EECS 16B   Module 3, Lecture 7.   Apr 22, 2021.

Today

- Machine Learning terminology.
- Classification continued.

Terminology

Training data: Data to help you learn (your classifier, predictor etc.) e.g. $A\vec{x} = \vec{b}$
$(\vec{a}_1^T, b_1)$

Test data: Data to check / test if what you learned is any good.

$(\vec{a}_{n+1}^T, b_{n+1})$
$\hat{\vec{x}}$

Classification

Initial data: $\vec{x}_1, \vec{x}_2 \cdots \vec{x}_m$ (m data points)

→ pixels of an image
→ observations of a planet.

Associated label: $l_1, l_2 \cdots l_m$

Binary classification $\quad l_i \in \{+1, -1\}$

$\{+1\}$, $\{-1\}$
cat dog
Neuron 1 Neuron 2.

$$\left\{(\vec{x}_1, \underset{\phi}{\ell_1}), (\vec{x}_2, \underset{\phi}{\ell_2}) \cdots (\vec{x}_m, \ell_m)\right\}$$

Find a classifier. In particular, we want to find a linear classifier.

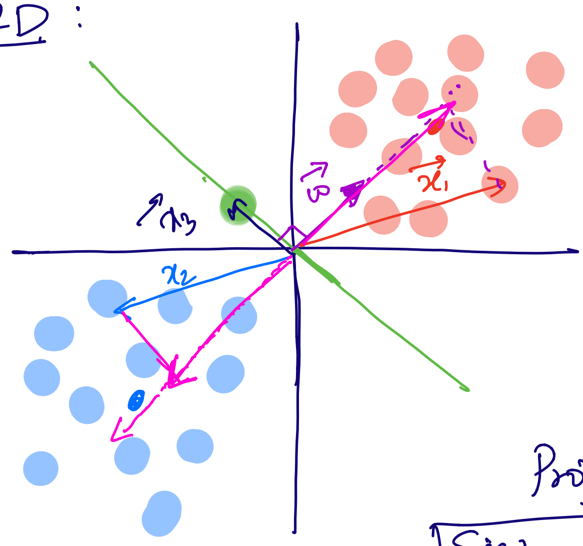
$$f(\vec{x}_1) \longrightarrow \ell_1$$

We want to find a vector $\vec{w}$ such that:

$$\text{sign}(\vec{x}^T \vec{w})$$

$$\text{sign}(x) = +1 \quad \text{if } x > 0$$
$$\text{sign}(x) = -1 \quad \text{if } x < 0$$

In 2D:



$\|\vec{w}\| = 1$

Find $\vec{w}$ such that

$\vec{x}_i^T \vec{w} > 0$ if $\vec{x}_i \in \{\text{Red}\}$

$\vec{x}_i^T \vec{w} < 0$ if $\vec{x}_i \in \{\text{Blue}\}$.

Proj of $\vec{x}$ onto $\vec{w}$: $\dfrac{\vec{x}^T\vec{w}}{\|\vec{w}\|^2} \cancel{\vec{w}}$, $\|\vec{w}\|^2 = 1$

Consider $\underbrace{\vec{x}_1^T \vec{w}} > 0$: $\boxed{\text{Signed}}$ $\boxed{\text{Magnitude}}$ of projection of $\vec{x}_1$ onto $\vec{w}$

Consider $\vec{x_2}^T \vec{w} < 0$: true for blue points

Goal: To find $\vec{w}$. How?

"Cost-function" $\rightarrow$ penalty for being wrong.

$\vec{x_i}^T \vec{w} \longrightarrow l_i$

One possible cost function:

$$\underset{\vec{w}}{\operatorname{argmin}} \sum_{i=1}^{m} \underbrace{(\vec{x_i}^T \vec{w} - l_i)^2}$$

$\hookrightarrow$ if $\vec{x_i}^T \vec{w} = l_i$ = good.

$\vec{x_i}^T \vec{w} \neq l_i$ = bad.

$$= \underbrace{\underset{\vec{w}}{\operatorname{argmin}} \left\| \begin{bmatrix} - \vec{x_1}^T - \\ - \vec{x_2}^T - \\ \vdots \\ - \vec{x_m}^T - \end{bmatrix} \begin{bmatrix} \\ \vec{w} \\ \\ \end{bmatrix} - \begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_m \end{bmatrix} \right\|^2}_{\text{Least squares.}}$$

Nothing is special about $(\vec{x}_i^T \vec{w} - l_i)^2$.

<u>General cost function:</u>

$$C(\vec{x}_i^T \vec{w}, l_i) \longrightarrow$$

$\vec{x}_i^T w$ has the same sign as l_i $\rightarrow$ good.
want $C(\cdot)$ to be small.

$\vec{x}_i^T \vec{w}$ has opposite sign as l_i
This is bad.
$C(\cdot)$ to be large.

