The Mathemmatical Engineering of Deep Learning

Chapter 2 - Lecture 2 - Part (2/3)

B. Liquet[1,2] and S. Moka[3] and Y. Nazarathy[3]

[1] Macquarie University [2] LMAP, Université de Pau et des Pays de L'Adour' [3] The University of Queensland

Outline of Lecture

- Convexity
  - Linear case
  - Logistic case
- Optimization
  - Gradient descent
  - Regularization
- Implementation

Convex Set

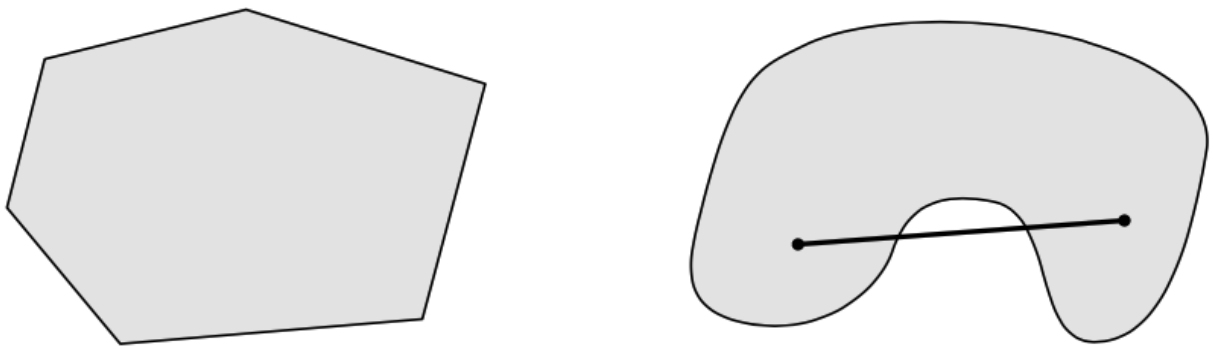
Definion

A set C is convex if, for any $x, y \in C$ and $\theta \in \mathfrak{R}$ with $0 \leqslant \theta \leqslant 1$,

$$\theta x + (1 - \theta) y \in C.$$

- take any two elements in C
- draw a line segment between these two elements

every point on that line segment also belongs to C



(a) (b)

Which set is a convex set ?

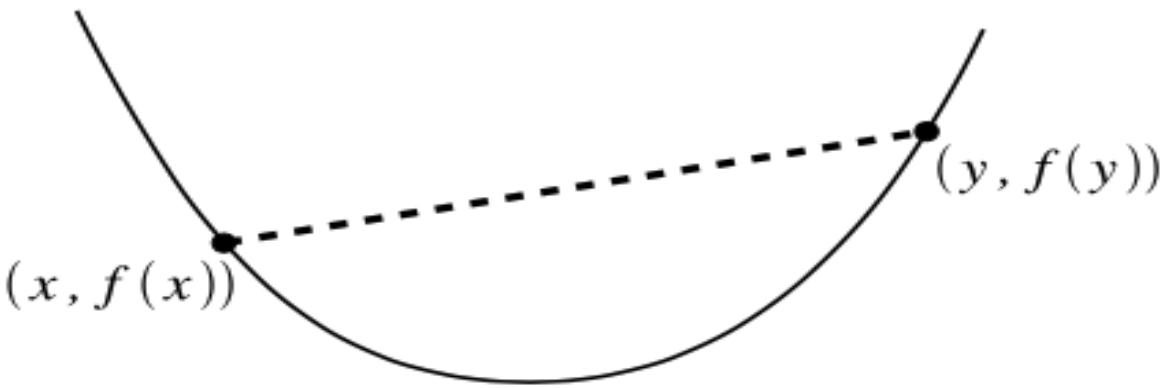
Convex Functions

Definion

A function $f : \mathfrak{R}^n \rightarrow \mathfrak{R}$ is convex if its domain ($\mathcal{D}(f)$) is a convex set, and if, all $x, y \in \mathcal{D}(f)$ and $\theta \in \mathfrak{R}$, $0 \leqslant \theta \leqslant 1$,

$$f(\theta x + (1 - \theta)y) \leqslant \theta f(x) + (1 - \theta)f(y)$$

- Pick any two points on the graph of a convex function
- draw a straight line between then the portion of the function between these two points will lie below this straight line.



