STA 314: Statistical Methods for Machine Learning I

Lecture 6 - Classification: the Bayes rule and logistic regression

Xin Bing

Department of Statistical Sciences
University of Toronto

Classification

The response variable Y is qualitative, taking values in an unordered set C

- email is $C = \{spam, non - spam\}$
- digit is $C = \{0, 1, ..., 9\}$
- eye color is $\{brown, blue, green\}$

Given the training data: $\mathcal{D}^{train} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, with $y_i \in C$ and $x_i \in \mathbb{R}^p$, our goals are to:

- Build a classifier (a.k.a. a rule) $\hat{f} : X \to C$ that assigns a class label in C to a future observation x.
- Assess the accuracy of this classifier $\hat{f}$
- Understand the roles of different features in $\hat{f}$.

The metric used in classification

Let (X, Y) be a random pair, independent of $\mathcal{D}^{train}$. Let us encode the labels as

$$C = \{1, 2, ..., K\}.$$

For any classifier $\hat{f}$, recall that we should evaluate it based on its **expected error rate**

$$\mathbb{E}\Big[1\{Y \neq \hat{f}(X)\}\Big].$$

Question: what is the best classifier?

The Bayes classifier

The best $\hat{f}(X)$ which minimizes the expected error rate is a classifier that assigns each observation to its most probable class.

This is known as **the Bayes classifier** (a.k.a. the Bayes rule). We use f^* to denote it.

Mathematically, for any $X = x$,

$$f^*(x) = j, \qquad \text{if} \quad j = \arg\max_{1 \le k \le K} \mathbb{P}\{Y = k \mid X = x\}.$$

For example, if $C = \{1, 2\}$,

$$f^*(x) = \begin{cases} 1 & \text{if } \mathbb{P}\{Y = 1 \mid X = x\} \ge 0.5; \\ \\ 0 & \text{if } \mathbb{P}\{Y = 0 \mid X = x\} \ge 0.5. \end{cases}$$

The Bayes Error Rate

Correspondingly, the expected error rate of the Bayes classifier is the smallest possible error rate, called the **Bayes error rate**.

- This is analogous to the irreducible error in regression problems.

The expected error rate of f^* at $X = x$ is

$$\mathbb{E}\left[1\{Y \neq f^*(x))|X = x\right] = 1 - \max_{1 \leq j \leq K} \mathbb{P}\{Y = j \mid X = x\}.$$

- The Bayes error rate is non-negative, and typically greater than zero.

The Bayes classifier is our target to estimate / learn in classification problems.

Binary classification

In binary classification, $C = \{1, 2\}$ and the Bayes classifier is

$$f^*(x) = \begin{cases} 1 & \text{if } \mathbb{P}\{Y = 1 \mid X = x\} \geq 0.5; \\ 0 & \text{if } \mathbb{P}\{Y = 0 \mid X = x\} \geq 0.5. \end{cases}$$

Learning the Bayes classifier reduces to estimate the conditional probability

$$p(X) := \mathbb{P}\{Y = 1 \mid X = x\},$$

a function of X.

How to do this? The same paradise:

- Parametric methods
- Non-parametric methods

Why Not Regression?

