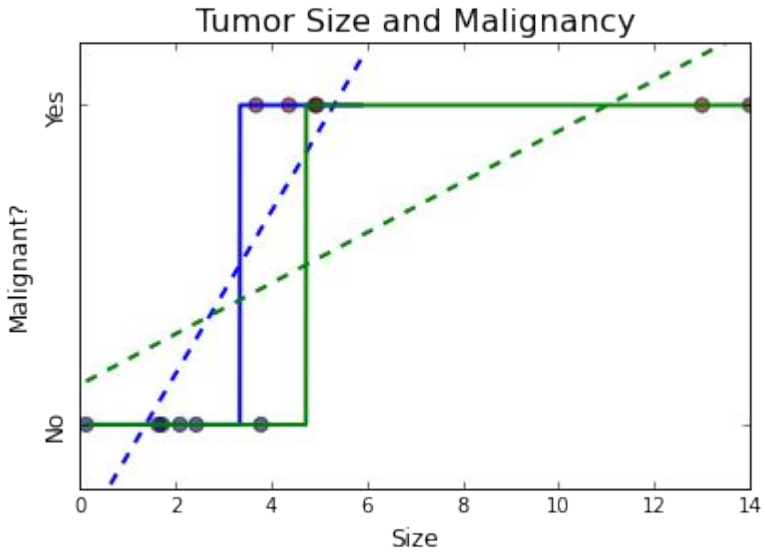
Is Linear Regression Suitable for Classification?

- We may then set a threshold
  - Predict 1 if $\hat{y} \geq 0.5$
  - Predict 0 if $\hat{y} < 0.5$
- The predicted values become binary:



Issues with Linear Regression

- It looks like the binary prediction with linear regression is not a bad idea. However, we do have the following two issues:
  - the continuous output often exceeds the unit interval [0, 1]. Therefore we cannot interpret it as a probability.
  - the prediction can be affected by outliers easily.

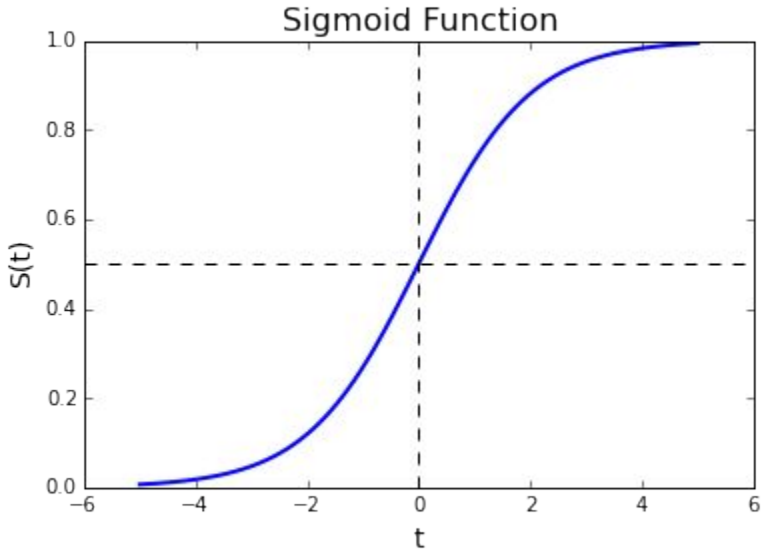


Sigmoid Function

- Sigmoid Function: a monotonically increasing smooth function which maps a real value to a positive value bounded between 0 and 1.

$$S(t) = \frac{e^t}{1 + e^t}$$

- ***e*** ≅ 2.718 is a mathematical constant (Euler's number).



Logistic Regression

Logistic regression, despite its name, is a linear model for classification rather than regression.

- Idea: if we transform the linear function $\beta_0+\beta_1 X$ using the sigmoid function $S(t)$, then no matter what values β_0, β_1 or X take, y will always have values between 0 and 1.
- *Logistic Regression models* use this form to estimate the probability that $y = 1$ given its size X:

$$Pr(Y = 1|X = x) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

- So different β_0 and β_1 will give different estimations Pr.
- The estimated probability is not explicitly determined by the data, but is estimated indirectly by **maximum likelihood**

Odds of Probabilities

- Given a point inside the feature space, we can estimate the sample probability (also called empirical probability) using the ratio of class 1 samples within a small neighborhood of the given point
- This constrains the logistic classifier coefficients through the above regression equation $Pr(Y = 1|X = x) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$
- To understand the meaning of the logistic regression equation, we introduce the concept of 'odds'
- The **odds** refers to class 1 to class 0 probability ratio p:(1-p) = p/(1-p), which dictates the imbalance of the class 1 vs class 0 frequencies

Probability & Odds

- Suppose I have a fair 6-sided die. Consider the following events:
  - Success: Rolling a 2 or 5.
  - Failure: Rolling a 1, 3, 4, or 6.

- What is the probability p of success?

$$p = \frac{\#\ Successes}{\#\ Events} = \frac{2}{6} = \frac{1}{3}$$

- What are the odds of success?

$$Odds = \frac{p}{1-p} = \frac{\frac{1}{3}}{1-\frac{1}{3}} = \frac{1}{2}$$

Linear Regression on Log Odds

- To see how does odds occur naturally, we derive

$$p = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}$$

$$1-p = 1 - \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} = \frac{1}{1+e^{\beta_0+\beta_1 X}}$$

$$\frac{p}{1-p} = \frac{\frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}}{\frac{1}{1+e^{\beta_0+\beta_1 X}}} = e^{\beta_0+\beta_1 X}$$

$$log(\frac{p}{1-p}) = log(e^{\beta_0+\beta_1 X}) = \beta_0+\beta_1 X$$

- Thus logistic regression can be viewed as a linear regression on the log odds
- While p varying in (0,1) interval, th odds ranges in $(0, \infty)$
- Log odds varies in the real line $(-\infty, \infty)$, which is naturally the range of the target of a linear regression

Outline

- Limitation of Linear Regression
- **Logistic Regression**
- Gradient descent

Log Odds as the Link Function

- **Log Odds** is known as the **link function** (using the terminology of **generalized linear model**), it is also called the **logit function**
- Log Odds and the sigmoid function are inverse to each other
- Sigmoid function, as the inverse function to log odds, is called **logistic function**
- Using the log odds as the link function, we are able to connect our linear model to our response in such a way that creates a sigmoidal curve to better fit our data.

