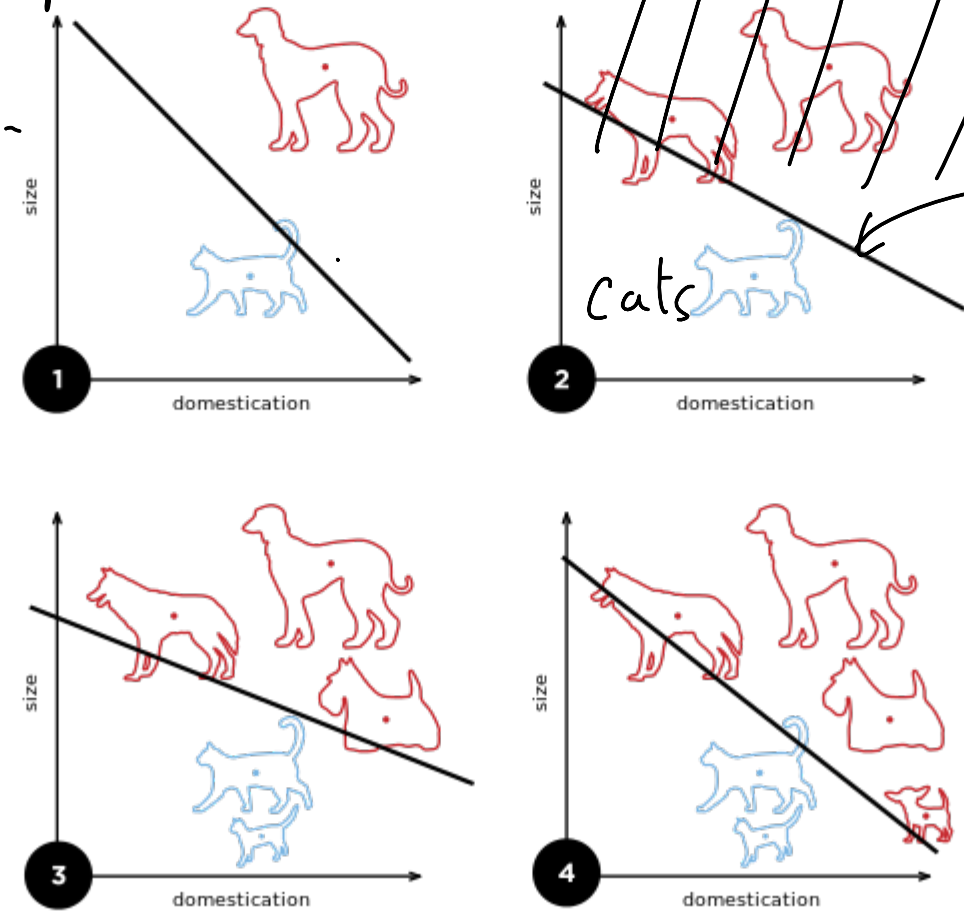
Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→Restart) and then **run all cells** (in the menubar, select Cell→Run All).

Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and collaborators below:

