Gradient Descent Solutions to Least-Square Problems

Objectives

- explain need for iterative algorithms
- derive gradient descent algorithm
- consider impact of step size on convergence
- introduce notion of convex functions

Iterative solution methods play an important role

Features/labels: $\underline{x}_i, d_i, i = 1, 2, \ldots N$ (N samples)

Classifier or model error: $e^2 = \sum_{i=1}^{N} (\underline{x}_i^T \underline{w} - d_i)^2$

$$\underline{A} = \begin{bmatrix} \underline{x}_1^T \\ \underline{x}_2^T \\ \vdots \\ \underline{x}_N^T \end{bmatrix} \quad \underline{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \quad e^2 = \|\underline{A}\underline{w} - \underline{d}\|_2^2$$

Regularized least squares: $\arg\min_{\underline{w}} \|\underline{A}\underline{w} - \underline{d}\|_2^2 + \lambda r(\underline{w})$

(In general, we use this regularized LS solution; $r(\underline{w})$: general regularizer)

1. Computational cost $(\underline{A}^T\underline{A})^{-1}$
2. Closed form solution may be unavailable
3. Adapt $\underline{w}$ to new features/labels

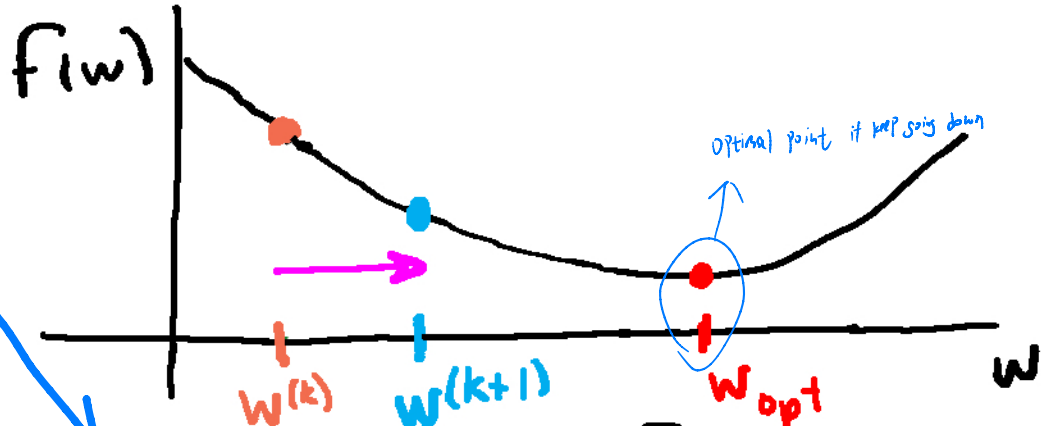
} The reason for develop iterative approach

Gradient descent finds the minimum

$$f(\underline{w}) = \|\underline{A}\underline{w} - \underline{d}\|_2^2$$

$$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \tau' \nabla_{\underline{w}} f(\underline{w}) \quad (\tau' > 0)$$

- $\underline{w}^{(k+1)}$: New weight; $(k+1)$: $k+1$th iteration
- $\underline{w}^{(k)}$: Previous weight; (k): the previous w at kth iteration
- τ': step size
- $\nabla_{\underline{w}} f(\underline{w})$: gradient