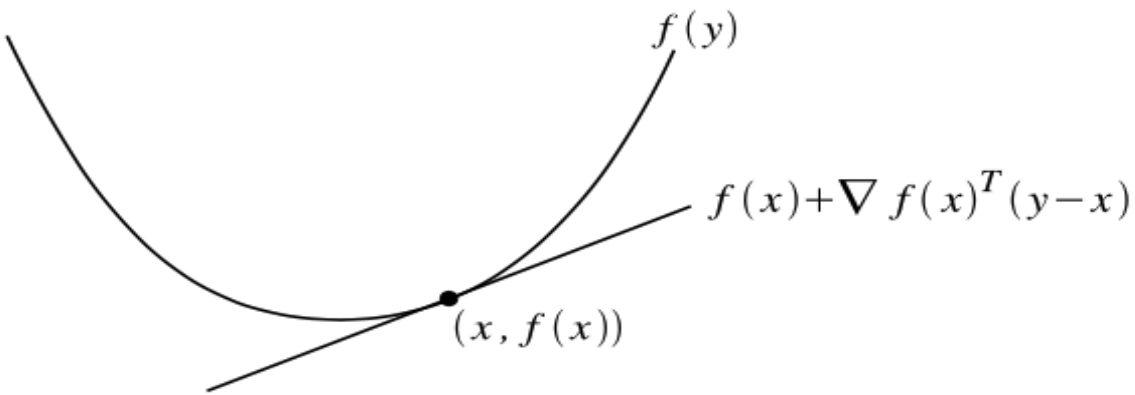
Convex Functions

First Order Condition

Assume a function $f : \mathfrak{R}^n \to \mathfrak{R}$ is differentiable. Then f convex if is convex if and only if $\mathcal{D}(f)$ is a convex set and for all $x, y \in \mathcal{D}(f)$,

$$f(y) \leqslant f(x) + \nabla_x f(x)^T (y - x).$$

- f is convex if and only if the tangent line is a global underestimator of the function f.
- Meaning: draw a tangent line at any point, then every point on this line will lie below the corresponding point on f.



Convex Functions

Second Order Condition

Assume a function $f : \Re^n \to \Re$ is twice differentiable. Then f convex if is convex if and only if $\mathcal{D}(f)$ is a convex set and its Hessian is positive semidefinite: i.e., for any $x \in \mathcal{D}(f)$,

$$\nabla_x^2 f(x) \succcurlyeq 0.$$

- In one dimension, this is equivalent to the condition that the second derivative $f''(x)$ always be positive
- reminder: an $n \times n$ symmetric real matrix M is positive semidefinite if $x^T M x \geqslant 0$ for all $x \in \Re^n$
- $\nabla_x^2 f(x) \in \Re^{n \times n}$, $(\nabla_x^2 f(x))_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$