- Notice that while logistic regression is a kind of nonlinear regression (because the equation for the probability p is not linear in respect to the coefficients), the nonlinearity is contained solely in the **link function**

Maximal Likelihood

- Let's write the likelihood function

$$p(x_i, \beta_0, \beta_1) = Pr(Y = 1|X = x_i)$$

to describe the **estimated** probability of observed outcomes to be of class 1 given $X = x_i$.

- Because there are only two classes, labeled 0 and 1, the estimated probability of observed outcomes to be of class 0 is given by, given $X = x_i$,

$$1 - p(x_i, \beta_0, \beta_1)$$

- This is because the probabilities of distinct possibilities always sum up to be 1

Maximum Likelihood

- Then, given an input *X* with *n* **independent** observations, the likelihood gives the **joint** probability of having the observations with the prescribed labels:

$$L(\beta_0, \beta_1) = \prod_{i, y_i=1} p(x_i, \beta_0, \beta_1) \prod_{i, y_i=0} (1-p(x_i, \beta_0, \beta_1))$$

where the first product gives the probability of successfully predicting the "1"s and the second product is the probability of successfully predicting the "0"s in the given data.

Maximum Likelihood

- The likelihood function $L(\beta_0, \beta_1)$ gives the probability of making the same prediction as the observed data.
- Among all the linear models, the pair with the higher L has a higher probability to produce the prescribed class labels.
- We want to pick β_0 and β_1 to maximize the likelihood $L(\beta_0, \beta_1)$, i.e., to maximize the "agreement" of the selected model with the observed data.
- The maximal likelihood function discussed here is closely tied to **Bernoulli** distribution (coin flipping) and **binomial** distribution (independent coin flippings) discussed later.

The Formulation of Maximum Likelihood Method

- ML estimation was formulated and populated in the 20th century by the famous bio-statistician Sir Ronald Fisher, following the footstep of Gauss, Laplace, Edgeworth
- This is one of the major achievements of 20th century statistics and the cornerstone of probabilistic machine learning
- Fisher invented Fisher discriminant analysis, introducing IRIS data set, inventing ANOVA ……

Log-Likelihood

- Recall that $log(e^g) = g$, where we use 'log' to denote the natural log function, ln.
- In practice it is often more convenient to work with the logarithm of the likelihood function, called the **log-likelihood**:

$$\log L(\beta_0, \beta_1) = \sum_{i=1}^{n} \{y_i \log p(x_i, \beta_0, \beta_1) + (1 - y_i) \log(1 - p(x_i, \beta_0, \beta_1))\}$$

$$= \sum_{i=1}^{n} \{y_i(\beta_0 + \beta_1 X) - \log(1 + e^{\beta_0 + \beta_1 X})\}$$

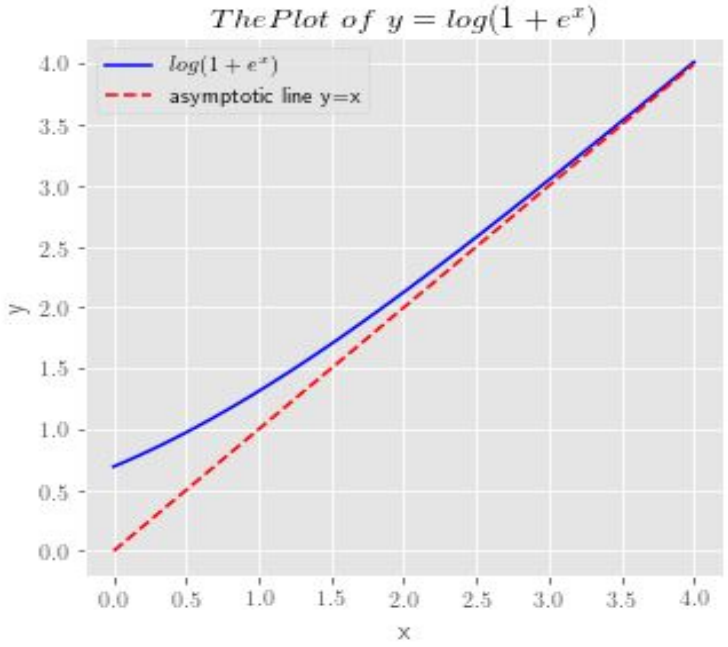
- Logistic regression models are usually fitted by maximum log-likelihood, i.e., to find β_0 and β_1 that maximize the log likelihood function above.

