Gradient Descent

Seminar 6

Kirill Alekseev

Machine Learning in Bioinformatics

October 14, 2024

Contents

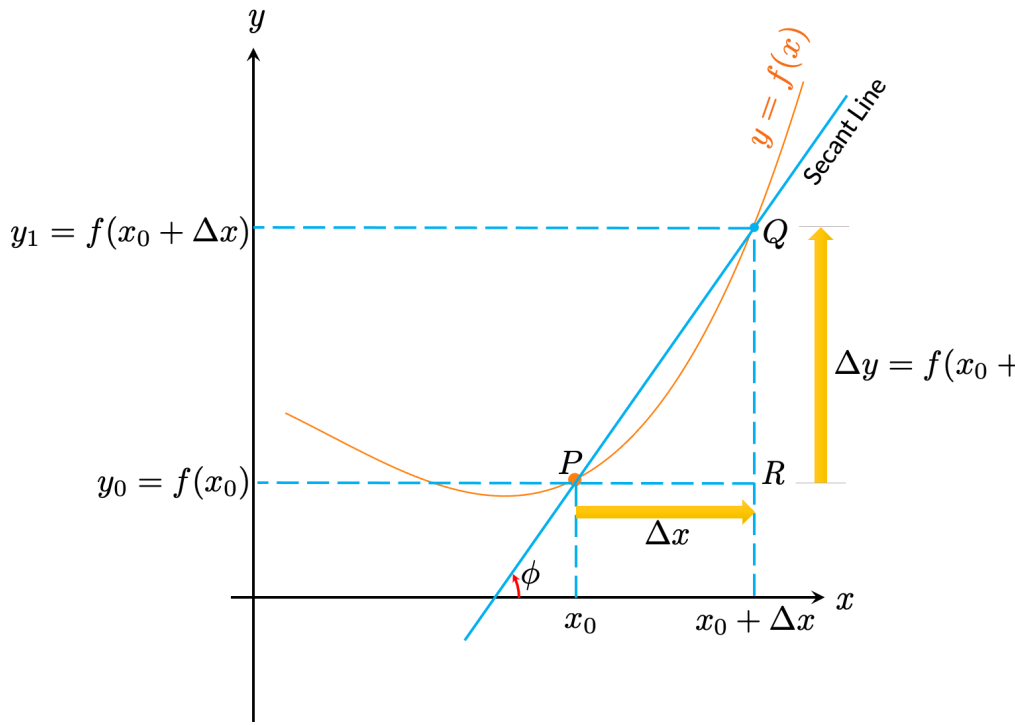
Gradient definition

Derivative definition

Derivative indicate the rate of change of a function with respect to that variable surrounding an infinitesimally small region near a particular point:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \frac{df}{dx}$$

Basically, the derivative is the slope of the tangent.



Derivative geometrical interpretation

Gradient definition

Gradient of a function $f : \mathbb{R}^d \to \mathbb{R}$: is defined as a vector of it's partial derivatives:

$$\nabla f(x_1, ..., x_d) = \left(\frac{\partial f}{\partial x_j} \right)_{j=1}^{d}$$

Gradient is the direction of steepest ascent of the function.
Likewise, antigradient $(-\nabla f)$ is the direction of steepest **descent**.

Gradient descent

Why use GD at all?

- We're working with OLS problem: $Xw = y$
- We're trying to solve with respect to w
- Solution: $w = (X^T X)^{-1} X^T y$
- What's wrong with this solution?

Why use GD at all?

- We're working with OLS problem: $Xw = y$
- We're trying to solve with respect to w
- Solution: $w = (X^T X)^{-1} X^T y$
- What's wrong with this solution?