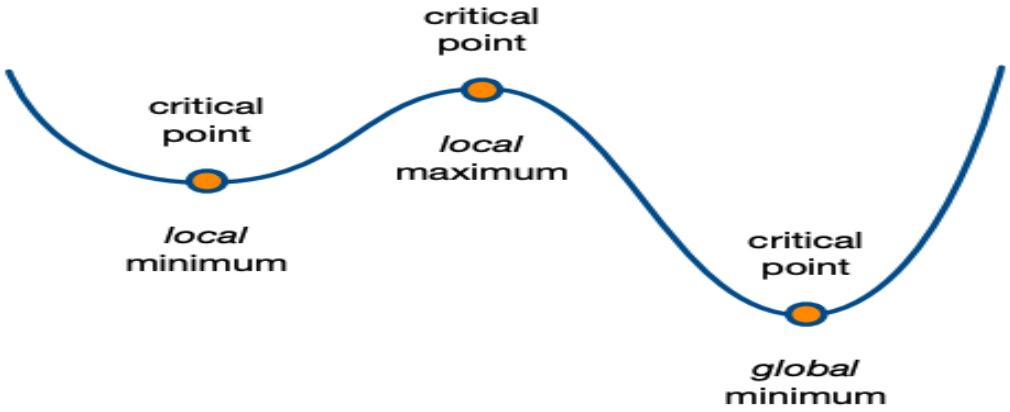
Examples

- **Exponential.** Let $f : \mathfrak{R} \to \mathfrak{R}$, $f(x) = \exp(ax)$ for any $a \in \mathfrak{R}$.
- **Negative logarithm.** Let $f : \mathfrak{R} \to \mathfrak{R}$, $f(x) = -\ln x$ for any $a \in \mathfrak{R}$.
- **Affine functions.** Let $f : \mathfrak{R}^n \to \mathfrak{R}$, $f(x) = b^T x + c$ for some $b \in \mathfrak{R}^n,\ c \in \mathfrak{R}$.
- **Norms.** Let $f : \mathfrak{R}^n \to \mathfrak{R}$,. Use triangle inequality.
- **Non-negative weighted sums of convex functions.** Let $f_1, f_2, \ldots, f_k$ be convex functions and $w_1, w_2, \ldots, w_k$ be nonnegative real numbers. Then

$$f(x) = \sum_{i=1}^{k} w_i f_i(x)$$

  is a convex function.

Why convexity is important



- Gradient descent finds a critical point, but it may be a local optima.
- Convexity is a property that guarantees that all critical points are global minima.
- For convex function, all critical points are minima

Operations that Preserve Convexity

Composition

Composition of a convex function with an affine map

$$f(x) = g(Ax + b)\ \forall x \in \Re^d$$

where $g : \Re^n \to \Re$ is convex, $A^{n \times d}$ is a matrix and $b \in \Re^n$

Composition

Maximum of convex functions

$$f = max\{f_1, \ldots, f_q\}$$

Road map to prove convexity

- From the definition of convexity
- Compute the Hessian of f and show that it is psd.
- Decompose f as a non-negative weighted sum of convex functions
- Decompose f as the composition of a convex and an affine function
- Decompose f as a maximum of convex functions

Linear model

- The least-square loss $\frac{1}{2}(y - \widehat{y})^2$ is convex as a function of $\widehat{y}$
- For a linear model: $z = w^T x + b$ is a linear function of w and b.
- If the loss function is convex as a function of z, then it is convex as a function of w and b.

Ridge Regression

To do as homework:

- Show that the following function $f(w)$ is convex:

$$f(w) = \sum_{i=1}^{m} (y_i - w^T x_i)^2 + \gamma ||w||^2,$$

where $\gamma > 0$.

- Solve this optimization problem

$$\min_{w \in \Re^d} f(w)$$

Logistic regression: cross-entropy

Show that the cross entropy loss for the logistic model (sigmoid function for the shallow NN) is a convex function.

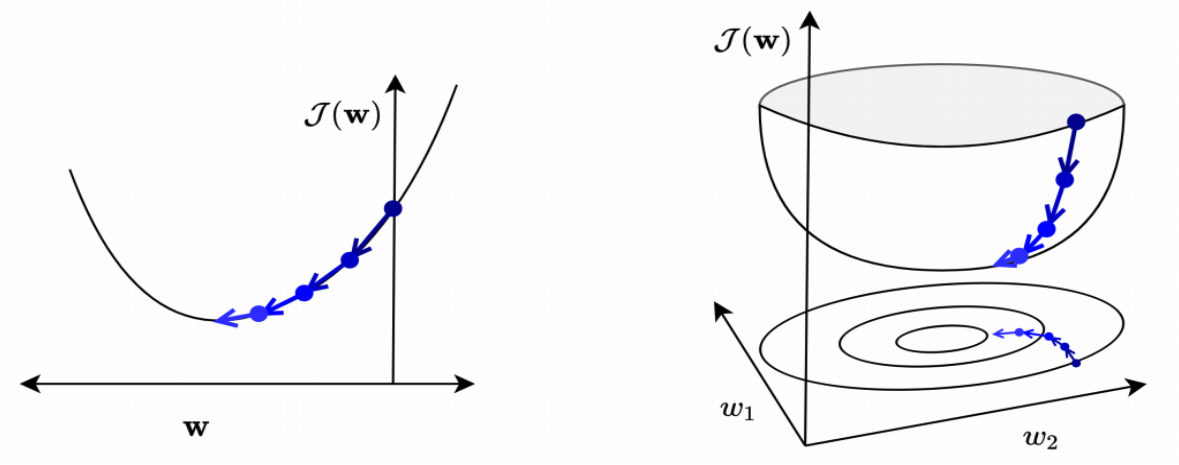
HINT: use the **Second Order Condition**