In [ ]:
```
NAME = ""
COLLABORATORS = ""
```

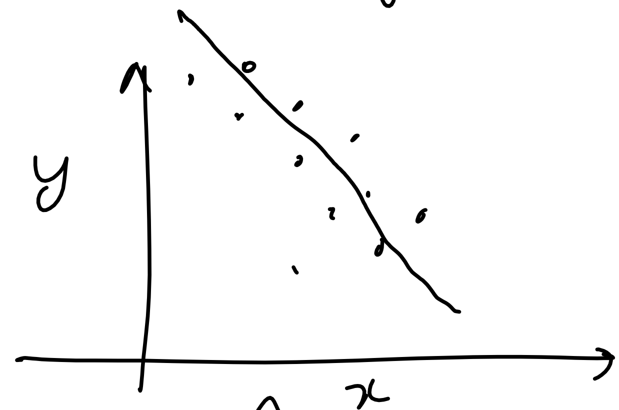
Perceptrons 1946

Dogs



Line

Linear classifiers



Linear regression $y \in \mathbb{R}$

$$y = f(x) \qquad y \in \mathbb{R}$$

$$y = f(\underline{x}) \qquad \underline{x} \in \mathbb{R}^n$$

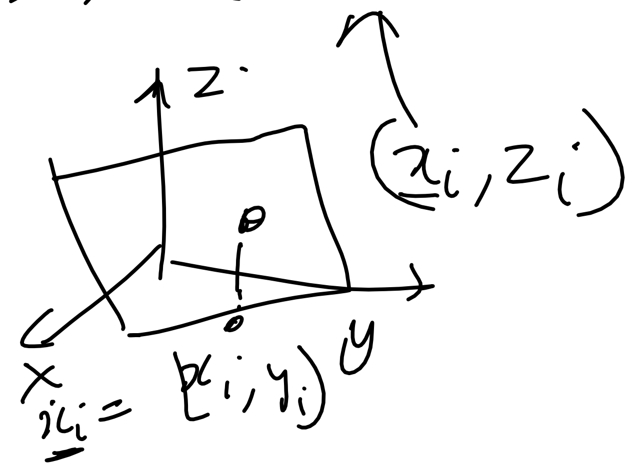
$$f(\underline{x}) = a x_i + b y_i + c$$

$$z_i = f(\underline{x}) = a \underline{x}_i + b y_i + c$$

for given Data = $\{(\underline{x}_i, y_i), \ldots (\underline{x}_n, y_n)\}$

$\underline{x}_i \in \mathbb{R}^n$

$z_i = f(\underline{x}_i)$ $\quad Z_i \in \mathbb{R}$



Find linear function f

Linear regression

What is a linear function

① First order poly nomial

② $g(\underline{x}) : \mathbb{R}^n \longmapsto \mathbb{R}$

a) $g(\underset{\in \mathbb{R}}{\alpha} \underline{x}) = \alpha g(\underline{x})$

b) $g(\underline{x} + \underline{y}) = g(\underline{x}) + g(\underline{y})$

Linear function

$g(\alpha \underline{x} + \beta \underline{y}) = \alpha g(\underline{x}) + \beta g(\underline{y})$

All linear functions can be written as

$$g(\underline{x}) : \mathbb{R}^n \longmapsto \mathbb{R}$$

$$g(\underline{x}) = \underline{w}^T \underline{x}$$

$$g\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = g\left(x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$$

$$= x_1 \underbrace{g\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)}_{w_1} + x_2 \underbrace{g\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)}_{w_2}$$

$$= \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= \underline{w}^T \underline{x}$$

---

in 2D they are lines

$\underline{w}^T \underline{x} + w_0$ → × Linear ✓ Affine

→ loosely called Linear

and Planes in 3D, Hyper planes in nD

Linear classification

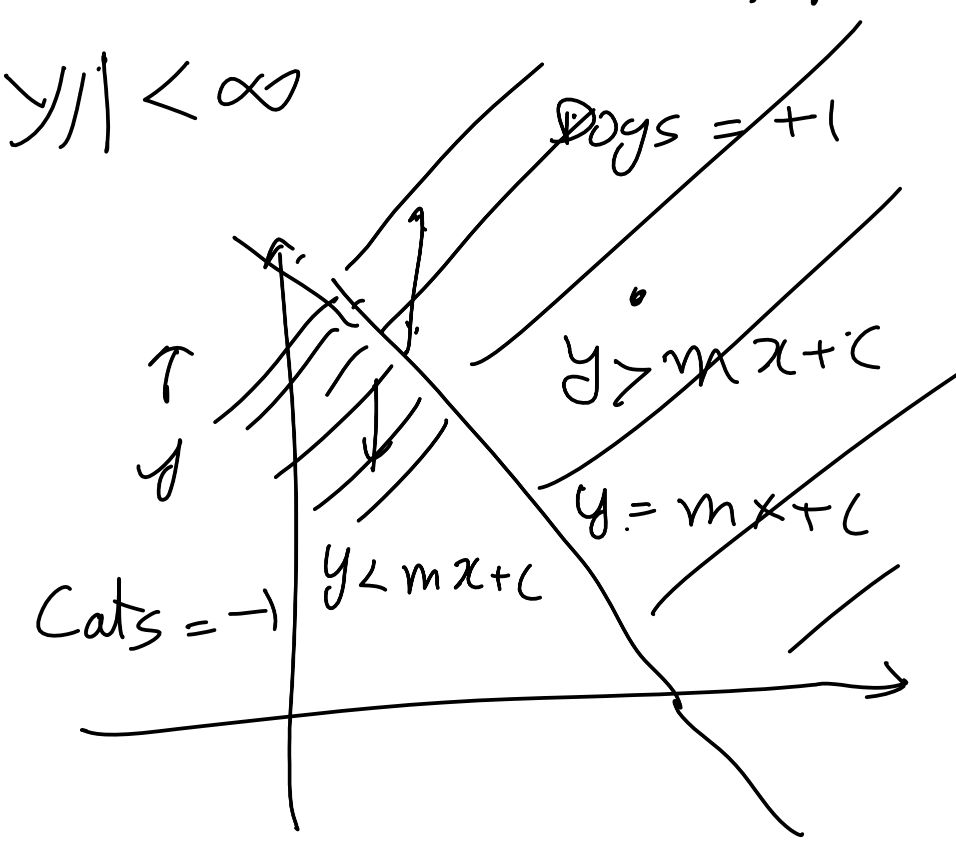
$$\text{Data} = \{(\underline{x}_1, l_1), (\underline{x}_2, l_2), \ldots (\underline{x}_n, l_n)\}$$

