E9 211: Adaptive Signal Processing

Stochastic Gradient Descent

Outline

1. Least mean squares (LMS) algorithm
2. Comuptational complexity
3. Stability condition
4. Normalized LMS algorithm
5. Connection to Kaczmarz

Least mean squares algorithm

- Consider the problem of finding the optimal beamformer for linear least mean square estimation.
- We have seen that the optimal beamformer can be obtained using steepest gradient descent (SGD) iterations of the form

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mu[\mathbf{R}_x \mathbf{w}^{(k)} - \mathbf{r}_{xs}],$$

  where the step size μ is appropriately selected to ensure convergence.
- However, true value of $\mathbf{R}_x$ and $\mathbf{r}_{xy}$ are not available in practice and needs to be estimated from available data.

Least mean squares algorithm

- The vector valued observations corresponding to different time instants are given by

$$\mathbf{x}_k = \mathbf{a}s_k + \mathbf{n}_k, \quad k = 1, 2, \ldots,$$

where $\mathbf{x}_k \in \mathbb{C}^M$, $\mathbf{a} \in \mathbb{C}^M$, $\mathbf{n}_k \in \mathbb{C}^M$, and $s_k \in \mathbb{C}$.

- In practice, we compute estimates of $\mathbf{R}_x$ and $\mathbf{r}_{xs}$ as

$$\hat{\mathbf{R}}_x = \frac{1}{N}\sum_{k=1}^{N} \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}, \quad \hat{\mathbf{r}}_{xs} = \frac{1}{N}\sum_{k=1}^{N} \mathbf{x}_k s_k^*.$$

- In the SGD update equation, we replace true gradient (computed using $\mathbf{R}_x$ and $\mathbf{r}_{xs}$) with a *noisy* version (computed using $\hat{\mathbf{R}}_x$ and $\hat{\mathbf{r}}_{xs}$) to get

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mu[\hat{\mathbf{R}}_x \mathbf{w}^{(k)} - \hat{\mathbf{r}}_{xs}],$$

- This is known as the stochastic gradient descent algorithm.

Least mean squares algorithm

- Consider a special case of the stochastic gradient descent algorithm with $N = 1$.
- We replace $\mathbf{R}_x$ and $\mathbf{r}_{xs}$ using the instantaneous estimates

$$\hat{\mathbf{R}}_x = \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}, \quad \hat{\mathbf{r}}_{xs} = \mathbf{x}_k s_k^*.$$

- The gradient at the k^{th} iteration is approximated as

$$\nabla J(\mathbf{w}^{(k)}) = \mathbf{R}_x \mathbf{w}^{(k)} - \mathbf{r}_{xs} \approx \nabla \hat{J}(\mathbf{w}^{(k)}) = \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} \mathbf{w}^{(k)} - \mathbf{x}_k s_k^*.$$

- This leads to the so-called least mean squares (LMS) algorithm. [Widrow, 1975][1]

[1] B. Widrow, J. McCool and M. Ball, "The complex LMS algorithm," in Proceedings of the IEEE, vol. 63, no. 4, pp. 719-720, April 1975, doi: 10.1109/PROC.1975.9807.

Least mean squares algorithm

- The LMS update equations are given by

$$\begin{aligned}\mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \mu \mathbf{x}_k e_k^* \\ e_k^* &= \mathbf{x}_k^{\mathrm{H}} \mathbf{w}^{(k)} - s_k^*.\end{aligned}$$

- Time index and iteration index are same for the LMS algorithm.
- The update directions are subject to random fluctuations (or gradient noise). The LMS will never converge exactly, i.e., LMS will respond to a new sample.

Computational complexity

- Each complex addition ($C+$) involves 2 real additions ($R+$).

$$(a + jb) + (c + jd) = (a + c) + j(c + d)$$

- Each complex multiplication ($C\times$) involves 4 real multiplications ($R\times$) and 2 real additions ($R+$).

$$(a + jb) \times (c + jd) = ((a \times c) - (b \times d)) + j((a \times d) + (b \times c))$$

Computational complexity

- Each iteration of the LMS algorithm involves 5 steps.