Linear Regression

- The squared error loss of linear regression is a convex function.
- Even for linear regression, where there is a direct solution, we sometimes need to use GD.
- Why gradient descent, if we can find the optimum directly?
  - GD can be applied to a much broader set of models
  - GD can be easier to implement than direct solutions
  - For regression in high-dimensional space, GD is more efficient than direct solution
    - Solution: $(X^T X)^{-1} X^T y$
    - Cost for matrix inversion $\mathcal{O}(d^3)$
    - Each update of SG costs $\mathcal{O}(md)$
    - Less with shochastic GD

Gradient Descent: why ?

- Taking derivatives of the cost function w.r.t to the weight and setting them to 0 doesn't have an explicit solution.
- Gradient descent is an iterative algorithm, which means we apply an update repeatedly until some criterion is met.
- Initialize the weights to something reasonable and repeatedly adjust them in the direction of steepest descent.



Gradient Descent

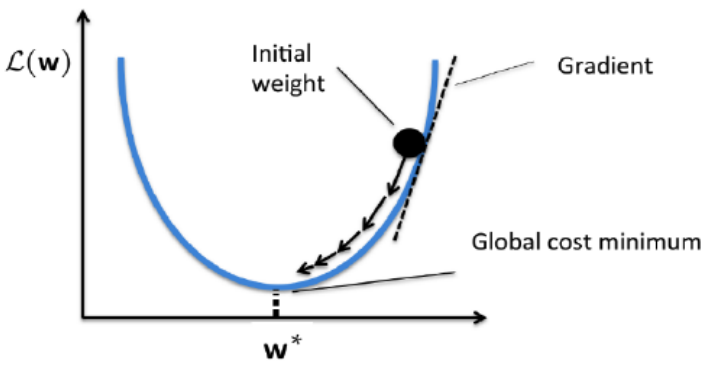
Consider this optimization problem: $\min_{w \in \mathbb{R}^d} \mathcal{L}(w)$

Principle: update rule

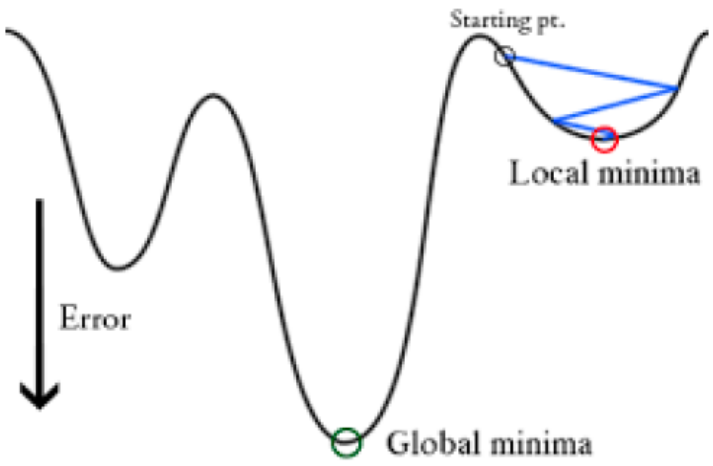
$$w^{(k+1)} = w^{(k)} - \alpha . \nabla \mathcal{L}(w^{(k)}), \quad k = 1, 2, 3, \ldots$$

Here, α is called the learning rate.