- In the binary case,

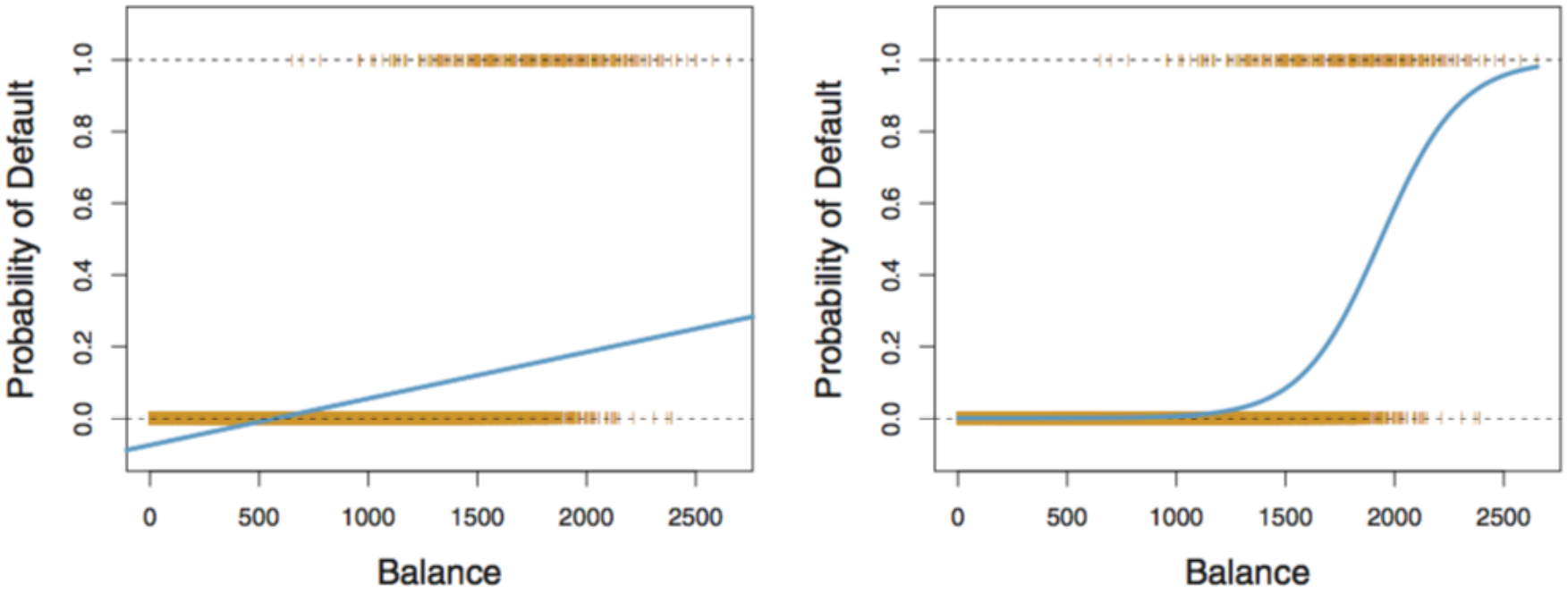
$$p(X) = \mathbb{P}\{Y = 1 \mid X\} = \mathbb{E}[Y \mid X].$$

  Recall the regression setting,

$$Y = f(X) + \epsilon = \mathbb{E}[Y \mid X] + \epsilon.$$

- Can we use the regression approach (such as OLS) to estimate $\mathbb{E}[Y \mid X]$?
  - Yes, we could (as commonly done in practice).
  - However, linear regression might produce $\hat{p}(X)$ less than zero or bigger than one.
  - A more tailored approach is needed!

Linear Regression versus Logistic Regression in binary classification



- Left: Estimated probability of default using linear regression. Some estimated probabilities are negative! The orange points represents the 0/1 values coded for default (No or Yes).
- Right: Predicted probabilities of default using logistic regression. All probabilities lie between 0 and 1.

Logistic Regression

Logistic Regression is a parametric approach that assumes parametric structure on

$$p(X) = \mathbb{P}(Y = 1 \mid X) = \mathbb{E}[Y \mid X].$$

- Logistic regression assumes the following structure on $p(X)$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}.$$

  The function $f(t) = e^t/(1 + e^t)$ is called the logistic function. $\beta_0, \ldots, \beta_p$ are the parameters.

- It is easy to see that we always have $0 \leq p(X) \leq 1$.
- Note that $p(X)$ is **NOT** a linear function either in X or in β.

Logistic Regression

- A bit of rearrangement gives

$$\underbrace{\frac{p(X)}{1-p(X)}}_{odds} = e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p},$$

$$\underbrace{\log\left[\frac{p(X)}{1-p(X)}\right]}_{log-odds} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p.$$

  odds $\in [0, \infty]$ and log-odds $\in [-\infty, \infty]$.

- Similar interpretation as linear models:
  each β_j represents the change of log-odds for one unit increase in X_j (with other features held fixed).

- How to estimate $\beta_0, \ldots, \beta_p$?

Maximum Likelihood Estimator (MLE)

Given $\mathcal{D}^{train} = \{(x_1, y_1), ..., (x_n, y_n)\}$ with $y_i \in \{0, 1\}$, we estimate the parameters by **maximizing the likelihood**.

- The maximum likelihood principle is that we seek the estimates of parameters such that the fitted probability are the closest to the individual's observed outcome.