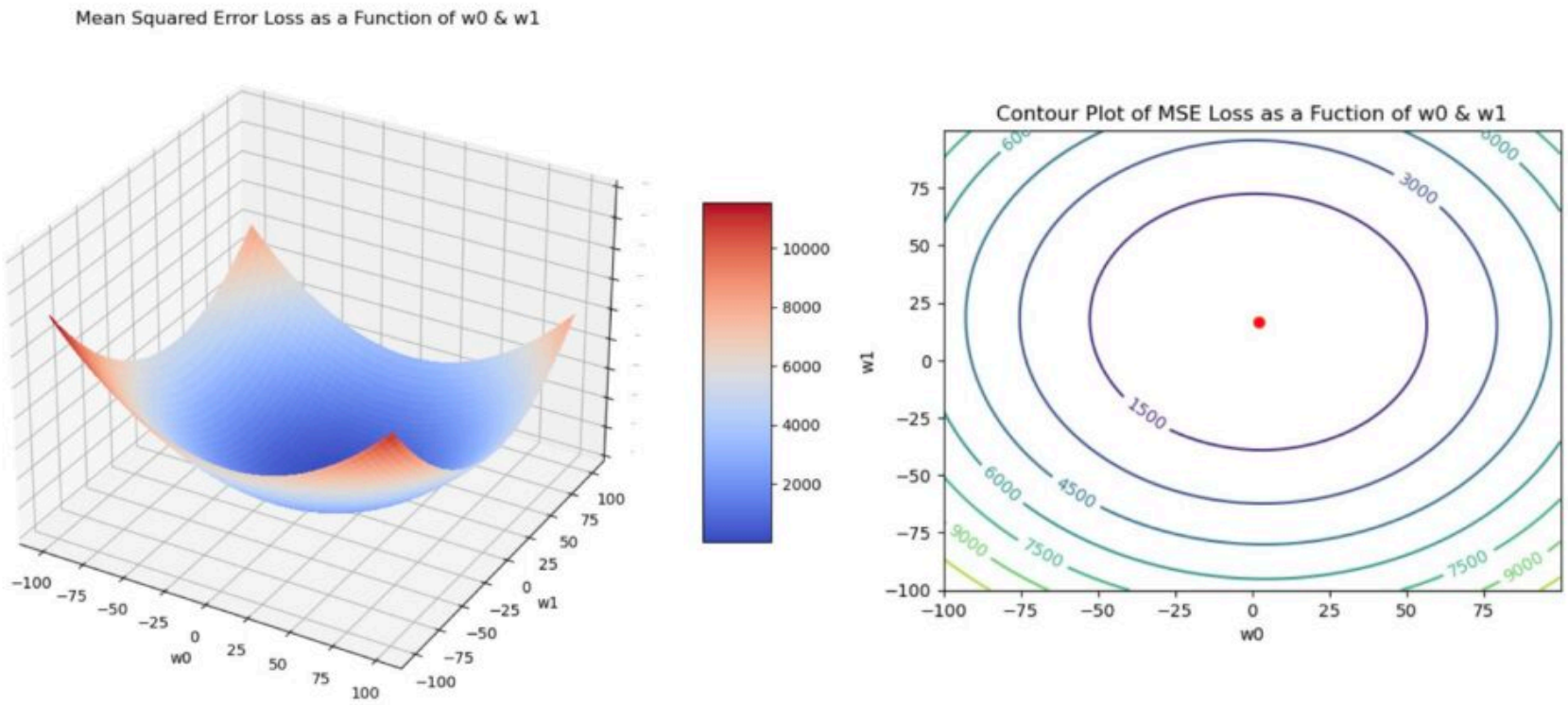
1. Computationally expensive $O(D^2 N + D^3)$.
2. While matrix $(X^T X)^{-1}$ can often be inverted, it has bad **condition number**.

Condition number of a function measures how much the output value of the function can change for a small change in the input argument.

In fact, condition number determines how wide would ellipsoids formed by loss function be.

