From LMMSE to LMS and Adaptive Filtering

EE599 Deep Learning

Kuan-Wen (James) Huang and Arnab Sanyal

Spring 2020

Contributors: Kuan-Wen Huang, Keith Chugg, Arindam Jati

Problem: estimate $y(u)$ from $\boldsymbol{x}(u) = \boldsymbol{x}$

- Problem setting: Given a vector observation $\boldsymbol{x}(u) = \boldsymbol{x}$, we would like to estimate $y(u)$ via a filter $\hat{y} = \boldsymbol{w}^T \boldsymbol{x}$ where mean squared error (MSE) is minimized.

- The objective (cost function) to be minimized is MSE $= \boldsymbol{E}\{[y(u) - \boldsymbol{w}^T \boldsymbol{x}]^2\}$. The design variables of the filter are $\boldsymbol{w}$.

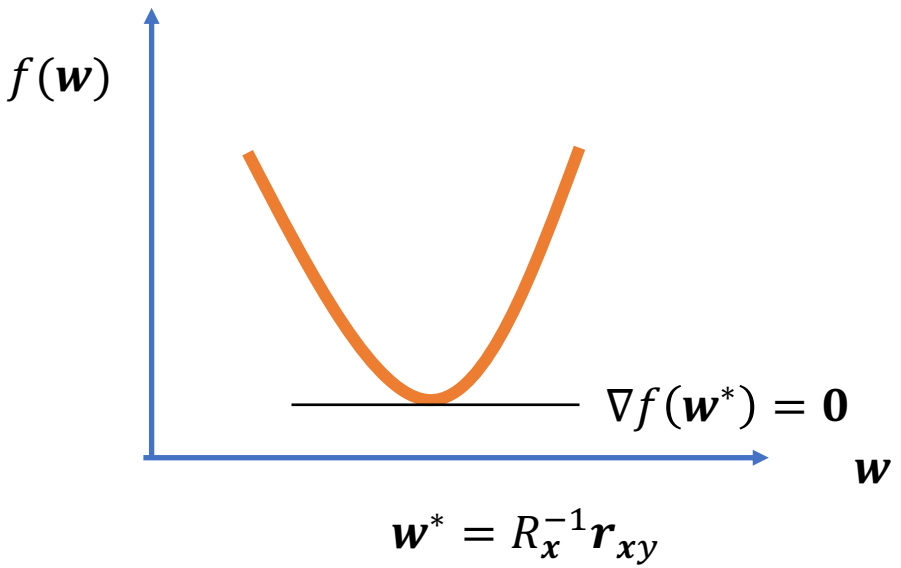
LMMSE

- Assume that we know the second order statistics of the joint distribution $p_{\boldsymbol{x}(u),y(u)}(\boldsymbol{x}, y)$, i.e. $\boldsymbol{m}_x, m_y, R_{\boldsymbol{x}}, r_y, \boldsymbol{r}_{\boldsymbol{xy}}$.

- The objective is a quadratic function of $\boldsymbol{w}$,

$$\begin{aligned}\text{MSE} &= \boldsymbol{E}\{[y(u) - \boldsymbol{w}^T\boldsymbol{x}]^2\} \\ &= r_y - 2\boldsymbol{r}_{yx}^T\boldsymbol{w} + \boldsymbol{w}^T R_{\boldsymbol{x}}\boldsymbol{w}.\end{aligned}$$

- The global optimal occurs at $\boldsymbol{w}^*$, where $\nabla f(\boldsymbol{w})|_{\boldsymbol{w}=\boldsymbol{w}^*} = \boldsymbol{0}$.



Gradient Descent

- Instead of obtain the optimal $\boldsymbol{w}^*$ directly, we can also find it iteratively via
  1. Initialize $\boldsymbol{w}_0$
  2. $\boldsymbol{w}_{n+1} = \boldsymbol{w}_n - \eta \nabla f(\boldsymbol{w}_n)$
     $= \boldsymbol{w}_n + 2\eta(\boldsymbol{r}_{xy} - R_x \boldsymbol{w}_n)$

source: https://blog.clairvoyantsoft.com/

Remark

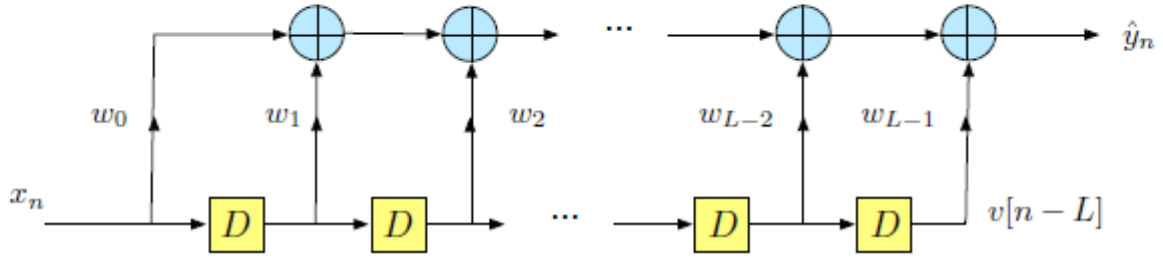
1. Closed-form global optimality can be derived if we have a convex and differentiable cost function.
2. Gradient descent works in any differentiable cost function.