Stop at some point: $\|w^{(k+1)} - w^{(k)}\|_2^2 < \varepsilon$ or $\|\mathcal{J}^{(k+1)} - \mathcal{J}^{(k)}\|_2^2 < \varepsilon$



a) Convex function



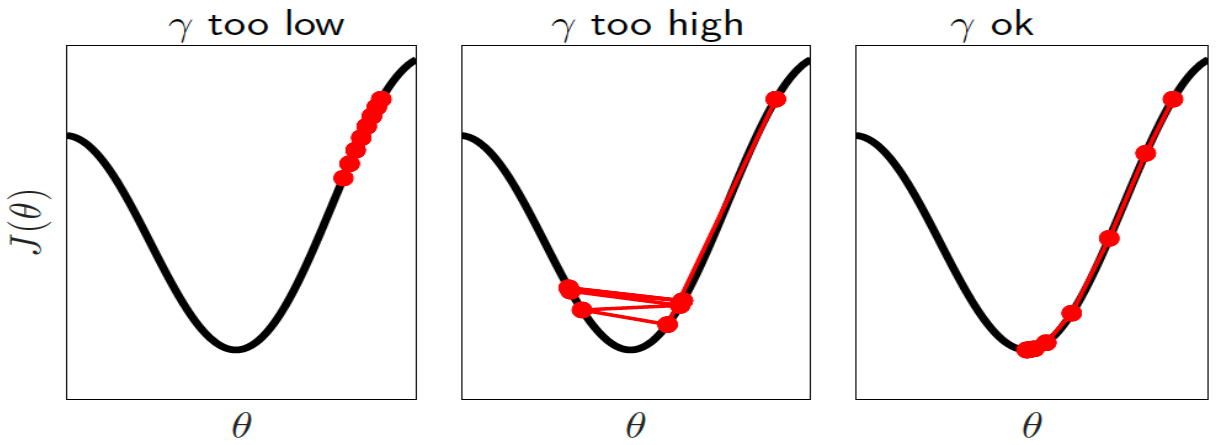
a) Non convex function

New notation

Cost function

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i, \theta), y_i) = \frac{1}{n} \sum_{i=1}^{n} L_i(\theta)$$

About Learning Rate



- if the error keeps getting worse or oscillates widely, reduce the learning rate
- if the error is fairly consistently but slowly increasing, increase the learning rate.

Gradient cost (1)

Each optimization update requires to compute the gradient

$$g_t \quad = \quad \nabla_\theta J(\theta_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta L(x_i, y_i, \theta_t).$$

- For neural network: $\dim(\theta)$ is big (a lot of parameters)
- Computing the gradient is costly.

Solution:

- NN is a composition of multiple layers.
- Each term $\nabla_\theta L(x_i, y_i, \theta)$ can be computed efficiently by repeatedly applying the chain rule.
- This is called the back-propagation algorithm.

Gradient cost (2)

Each optimization update requires to compute the gradient

$$g_t = \nabla_\theta J(\theta_t) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta L(x_i, y_i, \theta_t).$$

- For neural network: n is big (a lot of training data)
- Computing the gradient is costly.

Solution:

- For each iteration, we only use a small part of the data set to compute the gradient g_t.
- This is called the stochastic gradient descent.

Optimization for DL

To keep in mind in consideration:

- Non-convex (most of the times)
- Large-scale problem (large n and large dimension of θ)

Two main optimization procedures:

- Deterministic (batch)
- Stochastic (online)

Stochastic optimization

non-convex stochastic optimization problem:

$$\min_{\theta} J(\theta)$$

when only have access to noisy evaluations of $J(\theta)$ and its derivatives.

Stochastic optimization problems are common:

- When the cost function cannot be evaluated on the entire dataset.
- When numerical methods approximate $J(\theta)$ and $\nabla^i J(\theta)$

Intuition of using SG

A big data set is often redundant = many data points are similar.