- Step - 1 $[\mathbf{x}_k^{\mathrm{H}}\mathbf{w}^{(k)}]$
  - Inner product between two M dimensional complex vectors.
  - Involves $M\ C\times$ and $(M-1)\ C+$.
  - Which involves $4M\ R\times$ and $2M+2(M-1)=(4M-2)\ R+$.

- Step - 2 $[e_k^* = \mathbf{x}_k^{\mathrm{H}}\mathbf{w}^{(k)} - s_k^*]$
  - Complex addition.
  - Involves $2\ R+$.

Computational complexity

- Step - 3 $[\mu e_k^*]$
  - Multiplication of a real number with a complex number.
  - Involves $2\ R\times$
- Step - 4 $[\mathbf{x}_k \mu e_k^*]$
  - Multiplication of a complex scalar with a M dimensional complex vector.
  - Involves $M\ C\times \implies 4M\ R\times$ and $2M\ R+$.
- Step - 5 $[\mathbf{w}_k - \mathbf{x}_k \mu e_k^*]$
  - Addition of two M dimensional complex vectors.
  - Involves $M\ C+ \implies 2M\ R+$.

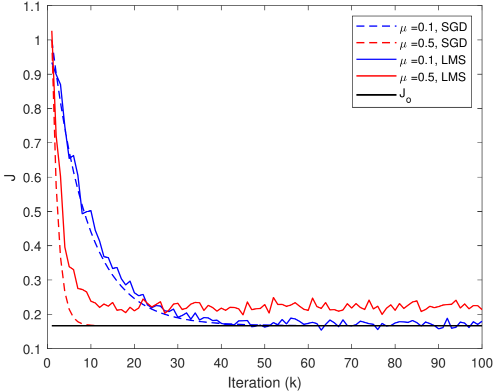
Computational complexity

Operation	Real multiplications	Real additions
$\mathbf{x}_k^{\mathrm{H}}\mathbf{w}^{(k)}$	4M	4M-2
$e_k^* = \mathbf{x}_k^{\mathrm{H}}\mathbf{w}^{(k)} - s_k^*$	-	2
μe_k^*	2	-
$\mathbf{x}_k \mu e_k^*$	4M	2M
$\mathbf{w}_k - \mathbf{x}_k \mu e_k^*$	-	2M
Total	8M + 2	8M

- One iteration of LMS involves (8M + 2) real multiplications and 8M real additions.
- Similarly, it can be seen that for real data (i.e., $\mathbf{x}_k, \mathbf{a}, \mathbf{n}_k \in \mathbb{R}^M$ and $s_k \in \mathbb{R}$), each update of LMS involves 2M real additions and (2M+1) real multiplications.
- Complexity of LMS is linear in M.

Convergence

- Step size μ needs to be selected to ensure convergence (we will soon derive the conditions).



- Solid lines correspond to LMS and dashed line corresponds to SGD.

- SGD converge monotonically. LMS fluctuates.

Stability

- Let us define the weight error vector $\mathbf{c}^{(k)} = \mathbf{w}^{(k)} - \mathbf{w}_0$. Then we have

$$\begin{aligned} \mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \mu(\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} \mathbf{w}_k - \mathbf{x}_k s_k^*) \\ \mathbf{w}_0 &= \mathbf{w}_0 - \mu(\mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}]\mathbf{w}_0 - \mathbb{E}[\mathbf{x}_k s_k^*]) \\ \hline \mathbf{c}^{(k+1)} &= \mathbf{c}^{(k)} - \mu[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} \mathbf{w}_k - \mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}]\mathbf{w}_0 - (\mathbf{x}_k s_k^* - \mathbb{E}[\mathbf{x}_k s_k^*])] \end{aligned}$$

- Since the LMS update equation is stochastic in nature, we describe the convergence of the LMS algorithm in the mean.

- To do that, we will consider the mean of the weight error vector ($\mathbb{E}[\mathbf{c}^{(k)}]$) for analysing the convergence.