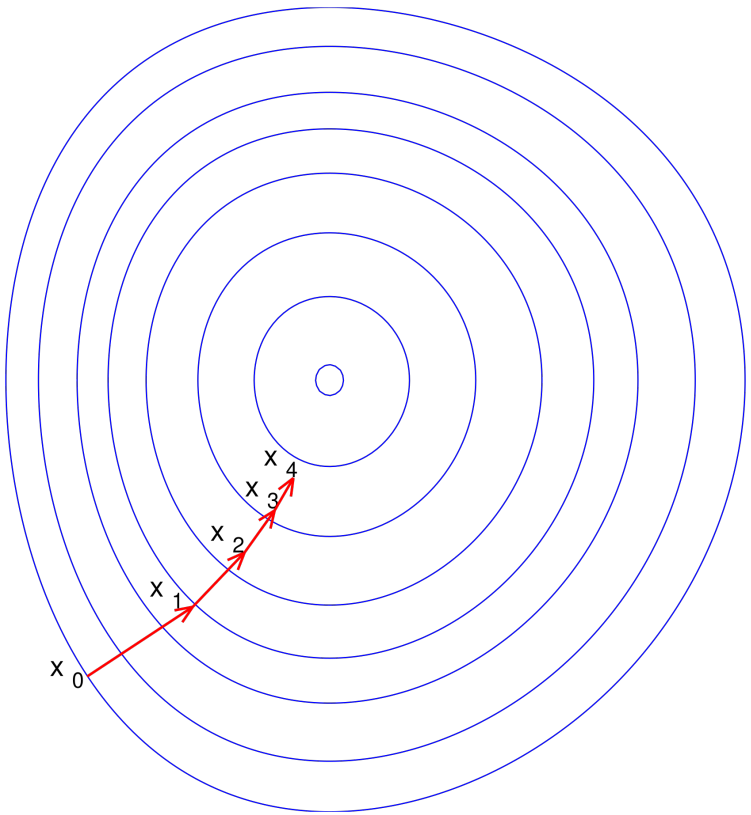
Gradient Descent

- MSE Loss as a function of weights is an ellipsoid with global minimum



Gradient Descent

- Idea: If our loss is differentiable, we can calculate gradients to shift our weights and hence minimize our loss.



Gradient Descent algorithm

- Start with an initial, random set of weights: w^0
- Given a differentiable loss function $\mathcal{L}$ (e.g. $\mathcal{L}_{SSE}$), compute $\nabla\mathcal{L}$
- For least squares:

$$\frac{\partial \mathcal{L}_{SSE}}{\partial w_i} = -2\sum_{n=1}^{N}(y_n - \vec{y_n})x_n$$

- In matrix form:

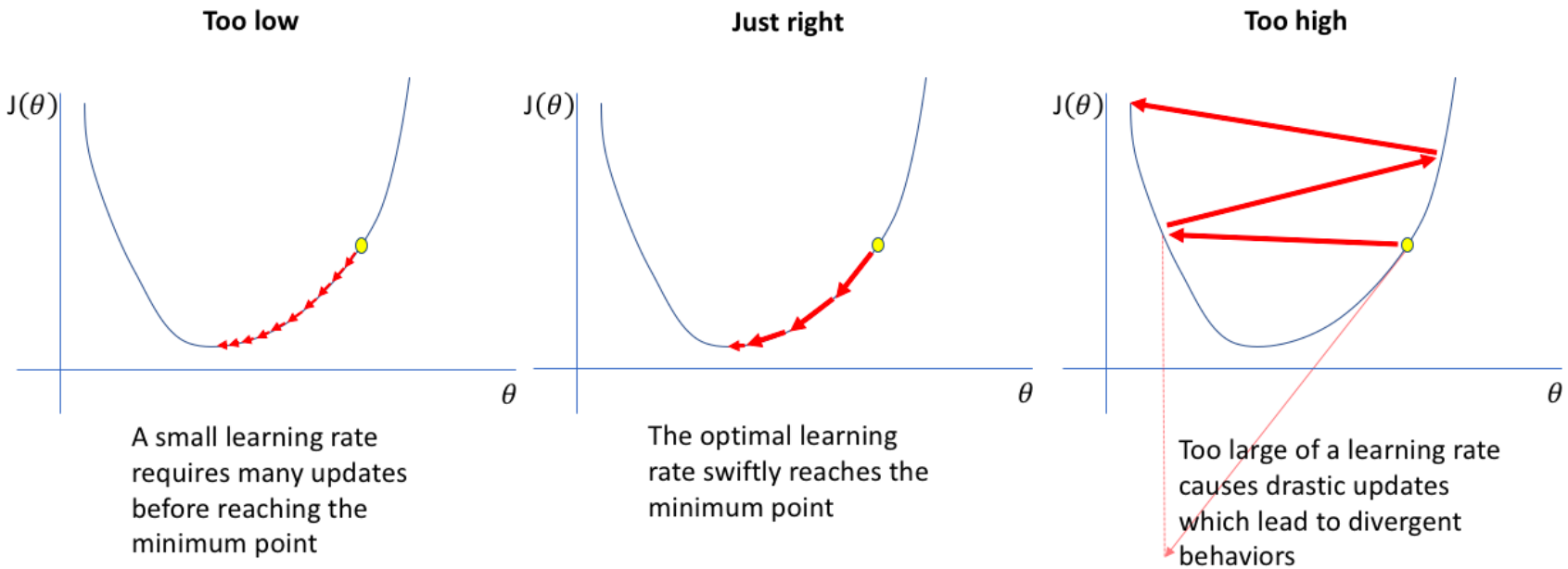
$$\nabla_w \mathcal{L} = \frac{2}{N}X^T(Xw - y)$$