The **likelihood function** of the observed data is

$$L(\beta_0, \beta_1, \cdots, \beta_p; \mathcal{D}^{train}) = \prod_{i=1}^{n} [p(x_i)]^{y_i} [1 - p(x_i)]^{1-y_i}.$$

We maximize the above likelihood over $(\beta_0, \ldots, \beta_p)$ and the resulting estimator is called **the Maximum Likelihood Estimator** (MLE).

Example: Default data

Consider the Default data using balance, income, and student status as predictors.

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}$$

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-10.8690	0.4923	-22.08	< 0.0001
`balance`	0.0057	0.0002	24.74	< 0.0001
`income`	0.0030	0.0082	0.37	0.7115
`student[Yes]`	-0.6468	0.2362	-2.74	0.0062

Inference under logistic regression

- Z-statistic is similar to t-statistic in regression, and is defined as

$$\frac{\hat{\beta}_j}{SE(\hat{\beta}_j)}, \qquad \forall j \in \{0, 1, \ldots, p\}.$$

- It produces p-value for testing the null hypothesis $H_0 : \beta_1 = 0$.
  A large (absolute) value of the z-statistic or small p-value indicates evidence against H_0.

- The Z-statistic is built on the statistical properties of the MLE, $\hat{\beta}_0, \cdots, \hat{\beta}_p$.
  - A general theory for the MLE.

Prediction of $\mathbb{P}(Y = 1 \mid X)$

Consider the Default data with student status as the only feature. What is our estimated probability of default for a student?

To fit the model, we encode Y as 1 for student and 0 otherwise.

	Coefficient	Std. Error	Z-statistic	P-value
`Intercept`	-3.5041	0.0707	-49.55	< 0.0001
`student[Yes]`	0.4049	0.1150	3.52	0.0004

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431,$$

$$\widehat{\Pr}(\texttt{default=Yes}|\texttt{student=No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292.$$

Computation of the MLE under Logistic Regression

For simplicity, let us set $\beta_0 = 0$ such that

$$p(x) = \frac{e^{x^\top \beta}}{1 + e^{x^\top \beta}}, \qquad 1 - p(x) = \frac{1}{1 + e^{x^\top \beta}}.$$

The log-likelihood at any β is

$$\begin{aligned}
\ell(\beta) &= \log \left\{ \prod_{i=1}^{n} [p(x_i)]^{y_i} [1 - p(x_i)]^{1-y_i} \right\} \\
&= \sum_{i=1}^{n} [y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))] \\
&= \sum_{i=1}^{n} \left[y_i \log \left(\frac{p(x_i)}{1 - p(x_i)} \right) + \log(1 - p(x_i)) \right] \\
&= \sum_{i=1}^{n} \left[y_i x_i^\top \beta - \log \left(1 + e^{x_i^\top \beta} \right) \right].
\end{aligned}$$