$l_1 \notin \mathbb{R}$

$l_i \in \{0, 1, \ldots, 10\}$

$l_i \in \mathcal{Y}$ $\quad |\mathcal{Y}| < \infty$

halfspaces



Find a linear function, f

$$l_i = \left[c_{j_1} > f(\underline{x}_i) > c_{j_2}\right]$$

$$l_i = c_{j_1} > \underline{w}^T \underline{x}_i > c_{j_2}$$

$l_i = +1$ if $\infty > mx + c > 0$

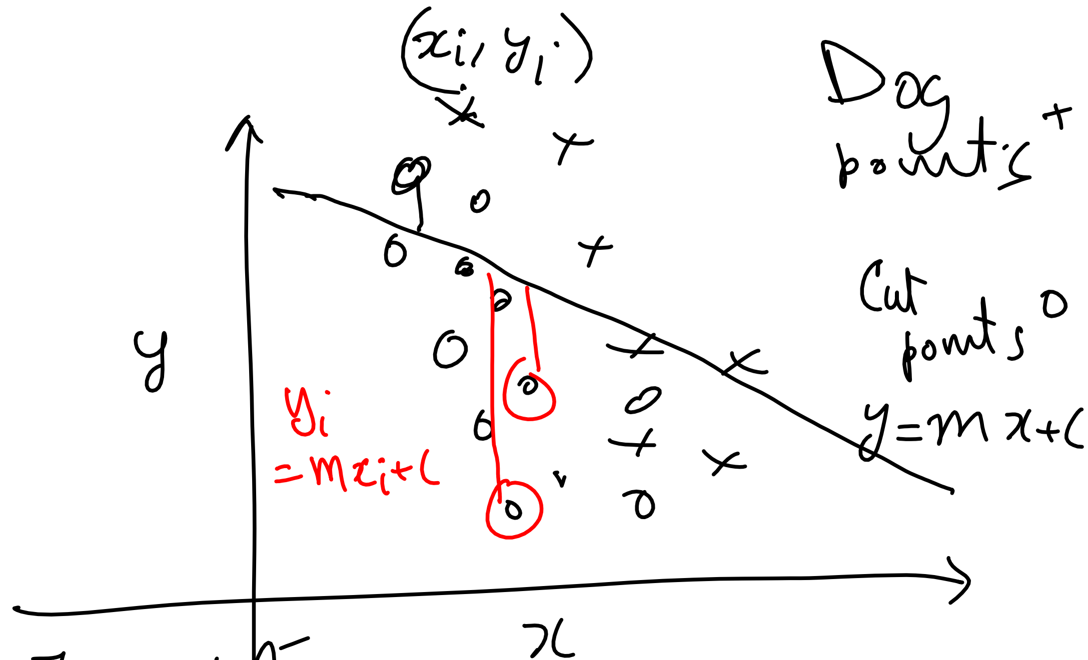
$l_i = -1$ if $0 > mx + c > -\infty$

$l \in \{-1, 1\} = \mathcal{Y}$ $\quad ||\mathcal{Y}|| = 2$

Optimization or minimization problem

① Random guess

$$\begin{bmatrix} m \\ c \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$



② Prediction according to the model

$$\hat{l}_i = \begin{cases} 1 & \text{if } y_i = mx_i + c > 0 \\ -1 & \text{if } y_i = mx_i + c < 0 \end{cases}$$