Concavity of the Log-Likelihood

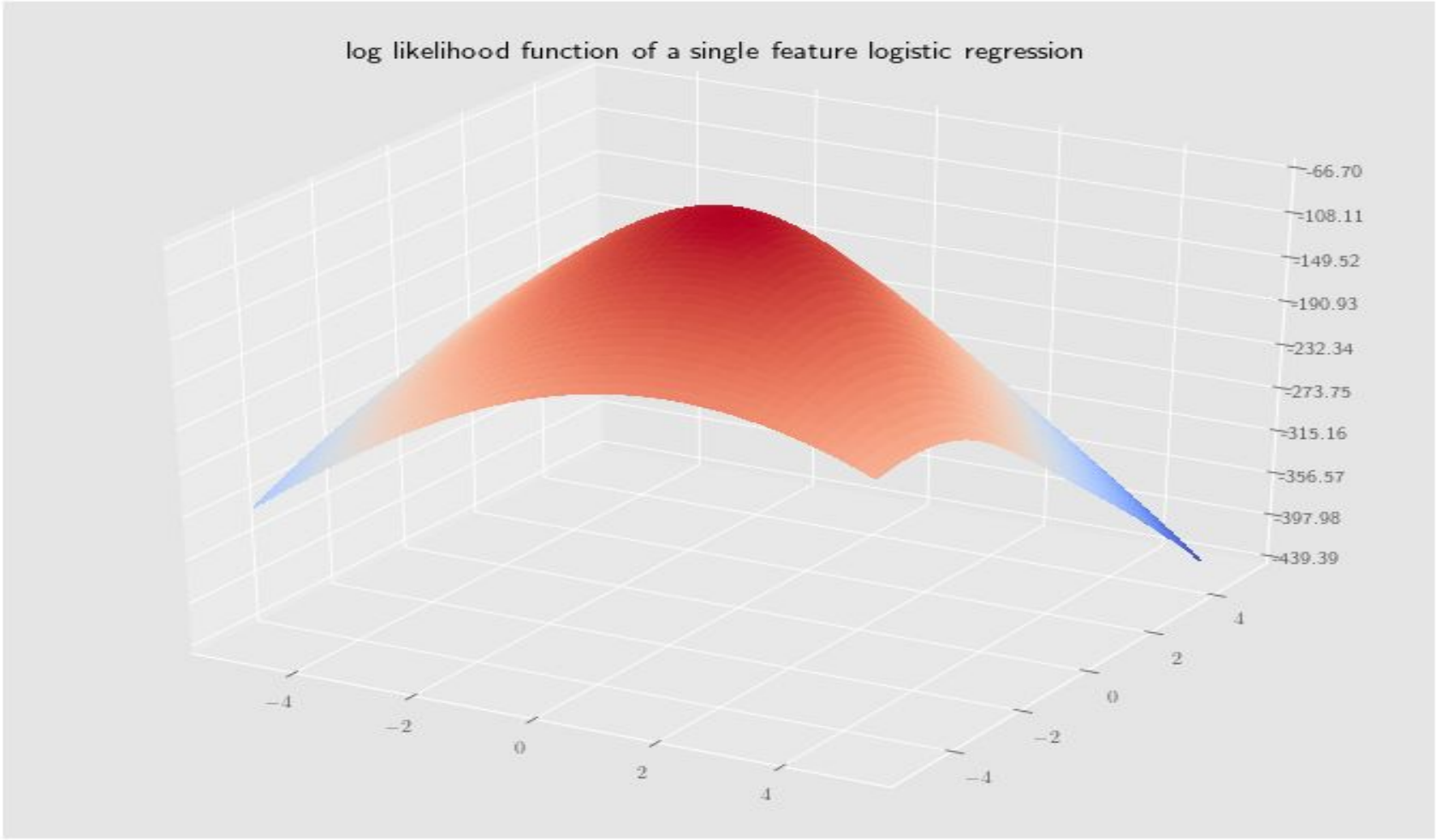
- The log likelihood function is a nonlinear concave function, which is known to have a unique maximal point. The convexity/concavity makes the procedure to maximize the log-likelihood well behaved
- The first term $\sum_{i \le n} y_i(\beta_0 + \sum_{j \le p} \beta_j X_{ij})$ is proportional to the covariance between the binary target and the linear inputs, which heuristically suggests the high covariance increases the log likelihood
- On the other hand, the first term in itself diverges when the coefficients $(\beta_0, \beta_1, \cdots, \beta_p)$ blow up. To compensate the linear growth of the first term, notice that $\sum_{i \le n} log(1 + e^{\beta_0 + \sum_{j \le p} \beta_j X_{ij}}) \cong \sum_{i \le n} (\beta_0 + \sum_{j \le p} \beta_j X_{ij})$. This allows the nonlinear second term to compensate the first linear terms for y =1
- This ensures the log-likelihood function to drop to negative infinity far from origin
  - We have made use of $log(1 + e^t) \cong log(e^t) = t$ above

Why is the Log-Likelihood Concave?