Stability

- Let us assume that $\mathbf{x}_k$ is independent of $\mathbf{w}^{(k)}$. Then we have

$$\mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} \mathbf{w}^{(k)}] = \mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}]\mathbb{E}[\mathbf{w}^{(k)}]$$

- Mean of the error vector can be computed as

$$\begin{aligned}
\mathbb{E}[\mathbf{c}^{(k+1)}] &= \mathbb{E}[\mathbf{c}^{(k)} - \mu[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} \mathbf{w}_k - \mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}]\mathbf{w}_0 - (\mathbf{x}_k s_k^* - \mathbb{E}[\mathbf{x}_k s_k^*])]] \\
&= \mathbb{E}[\mathbf{c}^{(k)}] - \mu[\mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}](\mathbb{E}[\mathbf{w}^{(k)} - \mathbf{w}_0]) - (\mathbb{E}[\mathbf{x}_k s_k^*] - \mathbb{E}[\mathbb{E}[\mathbf{x}_k s_k^*]])] \\
&= \mathbb{E}[\mathbf{c}^{(k)}] - \mu \mathbf{R}_x \mathbb{E}[\mathbf{c}^{(k)}] - 0 \\
&= (\mathbf{I} - \mu \mathbf{R}_x)\mathbb{E}[\mathbf{c}^{(k)}] \\
&= (\mathbf{I} - \mu \mathbf{R}_x)^{(k+1)}\mathbb{E}[\mathbf{c}^{(0)}]
\end{aligned}$$

- This expression is similar to the one we obtained for SGD.

Stability

- Using the same derivation that we have performed for the stability analysis of SGD, we can conclude that the LMS algorithm will converge in mean if $|1-\mu\lambda_i| < 1,\ i = 1, 2, ..., M$, where λ_i denotes the eigen values of $\mathbf{R}_x$.

- Hence the LMS algorithm will converge in mean if

$$0 < \mu < \frac{2}{\lambda_{\max}}$$

where $\lambda_{\max}$ is the largest eigen value of $\mathbf{R}_x$.

Stability

- Due to the stochastic nature of the update equation, the LMS algorithm suffers from an excess error. Cost function at the k^{th} iteration is given by

$$J(\mathbf{c}^{(k)}) = J_0 + \underbrace{\mathbb{E}[(\mathbf{c}^{(k)})^{\mathrm{H}}\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{c}^{(k)}]}_{\text{Excess Error}, J_{ex}(k)}$$

- Excess error at the k^{th} iteration, $J_{ex}(k)$ can be written as

$$\begin{aligned} J_{ex}(k) &= \mathbb{E}[(\mathbf{c}^{(k)})^{\mathrm{H}}\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{c}^{(k)}] \\ &= \mathbb{E}[\mathrm{Tr}((\mathbf{c}^{(k)})^{\mathrm{H}}\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{c}^{(k)})] \\ &= \mathbb{E}[\mathrm{Tr}(\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{c}^{(k)}(\mathbf{c}^{(k)})^{\mathrm{H}})] \\ &= \mathrm{Tr}(\mathbb{E}[\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}]\mathbb{E}[\mathbf{c}^{(k)}(\mathbf{c}^{(k)})^{\mathrm{H}}]) \\ &= \mathrm{Tr}(\mathbf{R}_x\mathbf{R}_e) \end{aligned}$$

- Excess error is the trace of the product of data covariance matrix $\mathbf{R}_x$, and the weight error covariance matrix $\mathbf{R}_e = \mathbb{E}[\mathbf{c}^{(k)}(\mathbf{c}^{(k)})^{\mathrm{H}}]$.