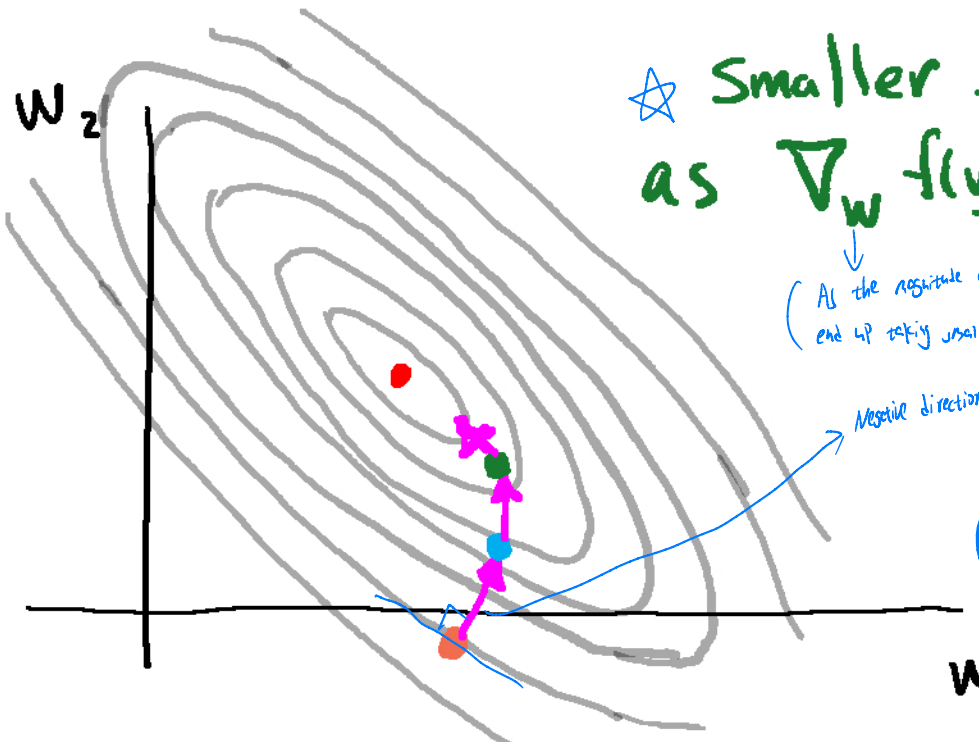
$$f(\underline{w}) = (\underline{A}\underline{w} - \underline{d})^T(\underline{A}\underline{w} - \underline{d})$$

($f(\underline{w})$: cost function)

$$= \underline{w}^T\underline{A}^T\underline{A}\underline{w} - 2\underline{w}^T\underline{A}^T\underline{d} + \underline{d}^T\underline{d}$$

$$\nabla_{\underline{w}} f(\underline{w}) = 2\underline{A}^T\underline{A}\underline{w} - 2\underline{A}^T\underline{d}$$

$$= 2\underline{A}^T(\underline{A}\underline{w} - \underline{d})$$



✡ Smaller steps as $\nabla_{\underline{w}} f(\underline{w})$ ↓

(As the magnitude of gradient decreases, we end up taking smaller steps since $\tau' \nabla_{\underline{w}} f(\underline{w})$)

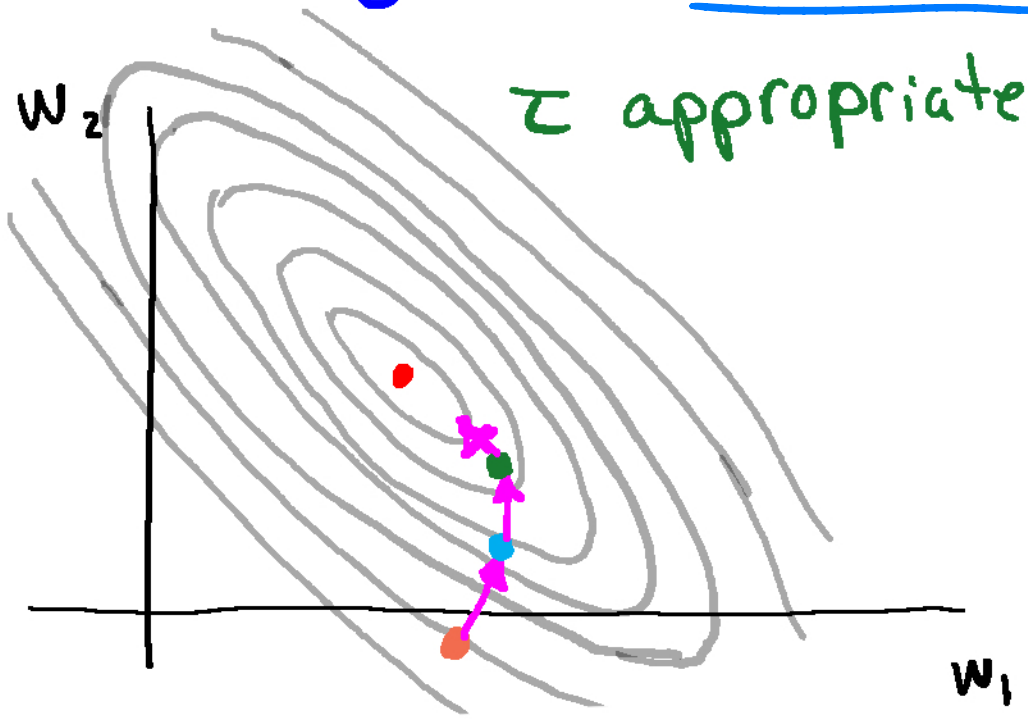
Negative direction to minimize the $f(\underline{w})$