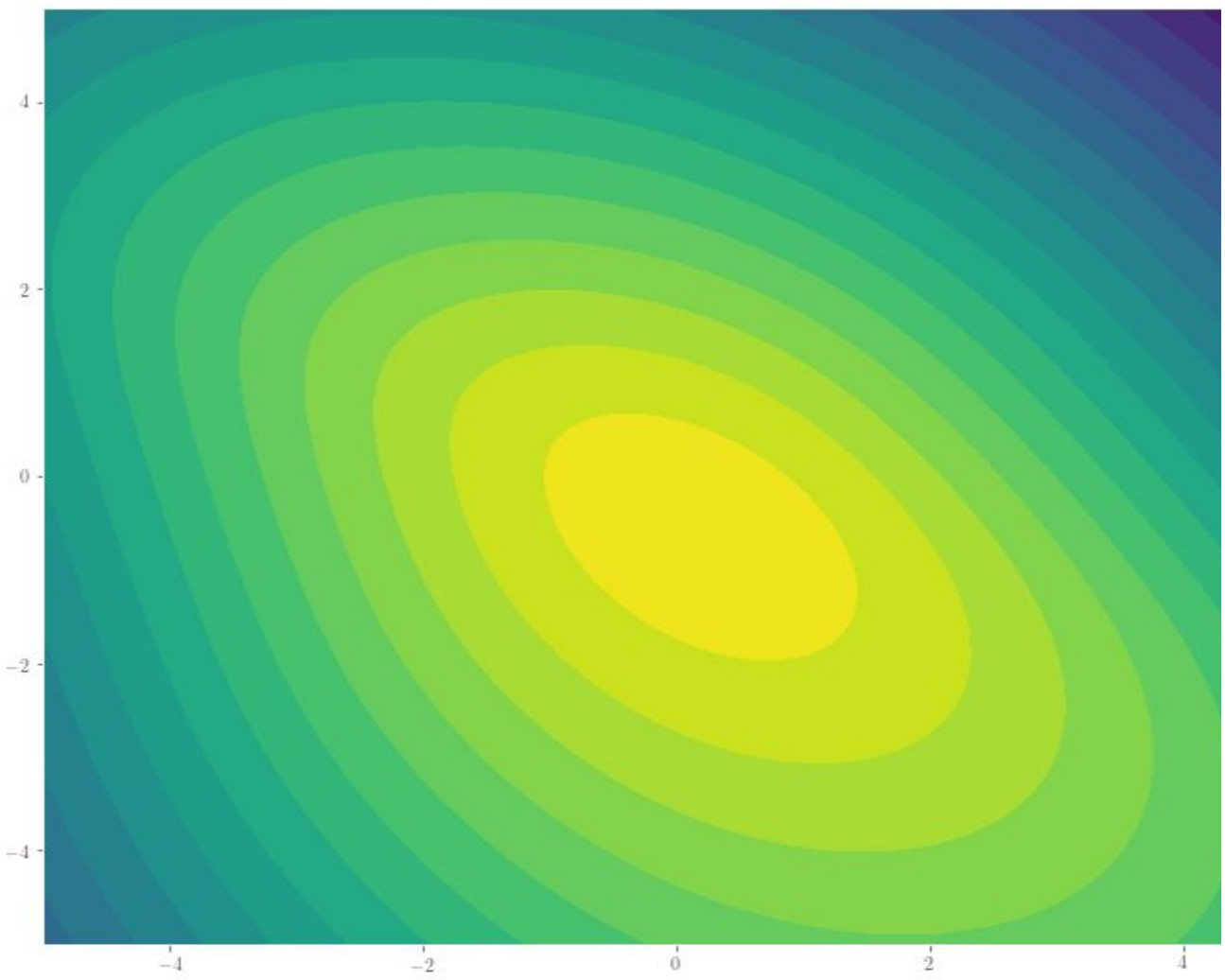
- Two key properties of $log(1 + e^t)$
  - It is asymptotic to the line y = x when x approaches + infinity
  - It is convex upward
- We also notice that log is a monotonically increasing function. Log-likelihood achieves maximum exactly when the likelihood itself achieves maximum



An example of Log Likelihood Function



The Contour Plot of the Previous Log Likelihood Function

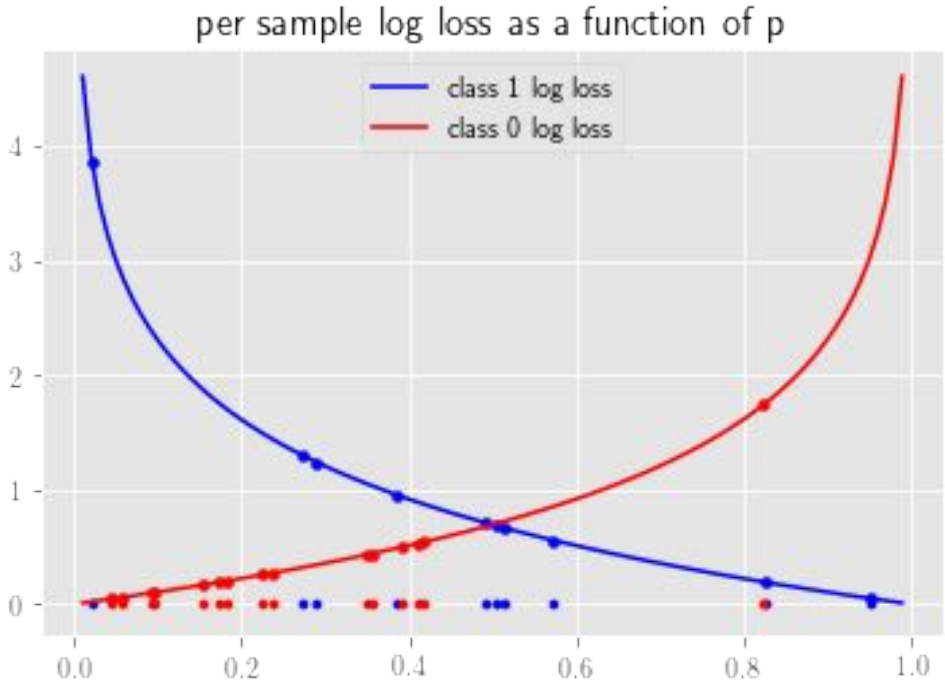


Log Loss and Deviance

- While our current goal is to maximize the log-likelihood function and this resulting model is the most likely model to fit the data, there is an alternative terminology in modern machine learning
- Not all models are probabilistic based. For these types of models, a loss function (also called score function) is used as the objective function and the model's goal is to minimize the loss function
- For linear regressions, we use **RSS** or **MSE** as the loss function
- For logistic regression, we use the **negation** of log likelihood instead
- Thus the concavity of log-likelihood translates directly to the convexity of the log-loss function
- In statistics literature, a related concept called **deviance** is, up to an added constant, minus 2 times log-likelihood function

Log Loss as a Sample Sum of Per-Sample Log Loss

- A set of slope and intercept parameters $\beta_0, \beta_i; 1 \leq i \leq p$ transforms a data set into a collection of probabilities through the sigmoid function $p_j = \frac{e^{\beta_0 + \beta \cdot X_j}}{1 + e^{\beta_0 + \beta \cdot X_j}}$. Changing these parameters will move the corresponding predicted probabilities