3. What if we don’t have second order statistics but lots of samples?

   Single point gradient descent or small batch gradient descent!

LMS Algorithm

- In the case where the index n corresponds to time and we have tons of samples $\{(\boldsymbol{x}_n, y_n)\}$ instead of the second order statistics, we use an online learning algorithm called least mean square adaptive filtering.



$$\hat{y}_n = \sum_{l=0}^{L-1} w_l x_{n-l} = \mathbf{w}^{\mathrm{t}} \mathbf{v}_n$$

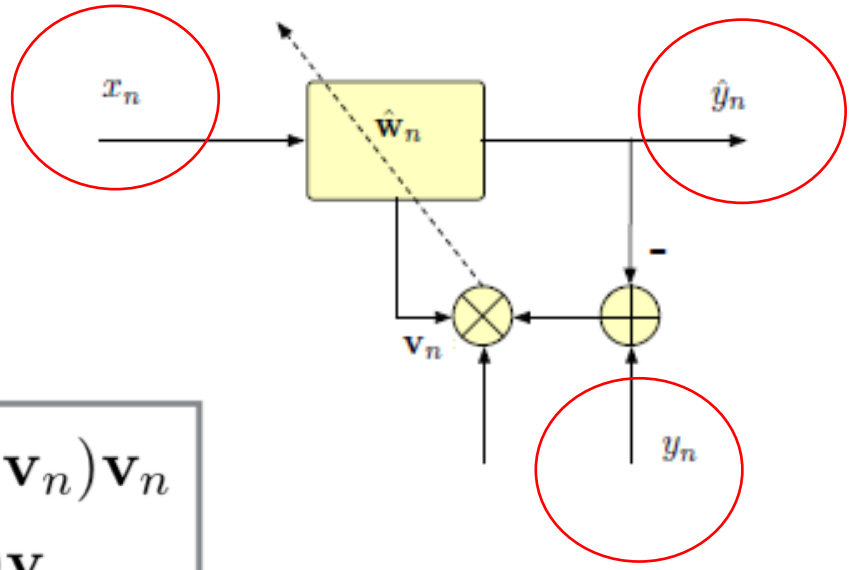
$$\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{L-1} \end{bmatrix}$$

$$\mathbf{v}_n = \mathbf{x}_{n-(L-1)}^{n} = \begin{bmatrix} x_n \\ x_{n-1} \\ \vdots \\ x_{n-L+1} \end{bmatrix}$$

LMS Algorithm

Single Point Stochastic Gradient Descent:

$$
\begin{aligned}
-\frac{1}{2}\nabla_{\mathbf{w}} E &= \mathbf{r}_n - \mathbf{R}_{\mathbf{v}_n}\mathbf{w} \\
&= \mathbb{E}\left\{y(u)\mathbf{v}_n(u) - \mathbf{v}_n(u)\mathbf{v}_n^{\mathrm{t}}(u)\mathbf{w}\right\} \\
&\approx (y_n - \mathbf{w}_n^{\mathrm{t}}\mathbf{v}_n)\mathbf{v}_n
\end{aligned}
$$



LMS Algorithm

$$
\begin{aligned}
\hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n + \eta(y_n - \hat{\mathbf{w}}_n^t\mathbf{v}_n)\mathbf{v}_n \\
\hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n + \eta(y_n - \hat{y}_n)\mathbf{v}_n \\
\hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n + \eta e_n\mathbf{v}_n
\end{aligned}
$$

(this is an on-line linear regressor)

- Same update equation: $\widehat{\boldsymbol{w}}_{n+1} = \widehat{\boldsymbol{w}}_n + \eta\left(y_n - \widehat{\boldsymbol{w}}_n{}^T\boldsymbol{v}_n\right)\boldsymbol{v}_n$

LMS Algorithm

- Assumptions:
  - No access to the data signal, s(n). We can only observe y(n) = (s(n) + noise)
  - We have a reference noise signal, x(n), which is highly correlated with the noise present in data
- Why we need this?: Slowly drifting interfering sinusoid, static notch filter not enough
- Benefits:
  - Behaves as an adaptive notch filter
  - The notch can be very sharp, depending upon step size