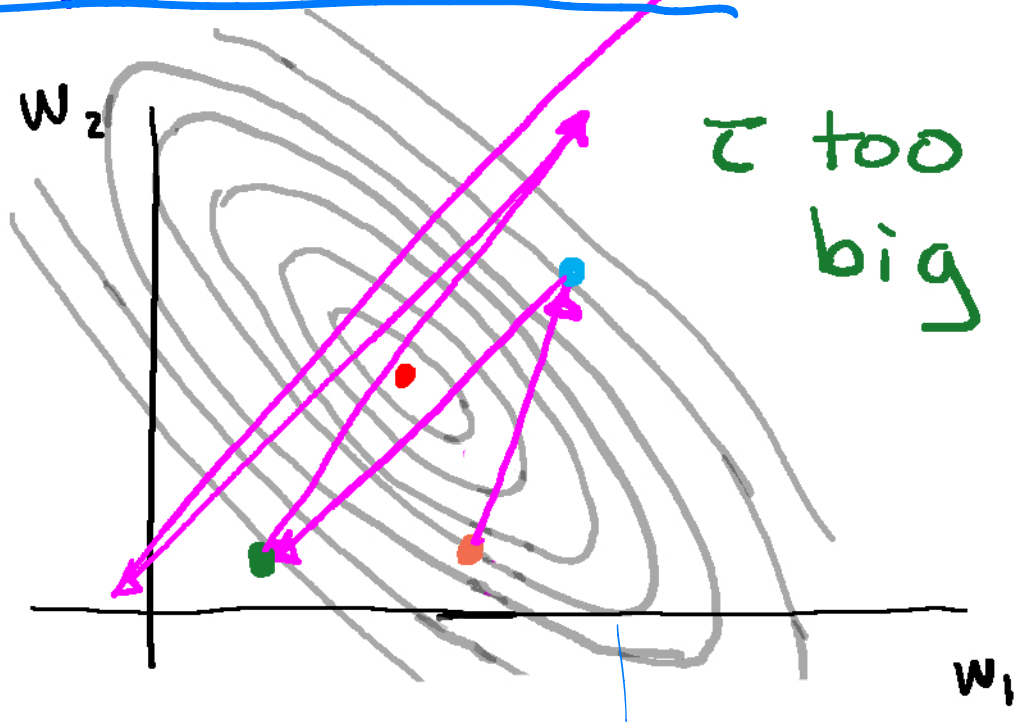
✡ $$\underline{w}^{(k+1)} = \underline{w}^{(k)} - \tau \underline{A}^T(\underline{A}\underline{w}^{(k)} - \underline{d})$$

(τ is absorbed within τ)

This is called (Landweber iteration)

Convergence behavior depends on τ





- τ too small: slow convergence
- τ too big: no convergence unstable!