- If feature $X_{:,i}$ is associated with big errors, the gradient wrt w_i will be large
- Update all weights slightly (by step size or learning rate) in "downhill" direction
- Basic update rule:

$$w_j^{t+1} = w_j^t - \eta\nabla\mathcal{L}(w_j^t)$$

Gradient Descent hyperparameters

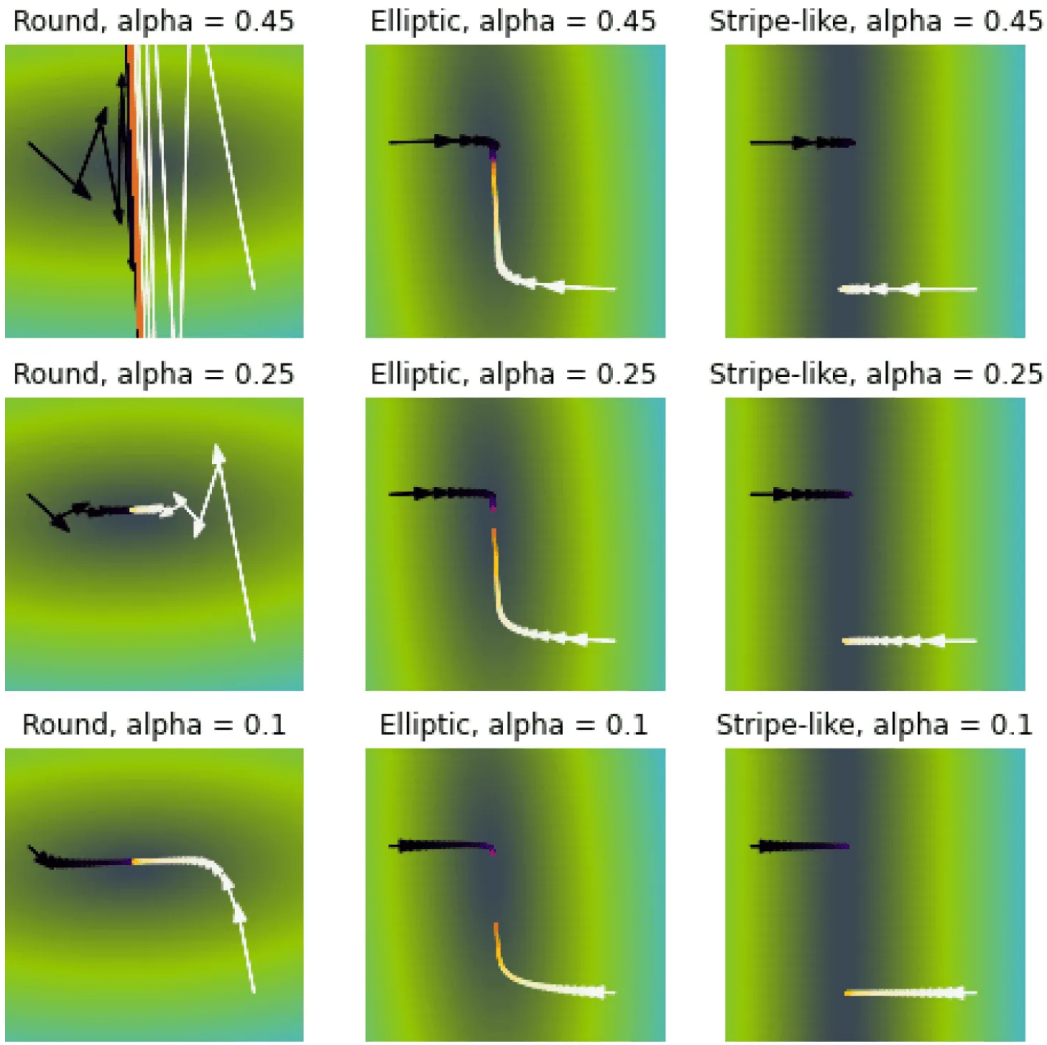
- Learning rate η
  - Too small: slow convergence; Too large: possible divergence
- Maximum number of iterations
  - Too small: no convergence; Too large: wasted resources
- Learning rate with decay rate
- Adaptive techniques: depend on how much loss improved in previous step