Stability

- Under certain conditions, approximate expression for the asymptotic excess error, $J_{ex}(\infty)$ can be computed.
- It can be shown that [Haykin, 2002][2]

$$J_{ex}(\infty) = J_0(\frac{\gamma}{1-\gamma})$$

  if $\gamma < 1$ where

$$\gamma = \sum_{i=1}^{M} \frac{\mu\lambda_i}{2-\mu\lambda_i}$$

- If $\mu\lambda_i << 1$ and $\gamma << 1$,

$$J_{ex}(\infty) \approx \gamma J_0 \approx J_0 \frac{\mu}{2} \sum_{i=1}^{M} \lambda_i = J_0 \frac{\mu}{2} \mathrm{Tr}(\mathbf{R}_x)$$

- The ratio of $J_{ex}(\infty)$ to J_0 is defined as the misadjustment, $\mathcal{M}$, which indicates the asymptotic convergence of the LMS algorithm.

[2] Adaptive Filter Theory, Simon Haykin, fourth edition, Pearson India, 2002.

Stability

- We note that
  $\mathrm{Tr}(\mathbf{R}_x) = \mathrm{Tr}(\mathbb{E}[\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}]) = \mathbb{E}[\mathrm{Tr}(\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}})] = \mathbb{E}[\mathrm{Tr}(\mathbf{x}_k^{\mathrm{H}} \mathbf{x}_k)] = \mathbb{E}[\|\mathbf{x}_k\|^2]$.
- Then the total cost at $k = \infty$ can be written as

$$J(\infty) = J_0 + J_{ex}(\infty) = J_0(1 + \mathcal{M}) \approx J_0(1 + \frac{\mu}{2}\mathbb{E}[\|\mathbf{x}_k\|^2]),$$

  where $\mathcal{M} = J_0/J_{ext}(\infty)$ is the misadjustment.
- The step size is assumed to satisfy

$$0 < \mu < \frac{2}{\mathbb{E}[\|\mathbf{x}_k\|^2]}$$

Normalized LMS

- Update direction in the LMS algorithm is a scaled version of the regressor $\mathbf{x}_k$. Thus the change from $\mathbf{w}^{(k)}$ to $\mathbf{w}^{(k+1)}$ is sensitive to the changes in signal scale.

- To avoid this issue, a normalized version of the LMS algorithm is considered where the update equation is given by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \frac{\mu}{\|\mathbf{x}_k\|^2}\mathbf{x}_k e_k^*,$$

- To avoid the scale dependency, NLMS use a varying step size ($\frac{\mu}{\|\mathbf{x}_k\|^2}$) at each iteration.

Normalized LMS

- Consider a scenario where we select varying step size in each iteration.
- For the k^{th} iteration, we choose the step size μ_k so that the quadratic cost function is minimized.
- We have

$$\begin{aligned} J(\mathbf{w}^{(k)}) = {} & (\mathbf{w}^{(k)} - \mu_k \nabla J_k)^{\mathrm{H}} \mathbf{R}_x (\mathbf{w}^{(k)} - \mu_k \nabla J_k) \\ & - (\mathbf{w}^{(k)} - \mu_k \nabla J_k)^{\mathrm{H}} \mathbf{r}_{xs} - \mathbf{r}_{xs}^{\mathrm{H}} (\mathbf{w}^{(k)} - \mu_k \nabla J_k) + 1 \end{aligned}$$

where $\nabla J_k = \mathbf{R}_x \mathbf{w}^{(k)} - \mathbf{r}_{xs}$.

Normalized LMS

- We choose μ_k so that $J(\mathbf{w}^{(k)})$ is minimized. In other words, we select μ_k so that
$$\left.\frac{\partial J(\mathbf{w}^{(k)})}{\partial \mu}\right|_{\mu=\mu_k} = 0$$

- We have
$$\begin{aligned}\frac{\partial J(\mathbf{w}^{(k)})}{\partial \mu} &= -(\nabla J_k)^{\mathrm{H}}\mathbf{R}_x(\mathbf{w}^{(k)} - \mu_k \nabla J_k) - (\mathbf{w}^{(k)} - \mu_k \nabla J_k)^{\mathrm{H}}\mathbf{R}_x \nabla J_k \\ &\quad + (\nabla J_k)^{\mathrm{H}}\mathbf{r}_{xs} + \mathbf{r}_{xs}^{\mathrm{H}}\nabla J_k \\ &= 2\mu_k(\nabla J_k)^{\mathrm{H}}\mathbf{R}_x \nabla J_k - 2(\nabla J_k)^{\mathrm{H}}\nabla J_k\end{aligned}$$