Require $0 < \tau < 2/\|\underline{A}\|_{op}^2$ for convergence

(Put bounds at step size tau)

Recall $\|\underline{A}\|_{op} = \|\underline{A}\|_2 = \sigma_{max}(\underline{A})$

(Largest singular value of A σ_1)

☆ (Note: For matrix A^TA: A的最大singular value $\sigma_1 = \sqrt{\lambda_{max}}$ = A^TA 的最大 eigenvalue 的 square root)

Convergence: $f(\underline{w}^{(k+1)}) < f(w^{(k)})$ — cost decreases as k increases

$$\|\underline{A}\,\underline{w}^{(k+1)} - d\|_2^2 < \|\underline{A}\,\underline{w}^{(k)} - d\|_2^2$$

(Ensures you are actually going downhill: (k+1) round's cost < k round's cost)

(Definition of convergence, the cost needs to be smaller each iteration. If it goes up it does no good)

Notes – guaranteed convergence for

$$0 < \tau < 2/\|\underline{A}\|_{op}^2$$

$$\underline{w}^{(0)} = \underline{0}, \quad \underline{w}^{(k+1)} = \underline{w}^{(k)} - \tau \underline{A}^T(\underline{A}\,\underline{w}^{(k)} - \underline{d}) \xrightarrow{k} (\underline{A}^T\underline{A})^{-1}\underline{A}^T\underline{d}$$

(Let's initialize the weights to be 0 at first for convenience)