Gradient Descent for Logistic Regression

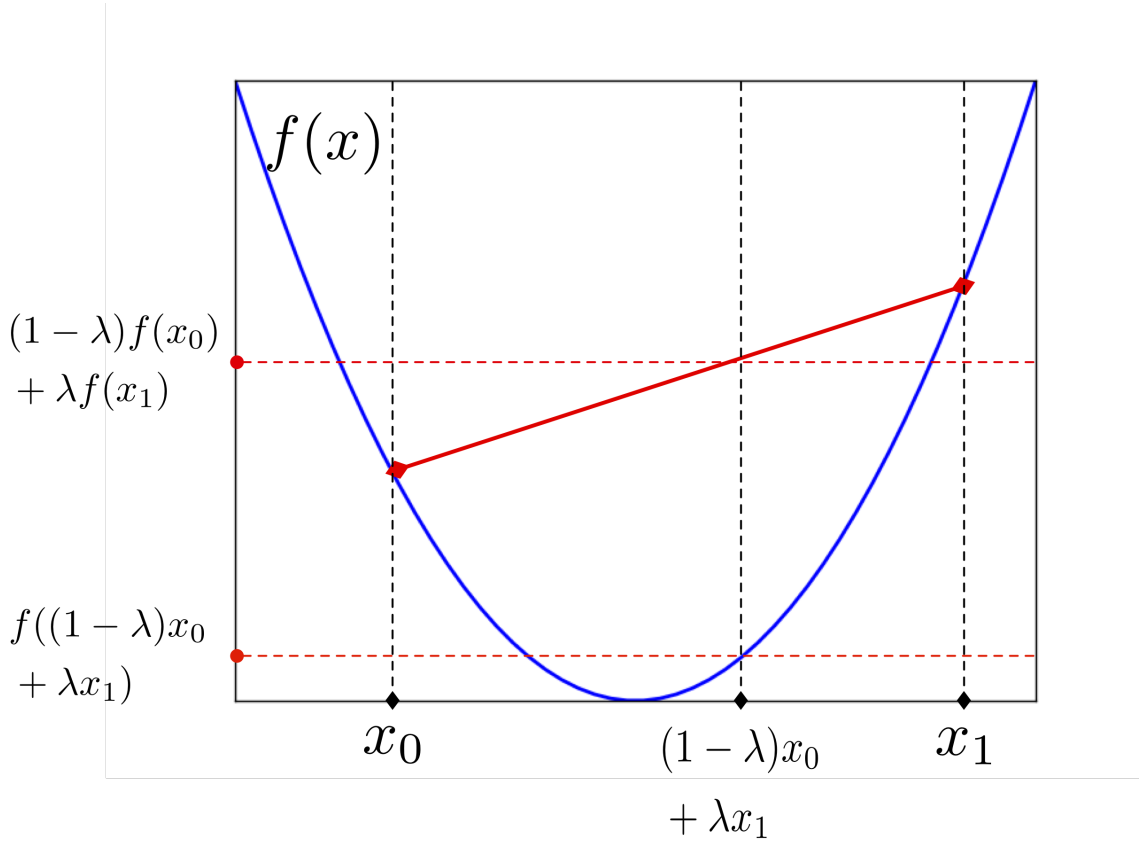
- How do we maximize the log-likelihood $\ell(\boldsymbol{\beta})$ for logistic regression?
  - It is equivalent to minimize the $-\ell(\boldsymbol{\beta})$.
  - No direct solution: taking derivatives of $-\ell(\boldsymbol{\beta})$ w.r.t. $\boldsymbol{\beta}$ and setting them to 0 doesn't have an explicit solution.
- Perhaps we should consider using the gradient descent from the last lecture? But will this work?
- Luckily it will, but we should be a bit careful to understand why it works.
- It works because the logistic loss is a convex function **in** $\boldsymbol{\beta}$.

Convex Functions

- A function f is convex if for any $\mathbf{x}_0, \mathbf{x}_1$ in the domain of f,

$$f((1-\lambda)\mathbf{x}_0 + \lambda\mathbf{x}_1) \leq (1-\lambda)f(\mathbf{x}_0) + \lambda f(\mathbf{x}_1)$$

- Equivalently, the set of points lying above the graph of f is convex.
- Intuitively: the function is bowl-shaped.



How to tell a loss is convex?

- Verify the definition.
- If f is twice differentiable and $f''(x) \geq 0$ for all x, then f is convex.
  - the least-squares loss function $(y - t)^2$ is convex as a function of t
  - the function

    $$-yt + \log\left(1 + e^t\right)$$

    is convex in t.
- There are more sufficient conditions for convex but non-differentiable functions!

- A composition rule: linear functions preserve convexity. E.g.
  - If f is a convex function and g is a linear function, then $f \circ g$ is a convex function.
    - the least-square loss $(y - x^\top \beta)^2$ is convex in β
    - the negative log-likelihood under logistic regression

      $$-yx^\top \beta + \log\left(1 + e^{x^\top \beta}\right)$$

      is convex in β.
  - Both $\sum_i (y_i - x_i^\top \beta)^2$ and $\sum_i \left[-y_i x_i^\top \beta + \log\left(1 + e^{x_i^\top \beta}\right)\right]$ are convex in β.

- There are more composition rules!

- A great book:

  Convex Optimization, Stephen Boyd and Lieven Vandenberghe.

Gradient descent for solving the MLE under logistic regression

- The key point here is that the logistic loss is convex.
- Convex functions have very nice properties.
  - All critical points are minima.
  - Gradient descent finds the optimal solution.
- So we can use gradient descent to find the minima of the logistic loss!
  - Recall: we initialize the weights to something reasonable and repeatedly adjust them in the direction of the steepest descent.
  - A standard initialization is $\beta = 0$.