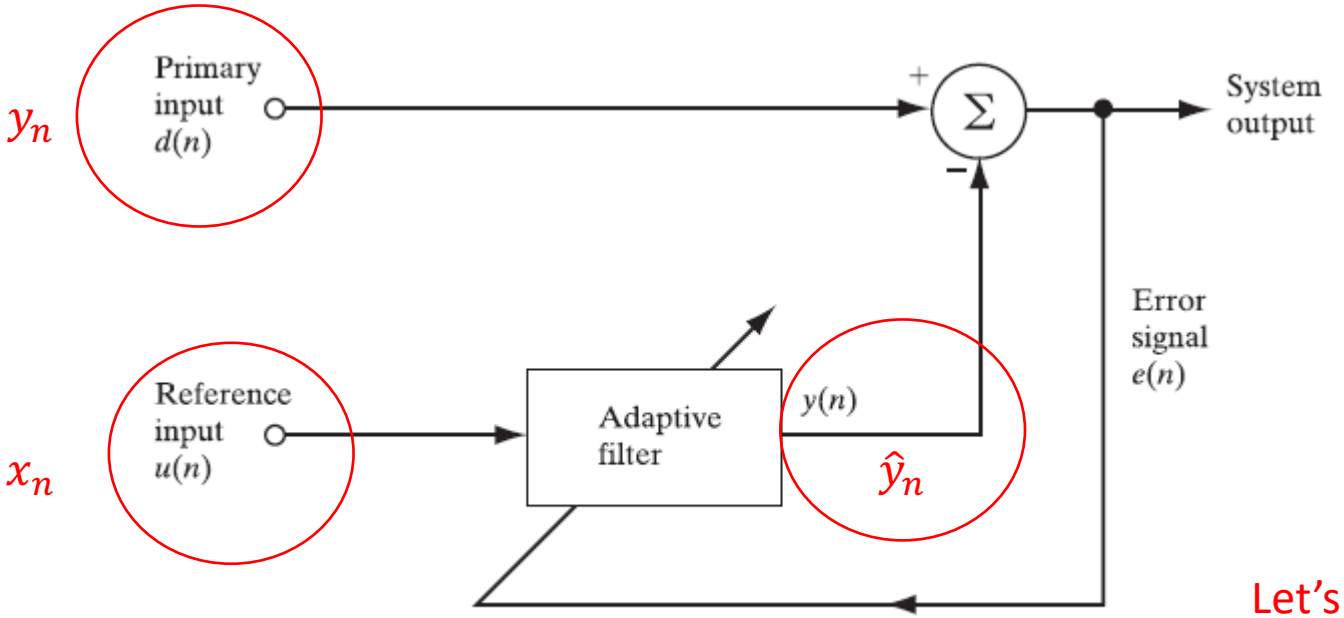


FIGURE 6.6 Block diagram of adaptive noise canceller.

- The reference input supplies a correlated version of the sinusoidal interference.
- For the adaptive filter, we may use an FIR filter whose tap weights are adapted by means of the LMS algorithm

Let's be consistent in notations between notes and book

- **Source**: *Simon Haykin, Adaptive Filter Theory* (5th Edition), Pearson, Section 6.3, Application 4: Adaptive Noise Cancelling Applied to a Sinusoidal Interference

LMS Algorithm: for sinusoidal noise cancellation

Primary input:

$$y(n) = s(n) + A_0 \cos(\omega_0 n + \phi_0) \tag{1}$$

Reference input:

$$x(n) = A \cos(\omega_0 n + \phi) \tag{2}$$

$s(n)$ = **information bearing signal / data signal.**

LMS: get current estimate and error:

$$\begin{aligned} \hat{y}(n) &= \sum_{i=0}^{L-1} \hat{w}_i(n) x(n-i) \\ &= \hat{\mathbf{w}}_n^T \mathbf{v}_n \end{aligned} \tag{3}$$

$$e(n) = y(n) - \hat{y}(n) = y(n) - \hat{\mathbf{w}}_n^T \mathbf{v}_n \tag{4}$$

Update filter tap weights:

$$\begin{aligned} \hat{\mathbf{w}}_{n+1} &= \hat{\mathbf{w}}_n + \eta e(n) \mathbf{v}_n \\ &= \hat{\mathbf{w}}_n + \eta (y(n) - \hat{\mathbf{w}}_n^T \mathbf{v}_n) \mathbf{v}_n \end{aligned} \tag{5}$$