- Thus
$$\mu_k = \frac{(\nabla J_k)^{\mathrm{H}}\nabla J_k}{(\nabla J_k)^{\mathrm{H}}\mathbf{R}_x \nabla J_k}$$

Normalized LMS

- For the LMS algorithm, we now replace $\mathbf{R}_x$, $\mathbf{r}_{xs}$ and ∇J_k with corresponding instantaneous approximations. We write $\mathbf{R}_x \approx \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}$ and $\mathbf{r}_{xs} \approx \mathbf{x}_k s_k^*$ to obtain

$$\nabla J_k = \mathbf{R}_x \mathbf{w}^{(k)} - \mathbf{r}_{xs} \approx \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} w^{(k)} - \mathbf{x}_k s_k^* = \mathbf{x}_k e_k^*$$

- The optimal step size μ_k can be written as

$$\begin{aligned}
\mu_k &= \frac{(\nabla J_k)^{\mathrm{H}} \nabla J_k}{(\nabla J_k)^{\mathrm{H}} \mathbf{R}_x \nabla J_k} \\
&\approx \frac{(\mathbf{x}_k e_k^*)^{\mathrm{H}} (\mathbf{x}_k e_k^*)}{(\mathbf{x}_k e_k^*)^{\mathrm{H}} \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} (\mathbf{x}_k e_k^*)} \\
&= \frac{|e_k|^2 \mathbf{x}_k^{\mathrm{H}} \mathbf{x}_k}{|e_k|^2 |\mathbf{x}_k^{\mathrm{H}} \mathbf{x}_k|^2} \\
&= \frac{1}{\|\mathbf{x}_k\|^2}
\end{aligned}$$

Normalized LMS

- Hence, we can write the update equations for the modified LMS algorithm with varying step size as

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \frac{\tilde{\mu}}{\|\mathbf{x}_k\|^2}\mathbf{x}_k e_k^*,$$

where $\tilde{\mu}$ is some real constant. This is the NLMS algorithm.

- If we choose $0 < \tilde{\mu} < 2$, then it can be shown that $J_{ex}(\infty)$ is bounded with total cost at $k = \infty$ given by

$$J(\infty) = J_0 + J_{ex}(\infty) \approx J_0(1 + \frac{1}{2}\tilde{\mu})$$

- In practice, a small positive number ϵ is added to the denominator of step size in the NLMS algorithm to avoid divide by zero errors, resulting in the update equation

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \frac{\tilde{\mu}}{\epsilon + \|\mathbf{x}_k\|^2}\mathbf{x}_k e_k^*.$$

Normalized LMS

- Let us recall the update equation of Newton's method, which is given by

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mu(\epsilon\mathbf{I} + \mathbf{R}_x)^{-1}(\mathbf{R}_x\mathbf{w}^{(k)} - \mathbf{r}_{xs}),$$

where ϵ is a small positive number.

- We can arrive at the update equations of NLMS by replacing $\mathbf{R}_x$ and $\mathbf{r}_{xs}$ with corresponding instantaneous estimates.

- Replacing $\mathbf{R}_x$ and $\mathbf{r}_{xs}$ with $\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}$ and $\mathbf{x}_k s_k^*$ respectively, yields

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} - \mu(\epsilon\mathbf{I} + \mathbf{x}_k\mathbf{x}_k^{\mathrm{H}})^{-1}(\mathbf{x}_k\mathbf{x}_k^{\mathrm{H}}\mathbf{w}^{(k)} - \mathbf{x}_k s_k^*)$$