③ Comparison with training label

$D = \{(x_i, l_i), \ldots\}$

Error / Loss / cost / optimization

$$e(x_i, l_i) = \begin{cases} 0 & \text{if } l_i = \hat{l}_i \\ |(mx_i + c)| & l_i \neq \hat{l}_i \end{cases}$$

$$\boxed{\underset{m,c}{\arg\min} \sum_{i=1}^{n} e(x_i, l_i, m, c)}$$

$$l_i \in \{-1, 1\}$$

$$e(x_i, l_i) = \begin{cases} 0 & \text{if } l_i(mx_i+c) > 0 \\ -l_i(mx_i+c) & \text{if } l_i(mx_i+c) < 0 \end{cases}$$

$$\left.\begin{aligned} &(mx_i + c) > 0 \\ &\text{and} \quad l_i = +1 \end{aligned}\right.$$

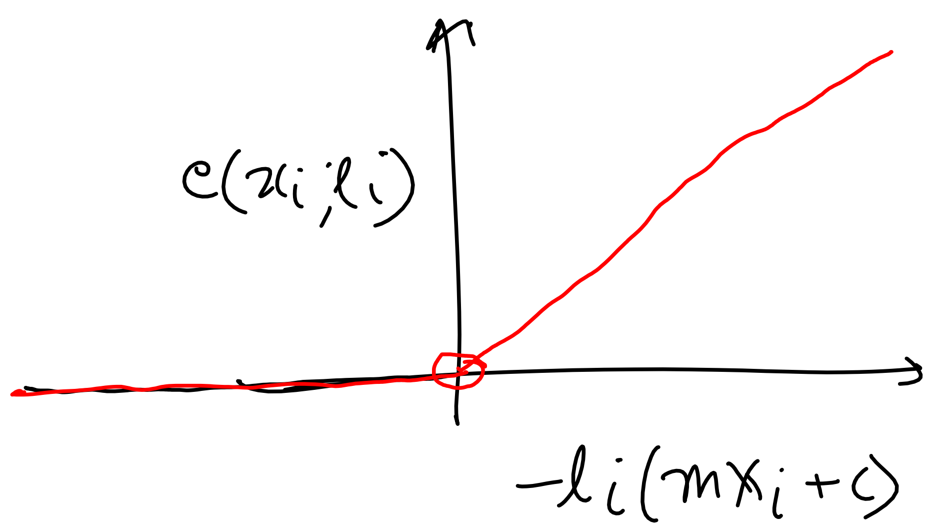
then $l_i(mx_i+c) > 0$

$$\left.\begin{aligned} &(mx_i + c) < 0 \\ &\text{and} \quad l_i = -1 \end{aligned}\right.$$

then $l_i(mx_i+c) > 0$

$$e(x_i, l_i) = \max\{0, \; -l_i(mx_i+c)\}$$

$$= \text{RELU}\{-l_i(mx_i+c)\}$$



$$\nabla_{m,c}\, e(x_i, l_i; m, c) = \nabla_{m,c} \max\{0, -l_i(mx_i + c)\}$$

$$= \nabla_{\underline{m}} \max\left\{0, -l_i\left([x_i \;\; 1]\underbrace{\begin{bmatrix} m \\ c \end{bmatrix}}_{\underline{m}}\right)\right\}$$

$$= \nabla_{\underline{m}} \max\{0, -l_i([x_i \;\; 1]\underline{m})\}$$

$$\underline{b}^T \underline{m}$$

$$= \max\{0, -l_i([x_i, 1])\}$$

$$\nabla_{\underline{m}}\, e\left(\underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}}_{\underline{x}}, \underbrace{\begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix}}_{\underline{l}}, \underline{m}\right) = \max\left\{\underline{0}, -\underbrace{\begin{bmatrix} l_1 \\ l_2 \\ \vdots \\ l_n \end{bmatrix}}_{\underline{l}} \odot \underbrace{\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \\ x_n & 1 \end{bmatrix}}_{X}\right\}$$

$$\nabla_{\underline{m}}\, e(\underline{x}, \underline{l}; \underline{m}) = \max\{\underline{0}, -\underline{l} \odot X\}$$

$$\underline{m}_t = \underline{m}_0 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