Data Storage – hd5

```python
import h5py
# Save data in a hdf5 file
# create a file, in 'w' mode for writing
with h5py.File("test.hdf5", "w") as f:
    f.create_dataset("x1", data=x1sr)
    f.create_dataset("x2", data=x2)
# File handler f is automatically closed once we exit the block
```

- HDF5 – Stands for Hierarchical Data Format
  - good for storing multiple datasets in a single file
  - fast non-sequential access, readers available
  - Readers available
  - Can store more than 20000 complex objects
  - Can be more than 2 gigabytes in size
  - Very consistent and built for speed not restricted to datatypes
  - Speed of writing a hd5 file is much faster than say npz
  - Supports parallel IO
  - Easy to use in threaded applications
- Further reading – https://support.hdfgroup.org/HDF5/doc/H5.intro.html

Data Storage – npz

- Numpy's very own data storage types – npy and npz
  - Save several arrays into a single file in uncompressed .npy format
  - If no keys passed then default keys are arr_0, arr1 etc. Else matches keys passed
  - Can be saved with `numpy.savez()` and is not a compressed archive
  - The .npz file format is a zipped archive of files, named after the variables they contain
  - Need to save `.npz` using the function `numpy.savez_compressed()`
  - Use numpy.load() for loading `.npz` and `.npy` files
- Further Reading,
  - https://docs.scipy.org/doc/numpy/reference/generated/numpy.savez.html
  - https://docs.scipy.org/doc/numpy/reference/generated/numpy.savez_compressed.html#numpy.savez_compressed
  - https://docs.scipy.org/doc/numpy/reference/generated/numpy.lib.format.html#module-numpy.lib.format

Data storage – mat

- Datatype first developed by Mathworks, popularly used in MATLAB
  - Stores files in binary data container format and can store vars. funcs. arrs., etc
  - If you double click on a `.mat` file then your app opens it (often MATLAB)
  - For `.mat` files conforming upto MATLAB v. 7.1, we can use `scipy.io`
  - Use `loadmat()` for loading and `savemat()` for saving
  - For v. 7.3 and higher, `mat` files are saved using the HDF5 format by default
  - Use the package `h5py` or `PyTables` to read `mat` files
- Further Reading,
  - https://scipy-cookbook.readthedocs.io/items/Reading_mat_files.html
  - https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.savemat.html
  - https://docs.scipy.org/doc/scipy/reference/generated/scipy.io.loadmat.html
  - https://stackoverflow.com/questions/36362831/creating-a-mat-file-of-v7-3-in-python

Data storage – pickle

```
import pickle

example_dict = {1:"6",2:"2",3:"f"}

pickle_out = open("dict.pickle","wb")
pickle.dump(example_dict, pickle_out)
pickle_out.close()
```

- "Serialises" the object first before writing it to file.
  - Converts python objs (lists, dicts) into a character stream and vice-versa
  - No restrictions imposed as pickle is Python-specific
  - However non-Python programs may not be able to reconstruct pickled Python objects.
  - Has optimal size characteristics and efficiently compresses data.
  - 6 different protocols for pickling – higher protocols used by more recent python versions
  - Reading, loading, writing pickle files is simple due to APIs present in `pickle` library

```
pickle_in = open("dict.pickle","rb")
example_dict = pickle.load(pickle_in)
```

- Further Reading,
  - https://docs.python.org/3/library/pickle.html
  - https://pythonprogramming.net/python-pickle-module-save-objects-serialization/

Pandas

- Stands for Python Data Analysis Library
  - Software library written for python for data manipulations and analysis
  - Open source BSD licensed
  - Provides high performance, data structures and data analysis tools
  - In particular offers data structures for manipulating numerical tables and time-series
  - Initial release in Jan 2008, most stable release in August 2019
  - Written in Python, Cython and C.
- Further Reading,
  - https://pandas.pydata.org/docs/user_guide/index.html
  - https://pandas.pydata.org/docs/reference/index.html
  - https://pandas.pydata.org/docs/getting_started/index.html

Tar gzip

- `tar` – used for packaging files (and/or compressing it)
- Usage –
  - `tar -czvf filename.tar.gz <files>`
  - `tar -xzvf filename.tar.gz`
- Flags – ‘c’ stands for **c**ompress and ‘x’ stands for e**x**pand
  - ‘z’ stands for gzipped compression
  - ‘v’ stands for verbose
  - ‘f’ implies that the file path is specified
- `tar` can work without gzipping and vice-versa `gzip -d file.gz` or `gunzip file.gz`
- Additional resources – type the following on your UNIX terminal
  - `man tar`
  - `man gzip`
  - `man gunzip`