Logistic Regression Used in Medicine

- The following table records the disease treatment drug test result

	Original Treatment	New Treatment	Row Total
46-65 age group			
Died	43	6	49
Survived	100	34	134
Total	143	40	183
30-45 age group			
Died	109	11	120
Survived	148	69	217
Total	257	80	337

- We would like to know if the new drug is effective in reducing mortality

Converting to the Mortality Odds

	Original Treatment	New Treatment
46-65 Age Group		
Died:Survived Odds	0.43	0.176
Log Odds	-0.844	-1.737
30-45 Age Group		
Died:Survived Odds	0.736	0.159
Log Odds	-0.307	-1.839

- Based on the log odds of mortality rates, we may build a linear model to explain the differences. We use age (young vs old) and treatment type (original vs new) as binary categorical explanatory variables
- The resulting linear model can be described by

$$-2.121 + 0.454 \cdot age\_group + 1.333 \cdot treatment\_type$$

- Young age and original treatment are coded as 0

Estimating the Coefficients: Maximum Likelihood Estimation

- Assume we have a training dataset of 4 observations and 1 predictor variable. Then the probability of success for each observation can be predicted once a pair of β_0, β_1 is given. Let's assume the prediction below is given by a particular pair of β_0, β_1 .

True Label	Predicted Probability of Success	Predicted Probability of Failure
Success	0.5	0.5
Success	0.9	0.1
Failure	0.8	0.2
Failure	0.2	0.8

Estimating the Coefficients: Maximum Likelihood Estimation

- Assume we have another pair of β_0, β_1 (therefore a different model), then the prediction would be different from the previous slide. Assume we end up with this table:

True Label	Predicted Probability of Success	Predicted Probability of Failure
Success	0.9	0.1
Success	0.5	0.5
Failure	0.2	0.8
Failure	0.1	0.9

- Which one give us a better result overall?

Estimating the Coefficients: Maximum Likelihood Estimation

- One set of parameter values is more likely than another set if it gives the observed outcome a higher probability of occurrence.
  - Using maximum likelihood estimation in tandem with observed outcomes allows us to hone in on the best estimates of the coefficients.

- We wish to select values of $\beta_0, \beta_1, \ldots, \beta_p$ such that the likelihood of observing our particular dataset is as high as possible.
  - If we could iterate across many combinations of parameter estimate values, we could select the combination that produces the highest likelihood for our data.

- The parameter estimates that yield the highest likelihood for our data are called the maximum likelihood estimates MLEs.

Notes on Maximum Likelihood Estimation Calculation

- We alluded to potentially using a computer routine to search through many possible parameter values to find the MLEs; however, this is unnecessary.

- Unfortunately, whereas in simple/multiple linear regression we calculated closed-form expressions for the estimates, this does not occur in logistic regression setting.

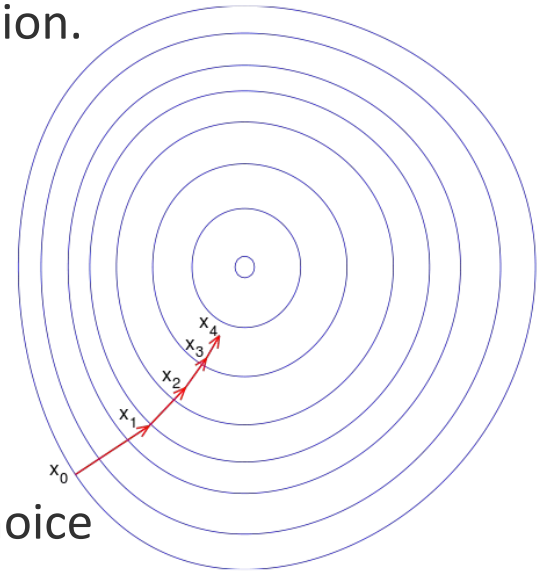
- For logistic regression, numerical methods are applied to search for the optimal coefficients. Gradient descent is a very popular algorithm for this.

Gradient Descent and Stochastic Gradient Descent

- The Idea of Gradient Descent
- The Stochastic Gradient Descent

Gradient Descent

- To maximize the log-likelihood, most packages, including scikit-learn, use a numerical method called ***gradient descent*** *or its variant*, i.e., to find the maximum or minimum by searching along a steepest path on the log-likelihood function.