<u>We chose:</u> $C(\vec{x}_i^T \vec{w}, l_i) = \exp(-l_i \vec{x}_i^T \vec{w})$

When $\text{sign}(\vec{x}_i^T \vec{w}) = l_i$

$\exp(\text{negative}) \longrightarrow \text{small}.$

$\text{sign}(\vec{x}_i^T \vec{w}) \neq l_i \leftarrow$ error

$\exp(\text{positive}) \longrightarrow \text{big}$ High cost function!

$$\rightarrow \underset{\vec{w}}{\text{argmin}} \sum_{i=1}^{m} \exp(-l_i \vec{x}_i^T \vec{w})$$

Our strategy: Make this cost function look like a quadratic. $\vec{w} \in \mathbb{R}^n$

Taylor approximation.

row vector

Derivative.

$$f(\vec{w}) \approx f(\vec{w}_*) + \left.\frac{df}{d\vec{w}}\right|_{\vec{w}=\vec{w}_*} (\vec{w} - \vec{w}_*)$$

$$+ \frac{1}{2}(\vec{w} - \vec{w}_*)^T \left.\frac{d^2 f}{d\vec{w}^2}\right|_{\vec{w}=\vec{w}_*} (\vec{w} - \vec{w}_*)$$

matrix

Hessian

$$= f(\vec{w}_*) + \left.\begin{bmatrix} \frac{\partial f}{\partial w_1} & \cdots & \frac{\partial f}{w_n} \end{bmatrix}\right|_{\vec{w}=\vec{w}_*} (\vec{w} - \vec{w}_*)$$

$$+ \frac{1}{2}(\vec{w} - \vec{w}_*)^T \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \vdots & & \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{bmatrix} (\vec{w} - \vec{w}_*)$$

To find our quadratic form, we need an operating point $\vec{w}_*$.

But to find our operating point, we need a quadratic form!?

Solution: Consider an iterative algorithm.
(Newton's method) $\rightarrow$ roots of a polynomial.

Algorithm:

1. Arbitrarily choose an operating point
$$\vec{w}_* = \vec{w}[0] = \vec{0}$$

2. Quadraticize around $\vec{w}_*$
$$f(\vec{w}) = \sum_{i=1}^{m} \underbrace{C(\vec{x}_i^T \vec{w}, l_i)}_{\text{cost function}} \quad \text{around } \vec{w}_*$$
$$f(\vec{w}) \approx \underset{\uparrow \atop \text{matrix}}{\vec{w}^T A \vec{w}} + \underset{\uparrow \atop \text{vector}}{\vec{b}^T \vec{w}} + \underset{\uparrow \atop \text{scalar}}{d} \quad \text{(Generic form of quadratic)}$$

3. Find the minimizer of the quadratic.
Call this $\vec{w}[1]$

4. Set $\vec{w}_* = \vec{w}[1]$, and go back to 2.

$\rightarrow$ Stop when $\vec{w}[k]$ and $\vec{w}[k+1]$ are very close to each other.

$f(\vec{w}) \approx$ **Quadratic approx**

$$f(\vec{w}_*) + \left[\frac{\partial f}{\partial w_1} \cdots \frac{\partial f}{\partial w_n}\right]\Bigg|_{\vec{w}=\vec{w}_*} (\vec{w}-\vec{w}_*)$$

$$+ \frac{1}{2}(\vec{w}-\vec{w}_*)^T \begin{bmatrix} \frac{\partial^2 f}{\partial w_1^2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_n} \\ \vdots & & \\ \frac{\partial^2 f}{\partial w_n \partial w_1} & \cdots & \frac{\partial^2 f}{\partial w_n^2} \end{bmatrix} (\vec{w}-\vec{w}_*)$$

$$f(\vec{w}) = \sum_{i=1}^{m} \exp(-l_i \vec{x}_i^T \vec{w})$$

Consider: Partial of

$$\frac{\partial(\exp(-l_i \vec{x}_i^T \vec{w}))}{\partial w_1}$$

$$= \frac{\partial(\exp(-l_i(x_1 w_1 + x_2 w_2 \cdots + x_n w_n)))}{\partial w_1}$$

$$= -l_i x_1 \exp(-l_i \vec{x_i}^T \vec{w})$$

Quadratic approximation:

$$\sum_{i=1}^{m} c(\vec{x_i}^T \vec{w}, l_i) \approx$$

constant

$$\sum_{i=1}^{n} \Big[c(\vec{x_i}^T \vec{w_*}, l_i) - l_i \exp(-l_i \vec{x_i}^T \vec{w_*}) \vec{x_i}^T (\vec{w} - \vec{w_*})$$

$$+ \frac{1}{2} \exp(-l_i \vec{x_i}^T \vec{w_*}) \langle \vec{x_i}, \vec{w} - \vec{w_*} \rangle^2 \Big]$$