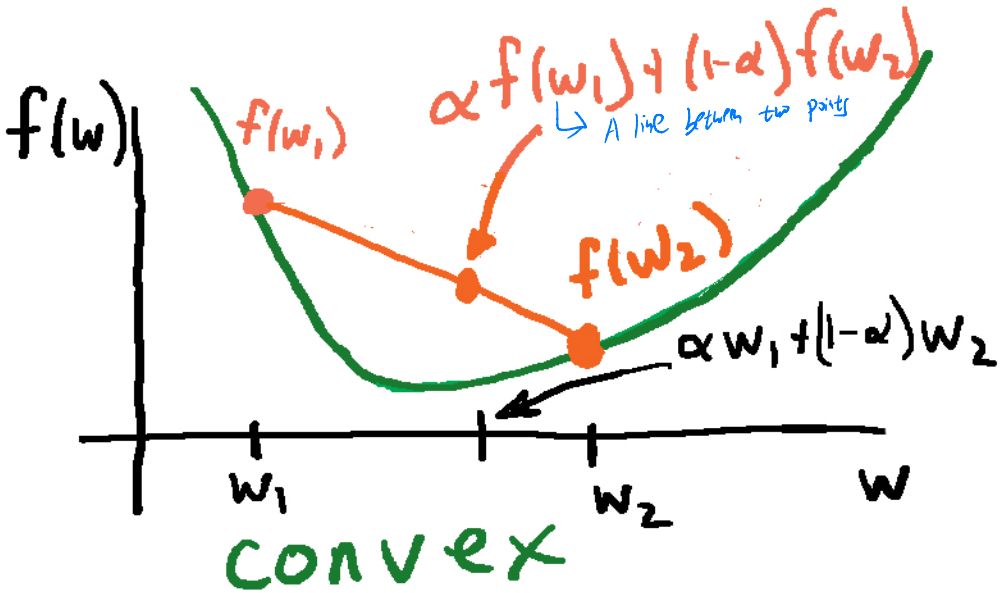
The iteration algorithm

After k rounds

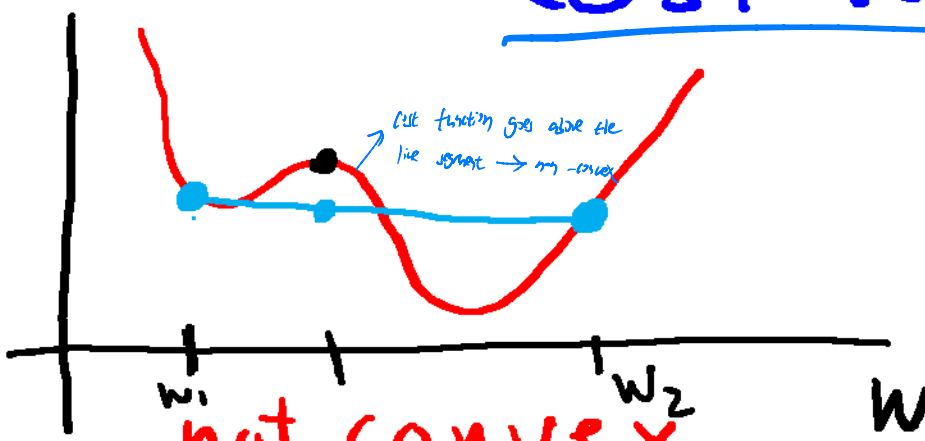
(Converge to the least squares solution)

Gradient descent is effective for convex cost functions

(☆ If non-convex, GD will not guaranteed to find the global minimum)



convex



not convex

local minima

(斜率一直在增加)

① Line segment test

$$f(\alpha w_1 + (1-\alpha) w_2) \leq \alpha f(w_1) + (1-\alpha) f(w_2);$$

(The cost function needs to be below the line segment for any two points → convex)

$$0 < \alpha < 1, \text{ all } w_1, w_2$$

② Second derivative test → (if function also does not have inflection points, then convex)

$$\frac{d^2}{dw^2} f(w) \geq 0$$

Multidimensional case → Every w from any direction needs to satisfy the second derivative test ≥0 for convexity

$$\left[\underline{H}(\underline{w})\right]_{i,j} = \frac{\partial^2}{\partial w_i \partial w_j} f(\underline{w})$$

Here $\underline{w} = [w_1, w_2, \ldots]$

Hessian matrix

☆ (second derivative respect to f(w) cost function in various coordinate directions)

☆ Quick way for testing $H \geq 0$:

H is symmetric (known)

↓

PSD: For all eigenvalue $\lambda \geq 0$

$$\underline{H}(\underline{w}) \geq 0$$

→ Needs to be semi-definite

Gradient Descent for Solving Least-Squares Problems Proof: Bounds on Step Size for Guaranteed Convergence

Gradient descent minimizes the cost function

$$f(\boldsymbol{w}) = ||\boldsymbol{A}\boldsymbol{w} - \boldsymbol{d}||_2^2$$

using the iterative algorithm

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \tau \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}), \quad k = 0, 1, 2, 3, \ldots \tag{1}$$

where $\tau > 0$ so that we modify the current iterate in the negative gradient direction. Often this algorithm is initialized with $\boldsymbol{w}^{(0)} = \boldsymbol{0}$. The initialization does not affect the convergence behavior because $f(\boldsymbol{w})$ is convex.

The iteration is guaranteed to converge to the minimum of the cost function if the squared error decreases with each iteration, that is, if

$$f(\boldsymbol{w}^{(k+1)}) = ||\boldsymbol{A}\boldsymbol{w}^{(k+1)} - \boldsymbol{d}||_2^2 < f(\boldsymbol{w}^{(k)}) = ||\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}||_2^2$$

Begin by substituting $\boldsymbol{w}^{(k)} - \tau \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})$ for $\boldsymbol{w}^{(k+1)}$ in $f(\boldsymbol{w}^{(k+1)})$ to write

$$\begin{aligned} f(\boldsymbol{w}^{(k+1)}) &= ||\boldsymbol{A}\left(\boldsymbol{w}^{(k)} - \tau \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})\right) - \boldsymbol{d}||_2^2 &(2)\\ &= ||(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}) - \tau\left(\boldsymbol{A}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})\right)||_2^2 &(3)\end{aligned}$$