$t = 0$

while $\nabla_{\underline{m}} e(\underline{x}, \underline{l}, \underline{m}) > 0.001$:

$\quad m_t := m_t - \alpha_t \nabla_{\underline{m}} e(\underline{x}, \underline{l}, \underline{m})$

$\quad$ if $t > 1000$:

$\qquad$ break

$\quad t \mathrel{+}= 1$

optimal value of $m_t = m_t$

Gradient descent

Optimization for classification

$$y = mx + c$$

$$e(y_i, x_i; m, c) = \begin{cases} 0 & \text{if } mx_i + c = l_i \\ |mx_i + c| & \text{if } mx_i + c \neq l_i \end{cases}$$

$$e(y_i, x_i; m, c) = \begin{cases} 0 & \text{if } mx_i + c = l_i \\ |mx_i + c| & \text{if } mx_i + c \neq l_i \end{cases}$$

$$\mathbf{m} = \begin{bmatrix} m \\ c \end{bmatrix}$$

$$e(y_i, x_i; \mathbf{m}) = \begin{cases} 0 & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} = l_i \\ |\begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m}| & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} \neq l_i \end{cases}$$

$$\nabla_{\mathbf{m}} e(y_i, x_i; \mathbf{m}) = \begin{cases} 0 & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} = l_i \\ |\begin{bmatrix} x_i & 1 \end{bmatrix}| & \text{if } \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m} \neq l_i \end{cases}$$

If $l_i \in \{-1, 1\}$, then we can write

$$e(y_i, x_i; \mathbf{m}) = \max\{0, -l_i \begin{bmatrix} x_i & 1 \end{bmatrix} \mathbf{m}\}$$

$$\nabla_{\mathbf{m}} e(y_i, x_i; \mathbf{m}) = \max\{0, -l_i \begin{bmatrix} x_i & 1 \end{bmatrix}\}$$

$$\mu_x(I) = \sum_{x=1}^{W} \frac{xI(x,y)}{\sum_{x=1}^{W} I(x,y)}$$

$$\sigma_x^2(I) = \sum_{x=1}^{W} \frac{(x-\mu_x)^2 I(x,y)}{\sum_{x=1}^{W} I(x,y)}$$

In [ ]:
```
def error(X, Y, bfm):
    # YOUR CODE HERE
    raise NotImplementedError()

def grad_error(Xw, Yw, bfm):
    # YOUR CODE HERE
    raise NotImplementedError()

def train(X, Y, lr = 0.1):
    # YOUR CODE HERE
    raise NotImplementedError()

OPTIMAL_BFM, list_of_bfms, list_of_errors = train(X, Y)
fig, ax = plt.subplots()
ax.plot(list_of_errors)
ax.set_xlabel('t')
ax.set_ylabel('loss')
plt.show()
```

In [ ]:
```
positive_label = 1
negative_label = 0
TP = np.sum((zero_one_test_labels == positive_label) & (zero_one_predic
TP
```

In [ ]:
```
TN =  np.sum((zero_one_test_labels == negative_label) & (zero_one_predi
TN
```

In [ ]:
```
FP = np.sum((zero_one_test_labels != positive_label) & (zero_one_predic
FP
```

In [ ]:
```
FN = np.sum((zero_one_test_labels != negative_label) & (zero_one_predic
FN
```

In [ ]:
```
# Confusion matrix
fig, ax = plt.subplots()
ax.imshow([[TN, FN],
           [FP, TP]])
ax.set_xlabel('predicted')
ax.set_ylabel('true')
ax.axis('off')
```

Next

2. Show visualization of 1D optimization and loss functions.
3. Build to visualizations in the UDL book. Connect to KD tree and nearest neighbor classification.
4. Show the tensflow js visualization.

References

1. http://playground.tensorflow.org
2. https://knowyourdata-tfds.withgoogle.com/#tab=STATS&dataset=tf_flowers
3. "Flowers", The TensorFlow Team. Jan 2019. Online http://download.tensorflow.org/example_images/flower_photos.tgz