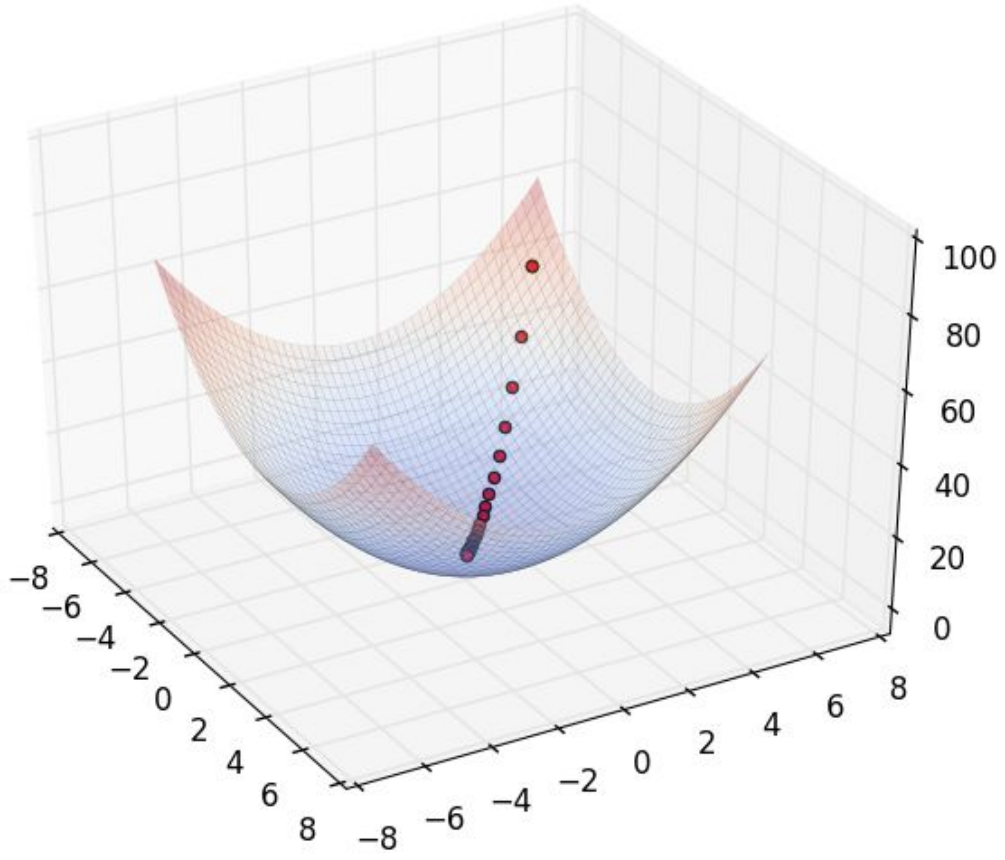


- Another popular choice

Is the Newton method or its variants, including **IRLS**.

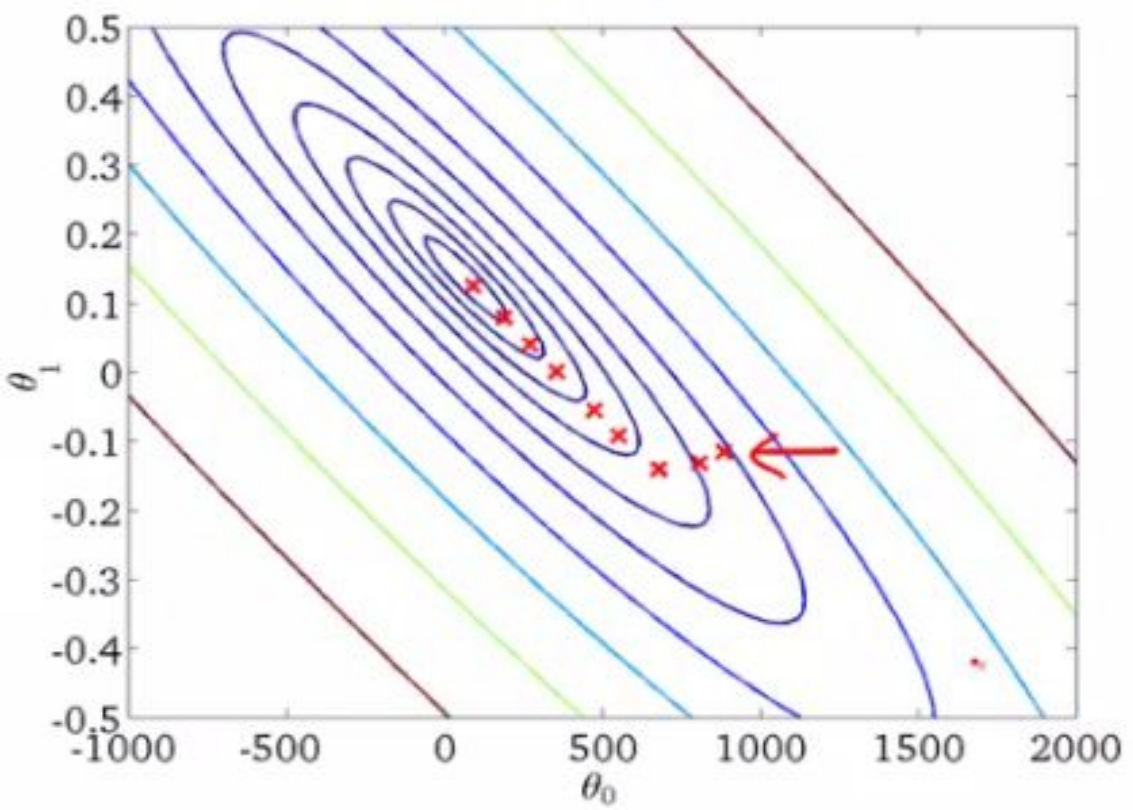
source: https://en.wikipedia.org/wiki/Gradient_descent

Gradient Descent



A Visualization of GD in 2D

- The descending path of GD is deterministic



The Idea of Gradient Descent in 1D

- Let us illustrate the idea of gradient descent in a 1D example:
- Consider a nice function (which can be differentiated) f:[a,b]$\rightarrow$R which we like to minimize. I.e. we would like to find a point s in [a, b] such that f(t) is minimal
- Not knowing where t is located, the gradient descent finds t through an iterative procedure where the true t is found in the limit
- In other words, we would like to find a sequence

$$x_1, x_2, \cdots, x_n \cdots$$

$$lim_{n \mapsto \infty} x_n = t$$

- We need to decide an initial guess x_1
- We also need to define an iterative procedure to define x_{n+1} in terms of x_n . . This is how the function gradient comes in to play