Normalized LMS

- Using matrix inversion lemma, we have

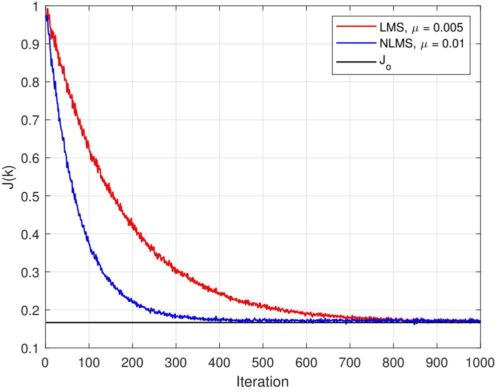
$$(\epsilon \mathbf{I} + \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}})^{-1} = \epsilon^{-1} - \frac{\epsilon^{-2}}{1 + \epsilon^{-1} \|\mathbf{x}_k\|^2} \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}$$

- Thus we get

$$\begin{aligned}
\mathbf{w}^{(k+1)} &= \mathbf{w}^{(k)} - \mu\Big(\epsilon^{-1} - \frac{\epsilon^{-2}}{1 + \epsilon^{-1} \|\mathbf{x}_k\|^2} \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}\Big)\big(\mathbf{x}_k \mathbf{x}_k^{\mathrm{H}} \mathbf{w}^{(k)} - \mathbf{x}_k s_k^*\big) \\
&= \mathbf{w}^{(k)} - \mu\Big(\epsilon^{-1} - \frac{\epsilon^{-2}}{1 + \epsilon^{-1} \|\mathbf{x}_k\|^2} \mathbf{x}_k \mathbf{x}_k^{\mathrm{H}}\Big) \mathbf{x}_k e_k^* \\
&= \mathbf{w}^{(k)} - \mu\Big(\epsilon^{-1} \mathbf{x}_k - \frac{\epsilon^{-2}}{1 + \epsilon^{-1} \|\mathbf{x}_k\|^2} \mathbf{x}_k \|\mathbf{x}_k\|^2\Big) e_k^* \\
&= \mathbf{w}^{(k)} - \frac{\mu}{\epsilon + \|\mathbf{x}_k\|^2} \mathbf{x}_k e_k^*
\end{aligned}$$

Normalized LMS

- Just like we have derived LMS from SGD, we have now derived NLMS from Newton's method. Recall that the convergence of Newton's method is superior to that of SGD.



- It turns out that the NLMS converges faster in comparison to the LMS for comparable values of excess errors.

Kaczmarz method

- Kaczmarz method is an algorithm to iteratively solve a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{y}$.
- Starting from an inital guess $\mathbf{x}_0$, Kaczmarz method refines this estimate by considering each row of $\mathbf{A}\mathbf{x} = \mathbf{y}$, one after other.
- In the i^{th} step, given the estimate $\mathbf{x}_{i-1}$ and the equation $\mathbf{a}_i^{\mathrm{H}}\mathbf{x} = y_i$, Kaczmarz method obtains $\mathbf{x}_i$ by solving the following optimization problem

$$\text{minimize}\|\mathbf{x}_i - \mathbf{x}_{i-1}\|^2 \quad \text{subject to} \quad \mathbf{a}_i^{\mathrm{H}}\mathbf{x}_i = y_i$$

where $\mathbf{a}_i^{\mathrm{H}}$ denotes the i^{th} row of $\mathbf{A}$.

Kaczmarz method

- Geometrically, in the i^{th} iteration, the Karczmarz method computes the point in the hyperplane $\mathbf{a}_i^{\mathrm{H}}\mathbf{x} = y_i$ which is nearest to $\mathbf{x}_{i-1}$ in the sense of Eucledian distance, i.e., $\mathbf{x}_i$ is the orthogonal projection of $\mathbf{x}_{i-1}$ onto the hyperplane $\mathbf{a}_i^{\mathrm{H}}\mathbf{x} = y_i$.

- Update equations of the Kaczmarz method is given by

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \frac{y_i - \mathbf{a}_i^{\mathrm{H}}\mathbf{x}_{i-1}}{\|\mathbf{a}_i\|^2}\mathbf{a}_i$$

- This is exactly similar to the update equations of NLMS method with $\mu = 1$ and $\epsilon = 0$.

- Even though NLMS and Kaczmarz method were developed using different approaches, both methods solve the same optimization problem.