Now let $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}$ and $\boldsymbol{e} = \tau\left(\boldsymbol{A}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})\right)$ be the first and second terms in parentheses so $f(\boldsymbol{w}^{(k+1)}) = ||\boldsymbol{c} - \boldsymbol{e}||_2^2 = (\boldsymbol{c} - \boldsymbol{e})^T(\boldsymbol{c} - \boldsymbol{e})$. Expand the product to write $f(\boldsymbol{w}^{(k+1)}) = ||\boldsymbol{c}||_2^2 + ||\boldsymbol{e}||_2^2 - 2\boldsymbol{e}^T\boldsymbol{c}$. Substituting for $\boldsymbol{c}$ and $\boldsymbol{e}$ we thus obtain

$$f(\boldsymbol{w}^{(k+1)}) = ||\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}||_2^2 + \tau^2||\boldsymbol{A}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})||_2^2 - 2\tau\left((\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})^T\boldsymbol{A}\boldsymbol{A}^T\right)(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}) \tag{4}$$

$$= f(\boldsymbol{w}^{(k)}) + \tau^2||\boldsymbol{A}\left(\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})\right)||_2^2 - 2\tau\left((\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})^T\boldsymbol{A}\right)\left(\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})\right) \tag{5}$$

Define $\boldsymbol{v} = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d})$ to simplify the expression and rewrite Eq. 5 as

$$f(\boldsymbol{w}^{(k+1)}) = f(\boldsymbol{w}^{(k)}) + \tau^2||\boldsymbol{A}\boldsymbol{v}||_2^2 - 2\tau\boldsymbol{v}^T\boldsymbol{v}$$

Note that $\boldsymbol{v}$ does not depend on τ. Thus, to prove $f(\boldsymbol{w}^{(k+1)}) < f(\boldsymbol{w}^{(k)})$, we must find the condition for which

$$q(\tau) = \tau^2||\boldsymbol{A}\boldsymbol{v}||_2^2 - 2\tau\boldsymbol{v}^T\boldsymbol{v}$$

is less than zero.

Recall the operator norm of a matrix $\boldsymbol{X}$ satisfies $\max_{\boldsymbol{g}} ||\boldsymbol{X}\boldsymbol{g}||_2 \leq ||\boldsymbol{X}||_{op}||\boldsymbol{g}||_2$, so the first term in $q(\tau)$ may be upper bounded as

$$\tau^2||\boldsymbol{A}\boldsymbol{v}||_2^2 \leq \tau^2||\boldsymbol{A}||_{op}^2||\boldsymbol{v}||_2^2$$

We may rewrite the second term in $q(\tau)$ as

$$-2\tau\boldsymbol{v}^T\boldsymbol{v} = -2\tau||\boldsymbol{v}||_2^2$$

Hence, we obtain an upper bound on $q(\tau)$

$$q(\tau) \leq \tau^2||\boldsymbol{A}||_{op}^2||\boldsymbol{v}||_2^2 - 2\tau||\boldsymbol{v}||_2^2$$

Factoring out the common terms we write

$$q(\tau) \leq \left(\tau||\boldsymbol{A}||_{op}^2 - 2\right)\tau||\boldsymbol{v}||_2^2$$

The second term in $q(\tau)$, $\tau||\boldsymbol{v}||_2^2$, is positive provided $\boldsymbol{v} \neq \boldsymbol{0}$, so we obtain $q(\tau) < 0$ by requiring

$$\left(\tau||\boldsymbol{A}||_{op}^2 - 2\right) < 0$$

which indicates τ must satisfy

$$\tau < \frac{2}{||\boldsymbol{A}||_{op}^2}$$

Note that $\boldsymbol{v} = \boldsymbol{0}$ implies $\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{w}^{(k)} - \boldsymbol{d}) = \boldsymbol{0}$, or $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{w}^{(k)} = \boldsymbol{A}^T\boldsymbol{d}$, or $\boldsymbol{w}^{(k)} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{d}$. Thus, if $\boldsymbol{v} = \boldsymbol{0}$, then the iteration has converged to the minimum of the squared error and the update term in Eq. 1 is zero.

Hence, the gradient descent algorithm will converge to the minimum of the squared error cost function provided the step-size τ satisfies $\tau < \frac{2}{||\boldsymbol{A}||_{op}^2}$.