The Rationale of Gradient Descent

- The recursive formula involves first derivative,

$$x_{n+1} = x_n - \gamma \cdot \frac{df}{dx}$$

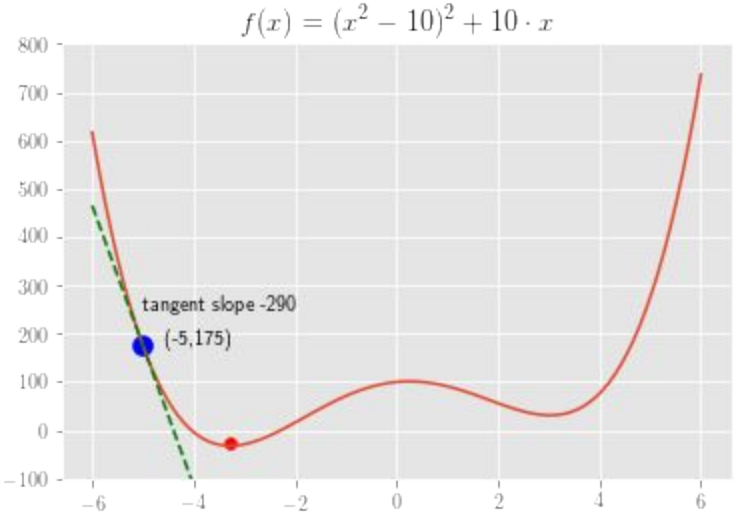
  where gamma is known as the step size or learning rate

- While negative gradient points to the descending direction, the parameter gamma controls how far to move
- Why does this formula guarantees $f(x_{n+1}) < f(x_n)$?
- We first notice that $f(x_{n+1}) \cong f(x_n) + (x_{n+1} - x_n)\frac{df}{dx}(x_n)$, from differentiable calculus
- On the other hand, the above recursive formula can be re-casted into $x_{n+1} - x_n = -\gamma \frac{df}{dx}(x_n)$, which can be plugged into the previous formula. We get

$$f(x_{n+1}) \cong f(x_n) - \gamma (f')^2(x_n) < f(x_n)$$

Why is the Learning Rate Relevant?

- To understand what role does the **learning rate** gamma play, let us look at the following example

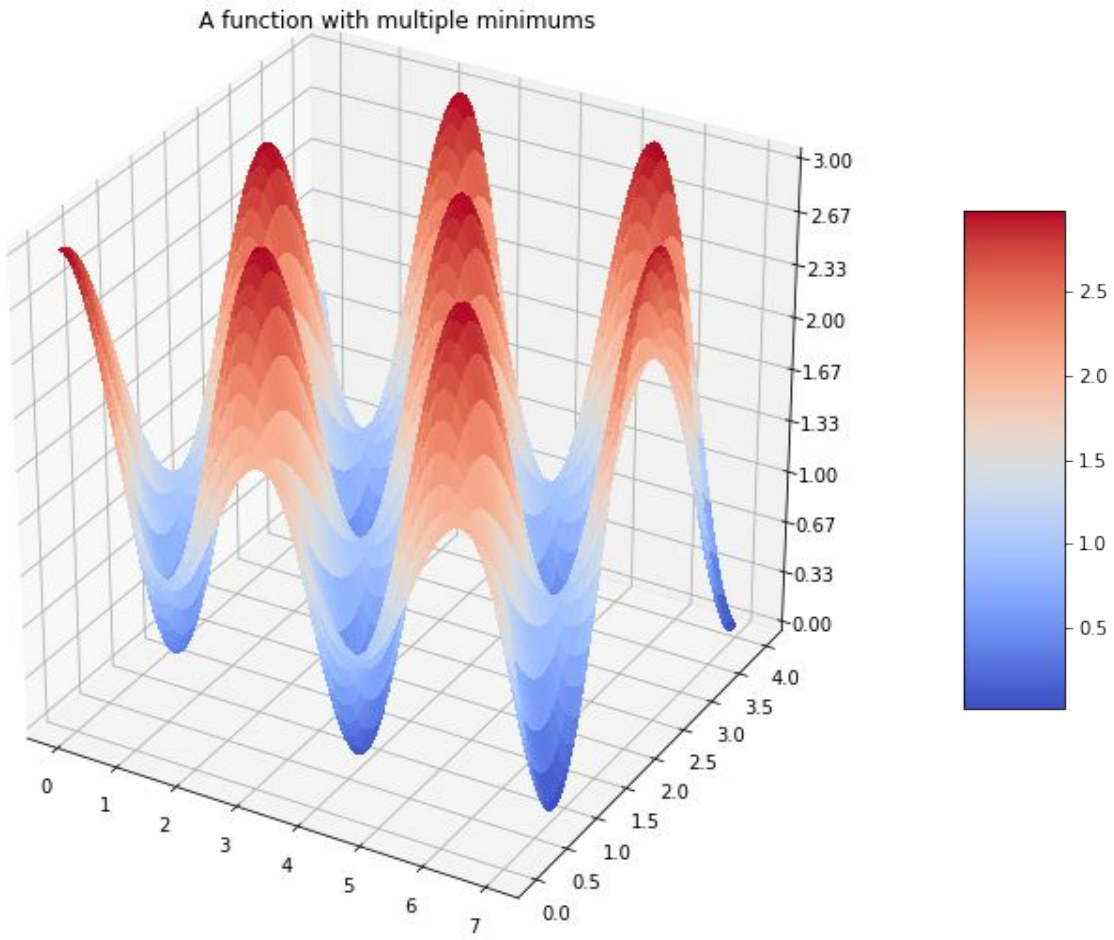


- In the above example the tangent slope at x=-5 is -290
- A naive approach setting x_{n+1} to be -5-(-290)= 285 will definitely overshoot

A Good Learning Rate

- In order for gradient descent algorithm to work properly, a properly tuned gamma needs to be chosen
- If the gamma is chosen too large, it will overshoot the minimum point even it moves in the right direction
- If the gamma is too small, it will waste many iterations to converge to the correct minimum
- For logistic regression, its log loss is concave, being a local minimum and being the global minimum are the same thing
- But for many other more sophisticated ML algorithms, gradient descent needs to combine with random initial conditions to avoid converging only to a local minimum