Gradient Descent convergence

- Convergence is guaranteed only when several assumptions are met, including loss function being convex and differentiable.
- You can still use GD with non-convex functions, but you may end up in a local minima or in a saddle point.
- Choice of starting point doesn't impact speed of convergence very much in case of convex functions.
- Bad condition number of matrix X impacts the ability of GD to converge greatly.

Gradient Descent convergence



*from Yandex ML Handbook

Gradient Descent overall

- There are other methods, but GD-based methods are the most popular ones
- For OLS problem, faster than analytical solution: $O(NDS)$, where N - length of sample, D - number of features, S - number of GD iterations
- Computationally expensive in case of big datasets, since we use all available data in each step. Hence, it is better to take some estimate of the full gradient, for example using SGD.

Stochastic Gradient descent

Idea behind SGD

- Insted of full gradient (average of gradients over all samples):

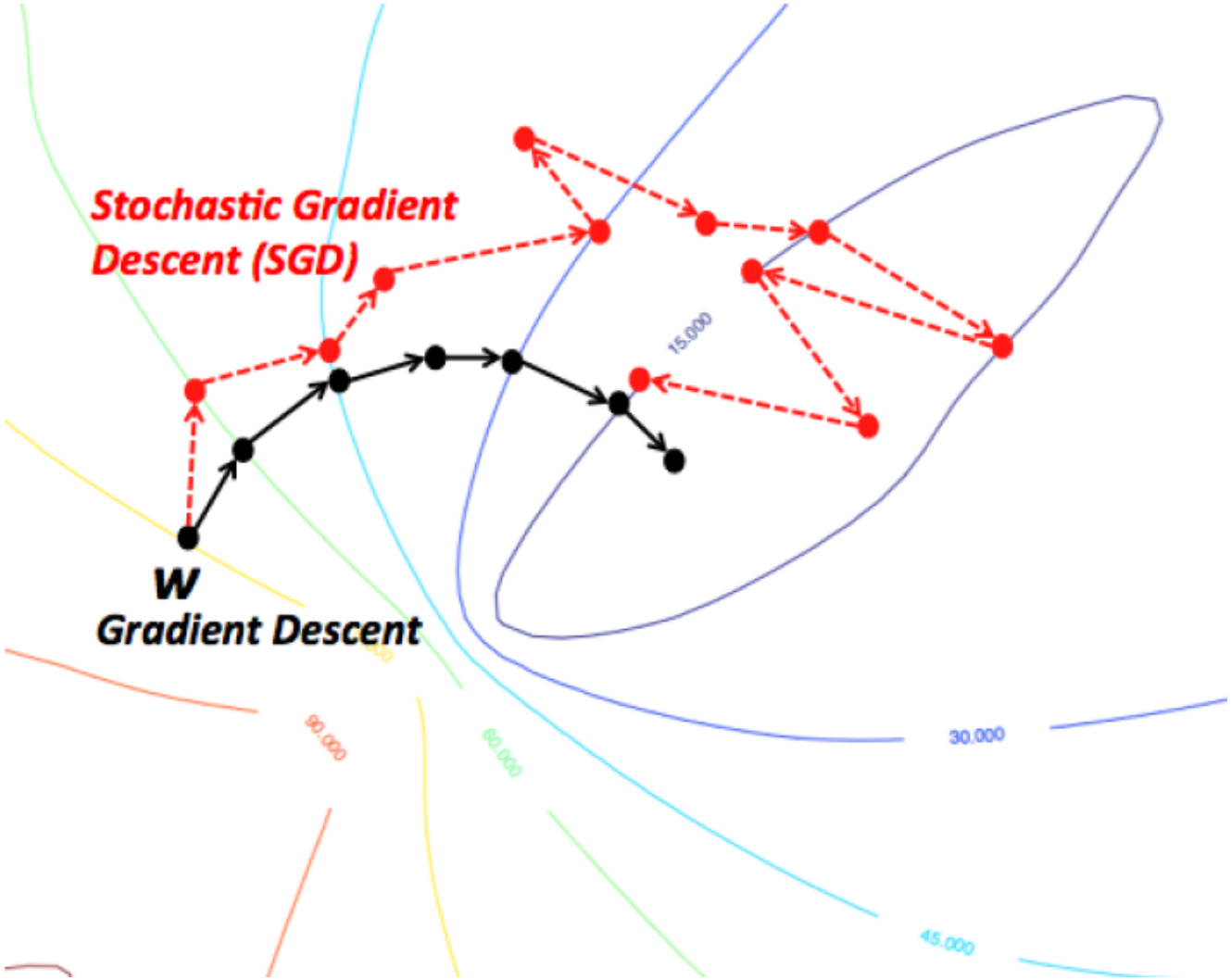
$$\nabla_w \mathcal{L}(w, X, y) = \frac{1}{N} \sum_{i=1}^{N} \nabla_w \mathcal{L}(w, x_i, y_i)$$