If the training data is big:

$$\nabla_\theta J(\theta) \approx \frac{1}{n/2} \sum_{i=1}^{\frac{n}{2}} \nabla_\theta L(x_i, y_i, \theta)$$

$$\nabla_\theta J(\theta) \approx \frac{1}{n/2} \sum_{i=\frac{n}{2}+1}^{n} \nabla_\theta L(x_i, y_i, \theta).$$

We can do the update with only half the computation cost!

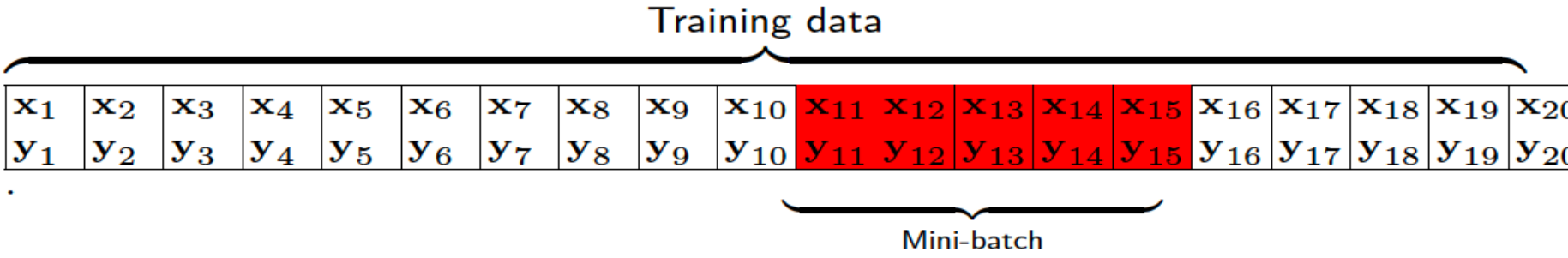
$$\theta_{k+1} = \theta_k - \gamma \frac{1}{n/2} \sum_{i=1}^{\frac{n}{2}} \nabla_\theta L(x_i, y_i, \theta_t),$$

$$\theta_{k+2} = \theta_{k+1} - \gamma \frac{1}{n/2} \sum_{i=\frac{n}{2}+1}^{n} \nabla_\theta L(x_i, y_i, \theta_{k+1}).$$

Minibatch SG

Stochastic gradient (SG) algorithm still the standard for large-scale machine learning and DL.

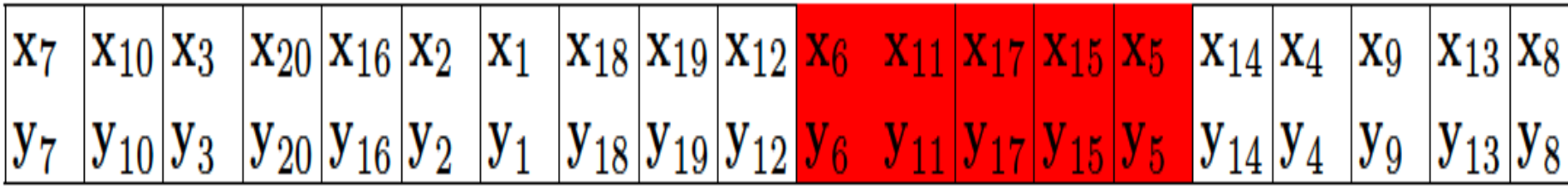
minibatch SG: combining the best properties of batch and stochastic algorithms



update step after 2 iterations:

$$\theta_3 = \theta_2 - \alpha \frac{1}{5} \sum_{i=11}^{15} \nabla_\theta L(x_i, y_i, \theta_2)$$