An Illustration of A Function with Multiple Local Minimums



Stochastic Gradient Descent

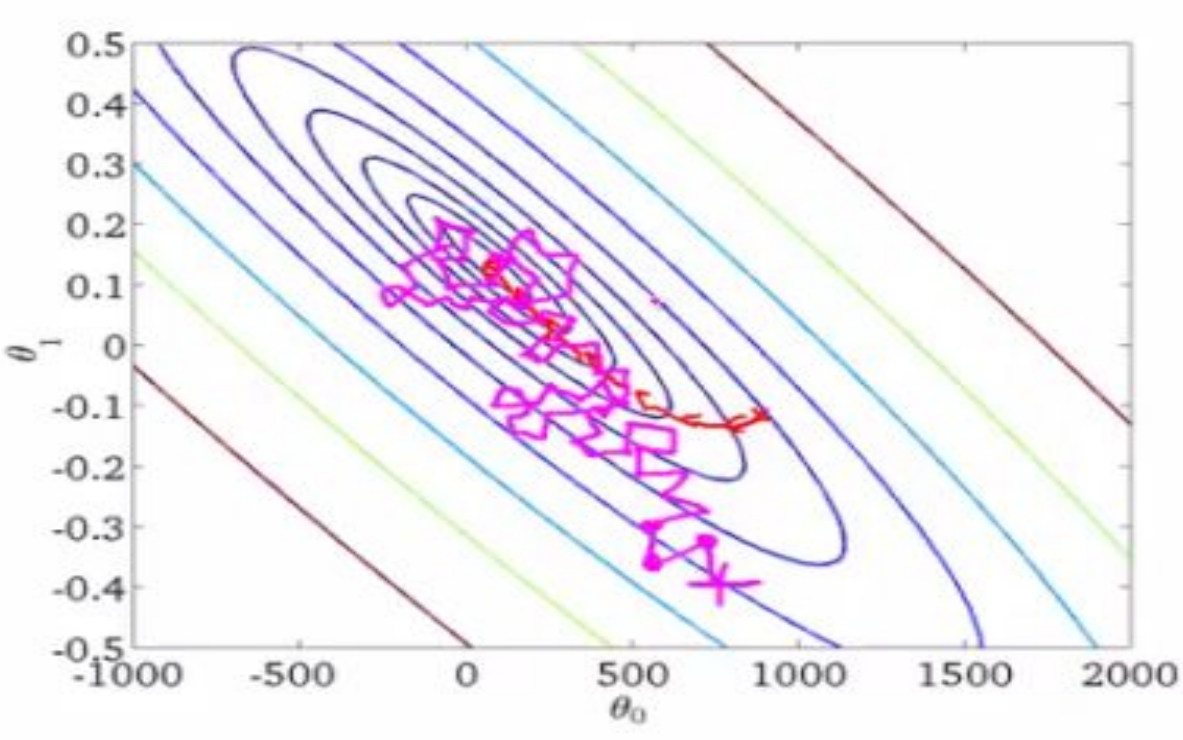
- The **empirical loss** function involves summing over all training samples
- Thus the computation loss gradient also involves all training samples
- This can become computationally intensive for large sample size/feature size
- The stochastic gradient descent is a modified **gradient descent** algorithm where the total sum over ALL samples is replaced by random sub-sampling in each iterative step (one sample at a time)
- The stochastic gradient descent has been proved to be an effective algorithm handling large data sets
  - It often converges faster than the traditional **gradient descent** algorithm
  - When the loss function has many local minimums, the randomness (i.e. the stochastic part) helps to avoid being trapped by a local minimum

SDG with Dynamical Shrinking Learning Rates

- In modern stochastic gradient descent algorithm, a dynamical shrinking learning rate algorithm is implemented
- The learning rate depends on the iterative step dynamically, which drops to zero as the number of iterations blows up to infinity
- The larger learning rate at the beginning helps to explore different regions within the parameter domain
- The dropping of learning rate for larger iterative steps helps to pin point to specific local minimum after enough exploration

A Visualization of SDG in 2D

- SGD introduces randomness into the gradient descent path



- This is known as the phenomenon of **random walk**, which we discuss in the time series analysis lecture in details

Making Predictions

- After estimating the parameters, the likelihood function $p(x, \beta_0, \beta_1)$ predicts the probability of output Y to be 1, given x. If we set a threshold, then we can predict binary outputs.
- Let's re-consider using tumor size to predict malignancy:
  - Hypothetically if the maximum likelihood gives $Pr(Y=1|X=x) = 0.2$, then our prediction is 20% chance of tumor being malignant, or equivalently, 80% chance it's benign.
  - If we set the threshold, for example, to be 0.5, or equivalently, at 50%, then we can predict the tumor to be benign.
  - Setting the cutoff at 0.5 is over-simplifying here. Even though the malignant cases are the real focus, their appearance is often much lower than 50% in real life. Setting the cutoff at 0.5 for such an **unbalanced** data set leads to uninteresting highly 'accurate' result.

Logistic Classifier and Multiclass Classification

- Logistic regression is naturally a binary classifier. In order to use logistic regression for handling multiclass classification task, scikit-learn implements an ensemble of logistic classifiers, which combines the wisdom of individual binary classifiers to handle the multiclass classification task. This is known as ‘ovr’, one vs the rest, or one vs all
- Another alternative way is to generalize into a **multinomial logistic regression** which handles multiclass classification naturally

Logistic Regression and Generalized Linear Models

- In logistic regression, we use log odds (logit) as our link function
- The generalized linear model replaces the logit link function by general link functions associated with the **exponential** family of probability distributions
- The well known examples are like poisson regression, probit regression, binomial regression, multinomial regression, exponential regression, gamma regression, etc
- We will not go into details to their model construction. Only mention that the common theme is a nonlinear link function relating the mean of the specified exponential distribution to the linear regression