Plane Fit Problem

Vikas Dhiman

Thursday 25th September, 2025

```
!pip install otter -grader jupyterlab -myst
```

```
# Initialize Otter
import otter
grader = otter.Notebook("02 -PlaneFitProblem.ipynb")
```

1 Homework (PlaneFit): Problem 1

(10 marks)

Find the minimum of the following function

$$f(\mathbf{x}) = \mathbf{x}^\top A^\top A\mathbf{x} + 2\mathbf{b}^\top \mathbf{x} + c \tag{1}$$

Type your answer here, replacing this text.

1.1 Homework (Plane fit): Problem 2

(10 marks)

Find the minimum of the following function (assume $\Sigma_1 \in R^{n\times n}$ and $\Sigma_2 \in R^{n\times n}$ to be symmetric and positive semi-definite matrices).

$$f(\mathbf{x}) = (\mathbf{x} - \boldsymbol{\mu}_1)^\top \Sigma_1 (\mathbf{x} - \boldsymbol{\mu}_1) + (\mathbf{x} - \boldsymbol{\mu}_2)^\top \Sigma_2 (\mathbf{x} - \boldsymbol{\mu}_2) \tag{2}$$

Assume $\mathbf{x} \in R^n$, $\boldsymbol{\mu}_1 \in R^n$ and $\boldsymbol{\mu}_2 \in R^n$.

Σ is not the summation symbol here, it is just another matrix.

Type your answer here, replacing this text.

1.2 Homework (Plane fit): Problem 3

(30 marks)

Least square fit in 2D fits a line to a given set of 2D points. This homework asks you to repeat the minimization procedure to find a plane that fits a 2D points. Below you are provided with helper code to load a point cloud, and visualize the points and the plane. Your task is to find the equation of plane that fits the point cloud.

1. Show the math and derivation for your code
2. Implement the code in python and numpy to find the coefficients of the equation of plane and write your answer in the space provided below. Submit this ipynb file to brightspace. (30 marks)

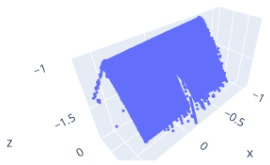
```
%pip install plotly requests
def wget(url, filename):
    """
    Download files using requests package
    """
    import requests
    r = requests.get(url)
    with open(filename, 'wb') as fd:
        for chunk in r.iter_content():
            fd.write(chunk)
wget('https://github.com/wecacuee/ECE490 -S24 -Neural -Networks/raw/master/notebooks/02 -lin
     'table_top.npy')
```

```
Requirement already satisfied: plotly in /home/vdhiman/.local/venvs/ece490/lib/python3.10/si
Requirement already satisfied: requests in /home/vdhiman/.local/venvs/ece490/lib/python3.10/
Requirement already satisfied: narwhals>=1.15.1 in /home/vdhiman/.local/venvs/ece490/lib/pyt
Requirement already satisfied: packaging in /home/vdhiman/.local/venvs/ece490/lib/python3.10
Requirement already satisfied: charset_normalizer<4,>=2 in /home/vdhiman/.local/venvs/ece490
Requirement already satisfied: idna<4,>=2.5 in /home/vdhiman/.local/venvs/ece490/lib/python3
Requirement already satisfied: urllib3<3,>=1.21.1 in /home/vdhiman/.local/venvs/ece490/lib/p
Requirement already satisfied: certifi>=2017.4.17 in /home/vdhiman/.local/venvs/ece490/lib/p

[notice] A new release of pip is available: 25.0.1 -> 25.2
[notice] To update, run: pip install - -upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

def points_on_plane(abc):
    a, b, c = abc
    x = np.linspace( -1, 1, 100, dtype=np.float32)
    y = np.linspace(0, 1, 100, dtype=np.float32)

    x, y = np.meshgrid(x, y)
    z = a*x + b*y + c
    return x, y, z

def visualize_plane(pts, abc, fig=None):
    _fig = fig
    fig = make_subplots() if fig is None else fig
```

```
    x, y, z = points_on_plane(abc)
    plt_surface = go.Surface(x=x, y=y, z=z)

    fig.add_trace(plt_surface)
    visualize_points(pts, fig=fig)
    if _fig is None:
        fig.show()

def visualize_points(pts, fig=None):
    _fig = fig
    fig = make_subplots() if fig is None else fig
    scatter_trace = go.Scatter3d(x=pts[:, 0], y=pts[:, 1], z=pts[:, 2], mode='markers', mark
    fig.add_trace(scatter_trace)
    if _fig is None:
      fig.show()
```

```
import numpy as np
pts = np.load('table_top.npy').astype('float32') # Nx3
visualize_points(pts)
pts.min(axis=0), pts.max(axis=0)
```




```
(array([ -1.0558e+00,  3.2414e -04, -1.8675e+00], dtype=float32),
 array([ 0.59304,  0.53329, -1.0252 ], dtype=float32))
```

```
def plane_fit(pts):
    ...
    return abc

# Find abc such that z = a x + b y + c
abc = plane_fit(pts)
visualize_plane(pts, abc)
def test_plane_fit(plan_fit, env):
    pts = env['pts']
    abc = plane_fit(pts)
    assert np.abs(pts[:,2] - np.hstack((pts[:, :2], np.ones((pts.shape[0],1)))) @ abc).mean(
test_plane_fit(plane_fit, env=globals())
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
NameError                                 Traceback (most recent call last)
Cell In[6], line 2
      1 # Find abc such that z = a x + b y + c
 - - - -> 2 abc = plane_fit(pts)
      3 visualize_plane(pts, abc)
      4 def test_plane_fit(plan_fit, env):

Cell In[5], line 3, in plane_fit(pts)
      1 def plane_fit(pts):
      2     ...
 - - - -> 3     return abc

NameError: name 'abc' is not defined

grader.check("q8")
```

1.3 Submission

Make sure you have run all cells in your notebook in order before running the cell below, so that all images/graphs appear in the output. The cell below will generate a zip file for you to submit. **Please save before exporting!**

```
# Save your notebook first, then run this cell to export your submission.
grader.export(pdf=False, run_tests=True)
```