Gradient descent for solving the MLE under logistic regression

Recall

$$-\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left[-y_i x_i^\top \boldsymbol{\beta} + \log\left(1 + e^{x_i^\top \boldsymbol{\beta}}\right) \right].$$

The gradient at any $\boldsymbol{\beta}$ is

$$\frac{\partial - \ell(\boldsymbol{\beta})}{\partial \beta_j} = \sum_{i=1}^{n} \left[-y_i + \frac{e^{x_i^\top \boldsymbol{\beta}}}{1 + e^{x_i^\top \boldsymbol{\beta}}} \right] x_{ij} \qquad \text{(verify this!)}$$

Therefore, at the $(k+1)$th iteration,

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} - \alpha \sum_{i=1}^{n} \left[-y_i + \frac{e^{x_i^\top \boldsymbol{\beta}^{(k)}}}{1 + e^{x_i^\top \boldsymbol{\beta}^{(k)}}} \right] x_i.$$

Batch Gradient Descent

- Recall that
  - OLS:

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} + \alpha \sum_{i=1}^{n} \left[y_i - x_i^\top \boldsymbol{\beta}^{(k)} \right] x_i.$$

  - Logistic regression:

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} + \alpha \sum_{i=1}^{n} \left[y_i - \frac{e^{x_i^\top \boldsymbol{\beta}^{(k)}}}{1 + e^{x_i^\top \boldsymbol{\beta}^{(k)}}} \right] x_i.$$

- Computing the gradient requires summing over *all* of the training examples. This is known as batch training.

Stochastic Gradient Descent

- Batch training is impractical if you have a large dataset (e.g. millions of training examples, $n \approx 10$ millions)!
- Stochastic gradient descent (SGD): update the parameters based on the gradient for a single training example,
  1. Choose $i \in \{1, \ldots, n\}$ uniformly at random
  2. Update the parameters by ONLY using this ith sample,

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} + \alpha \left[y_i - x_i^\top \boldsymbol{\beta}^{(k)} \right] x_i$$

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} + \alpha \left[y_i - \frac{e^{x_i^\top \boldsymbol{\beta}^{(k)}}}{1 + e^{x_i^\top \boldsymbol{\beta}^{(k)}}} \right] x_i.$$

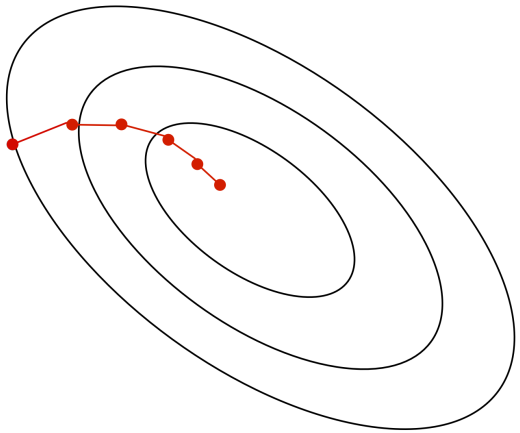
- Computational cost of each SGD update is independent of n!
- SGD can make significant progress before even seeing all the data!
- Mathematical justification: the gradients between SGD and GD have the same expectation for i.i.d. data.

Stochastic Gradient Descent

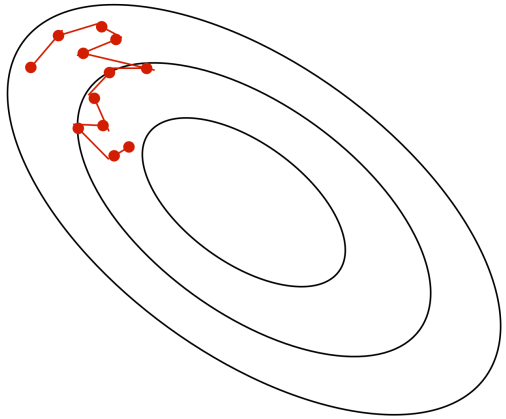
- Problems with using single training example to estimate gradient:
  - Variance in the estimate may be high
  - We can't exploit efficient vectorized operations
- Compromise approach:
  - compute the gradients on a randomly chosen medium-sized set of training examples $\mathcal{M} \subset \{1, \ldots, n\}$, called a mini-batch.
- Stochastic gradients computed on larger mini-batches have smaller variance.
- The mini-batch size $|\mathcal{M}|$ is a hyperparameter that needs to be set.