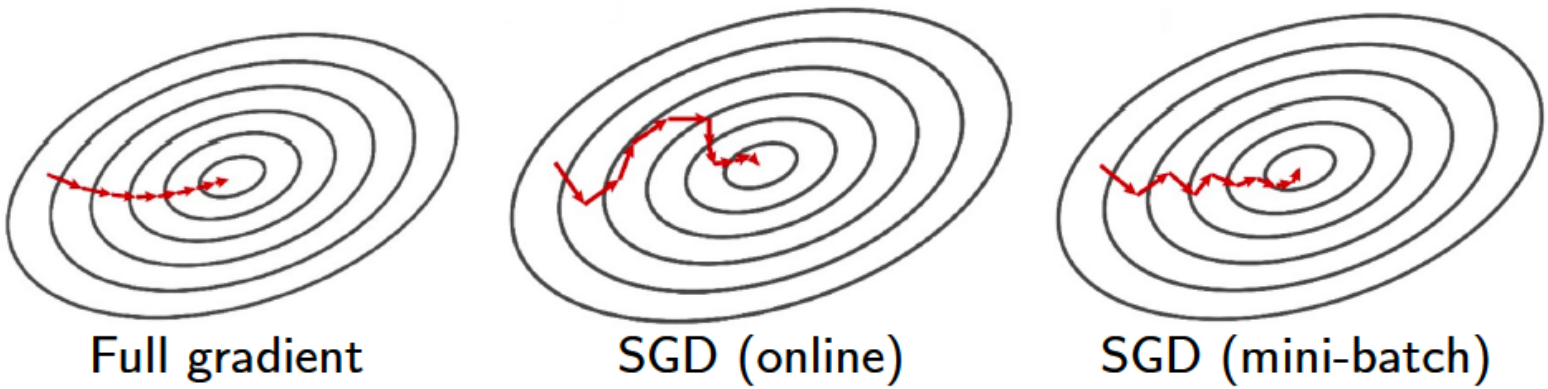
Minibatch SG



- If we pick the minibatches in order, they might be unbalanced and not representative for the whole data set.
- Therefore, we pick data points at random from the training data to form a minibatch.
- One implementation is to randomly reshuffle the data before dividing it into minibatches.
- After each epoch we do another reshuffling and another pass through the data set.

Minibatch SG

- The extreme version is to use only one data point at each training step (called online learning)
- One pass through the training data is called an epoch.
- minibatch version: a smaller set of samples between one data point and all data.



Minibatch Stochastic Gradient

The mini-batch gradient descent algorithm:

---

1. Initialize θ_0, set $k \leftarrow 1$, choose batch size n_b and number of epochs E.
2. For $i = 1$ to E
   (a) Randomly shuffle the training data $\{(x_i, y_i)\}_{i=1}^{n}$.
   (b) For $j = 1$ to $\frac{n}{n_b}$
      (i) Approximate the gradient of the loss function using the mini-batch $\{(x_i, y_i)\}_{i=(j-1)n_b+1}^{jn_b}$,
      $\widehat{g}_k = \frac{1}{n_b} \sum_{i=(j-1)n_b+1}^{jn_b} \nabla_\theta L_i(\theta)\Big|_{\theta=\theta_k} \approx g_k$.
      (ii) Do a gradient step $\theta_{k+1} = \theta_k - \gamma \widehat{g}_k$.
      (iii) Update the iteration index $k \leftarrow k + 1$.

At each time we get a stochastic approximation of the true gradient

Regularization

- **Regularization**: process to improve generalization, restrict the flexibility of the model to prevent overfitting
- **Example**: Ridge Regression
- **Principle**: add **Prior** $R(\theta)$ in training objective: $J(\theta) + \lambda R(\theta)$
- Gradient descent update to minimize $J(\theta)$: $\theta \leftarrow \theta - \alpha \frac{\partial J}{\partial \theta}$
- The gradient descent update to minimize the L^2 regularized cost $J(\theta) + \lambda R(\theta)$ results in weight decay:

$$\begin{aligned} \theta &\leftarrow \theta - \alpha \frac{\partial (J + \lambda R)}{\partial \theta} \\ &= \theta - \alpha \left(\frac{\partial J}{\partial \theta} + \lambda \frac{\partial R}{\partial \theta} \right) \\ &= \theta - \alpha \left(\frac{\partial J}{\partial \theta} + \lambda \theta \right) \\ &= (1 - \alpha\lambda)\theta - \alpha \frac{\partial J}{\partial \theta} \end{aligned}$$

Take Home of key concepts

- Convex set and convex function
- Gradient Descent
- Learning Rate
- Stochstic gradient (SG)
- Batch size and Epoch
- weight decay