- Make estimate using subsample:

$$\nabla_w \mathcal{L}(w, X, y) \approx \frac{1}{B} \sum_{t=1}^{B} \nabla_w \mathcal{L}(w, x_{i_t}, y_{i_t})$$

- Generally steps are more noisy, as we will see later.
- New hyperparameter B - size of the batch.
- In practice (sklearn) - SGDRegressor (SGDClassifier) class.

SGD vs GD


Stochastic Gradient
Descent (SGD)
W
Gradient Descent
15,000
30,000
45,000
60,000
90,000

SGD variations

- Single point SGD - use only one point at each step. Sometimes this one is called SGD, and the one we described earlier is called batch gradient descent.
- SAG (Stochastic Average Gradient) - you can look it up here: https://fmin.xyz/docs/methods/fom/SAG.html

Gradient Descent and Regularization

- L^2 loss:

$$\mathcal{L}(f_w, X, y) = ||Xw - y||^2 + \lambda||w||^2$$

$$\nabla_w \mathcal{L}(f_w, X, y) = 2X^T(Xw - y) + 2\lambda w$$

- Which step is correct in SGD?
  1. $w_i \to w_i - 2\eta(< w, x_j > -y_j)x_{ji} - \frac{2\eta\lambda}{N}w_i, i = 1, ..., D$
  2. $w_i \to w_i - 2\eta(< w, x_j > -y_j)x_{ji} - 2\eta\lambda w_i, i = 1, ..., D$
  3. $w_i \to w_i - 2\eta(< w, x_j > -y_j)x_{ji} - 2\lambda N w_i, i = 1, ..., D$

Gradient Descent and Regularization

- Which step is correct in SGD?
  1. $w_i \to w_i - 2\eta(< w, x_j > -y_j)x_{ji} - \frac{2\eta\lambda}{N}w_i, i = 1, ..., D$
  2. $w_i \to w_i - 2\eta(< w, x_j > -y_j)x_{ji} - 2\eta\lambda w_i, i = 1, ..., D$
  3. $w_i \to w_i - 2\eta(< w, x_j > -y_j)x_{ji} - 2\lambda N w_i, i = 1, ..., D$
-

$$\mathcal{L}(X, y, w) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(x_i, y_i, w)$$

$$\mathcal{L}(w) = \mathbb{E}_{x,y}\mathcal{L}(x, y, w)$$

$$\mathcal{L}_{reg}(w) = \mathbb{E}_{x,y}\mathcal{L}(x, y, w) + \lambda||w||^2$$

$$\nabla_w\mathcal{L}_{reg}(w) = \mathbb{E}_{x,y}\nabla_w\mathcal{L}(x, y, w) + 2\lambda w$$

$$\nabla_w\mathcal{L}_{reg}(w) = \frac{1}{B}\sum_{i=1}^{B}\nabla_w\mathcal{L}(x_{t_i}, y_{t_i}, w) + 2\lambda w$$