Stochastic Gradient Descent

- Batch gradient descent moves directly downhill. SGD takes steps in a noisy direction, but moves downhill on average.



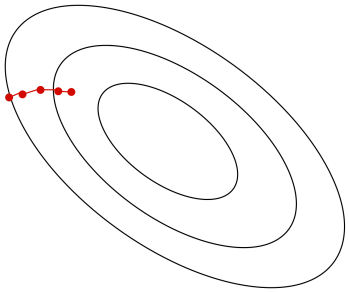
batch gradient descent



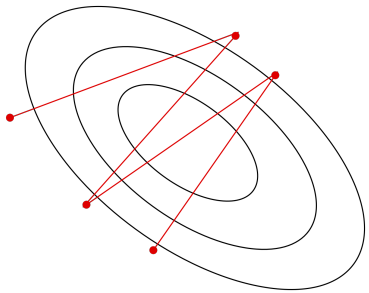
stochastic gradient descent

Learning Rate

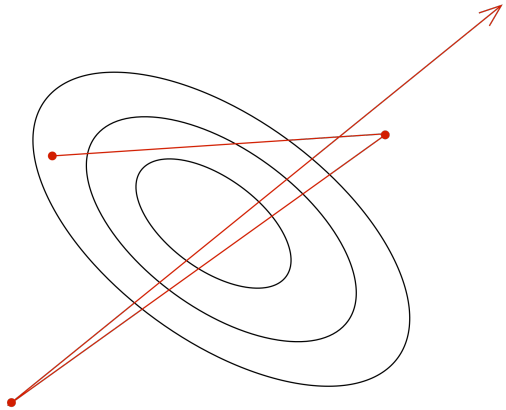
- In gradient descent, the learning rate α is a hyperparameter we need to tune. Here are some things that can go wrong:



α too small:
slow progress



α too large:
oscillations



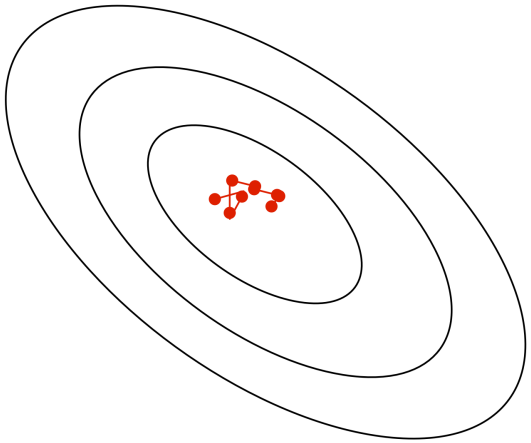
α much too large:
instability

- Good values are typically small. You should do a grid search if you want good performance (i.e. try $0.1, 0.03, 0.01, \ldots$).

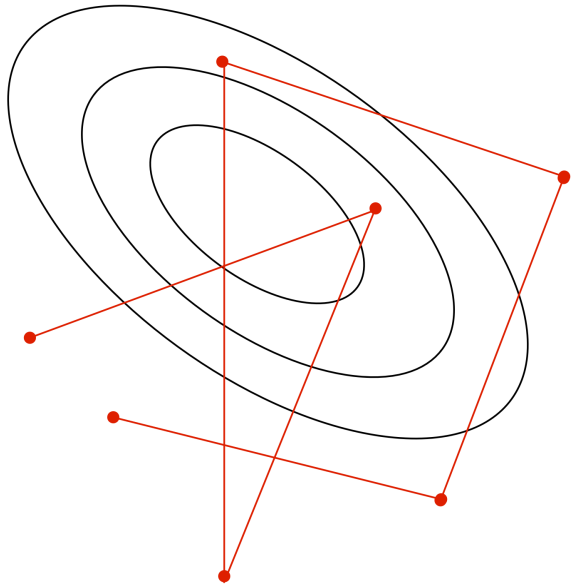
SGD Learning Rate

- In stochastic training, the learning rate also influences the fluctuations due to the stochasticity of the gradients.

small learning rate



large learning rate



- Typical strategy:
  - Use a large learning rate early in training so you can get close to the optimum
  - Gradually decay the learning rate to reduce the fluctuations