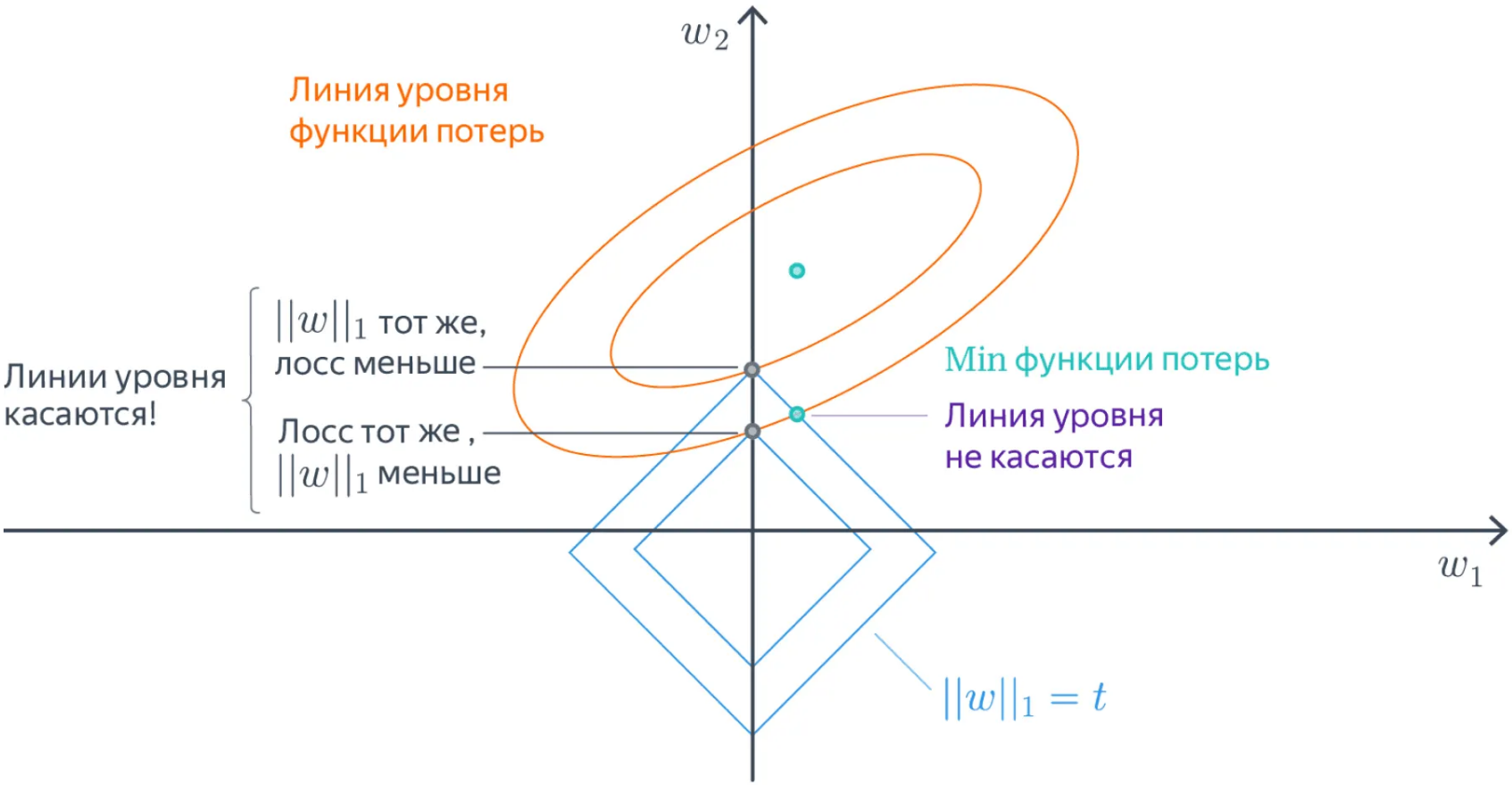
Gradient Descent and Regularization

- How does the step for SGD looks like with L^1 norm? Why are not concerned about loss not being differentiable?

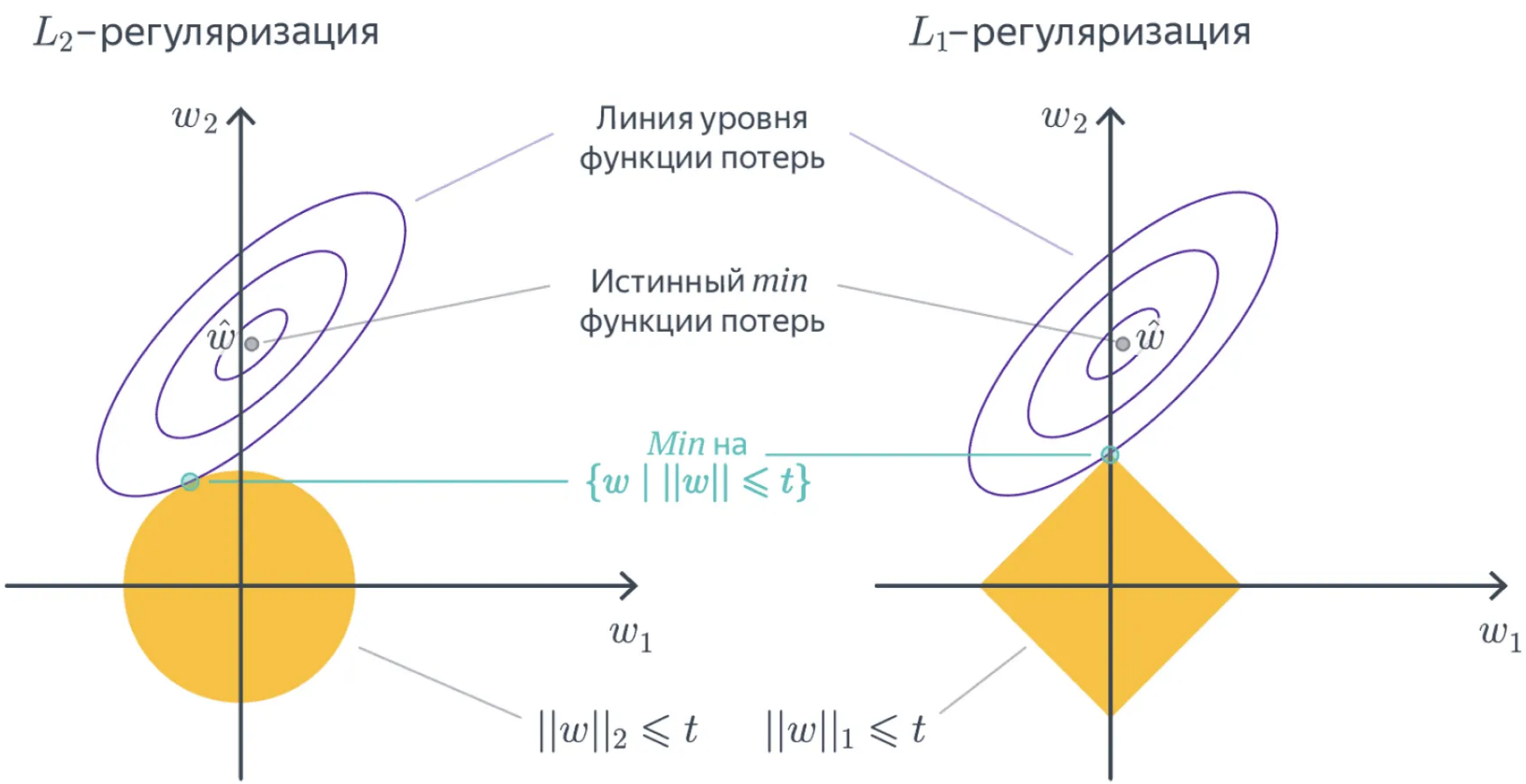
Gradient Descent and Regularization

- How does the step for SGD looks like with L^1 norm? Why we are not concerned about loss not being differentiable?
- $w_i \to w_i - \eta(< w, x_j > - y_j)x_{ji} - \frac{\lambda}{\eta} sign(w_i)w_i, i = 1, ..., D$
- It's very unlikely we hit exactly one specific point. For this case, we can manually redefine the value of derivative in this point.

Few words on Regularization



Few words on Regularization



Practice