$\boxed{0}$ vs. $\boxed{1}$

```
n_dark = 0
for pixel in row:
    if pixel is dark:
        n_dark += 1
if n_dark > 1:
    return 0
else:
    return 1
```

0 1 2 3 4 5 6 7 8 9

Machine learning:

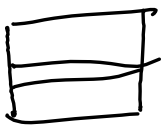
Program whose behavior is "learned" from data.

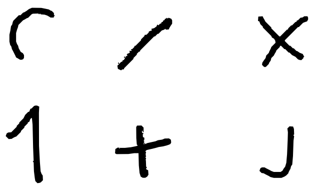
Old-fashioned ML:

1. Design features
2. Train a "simple" model

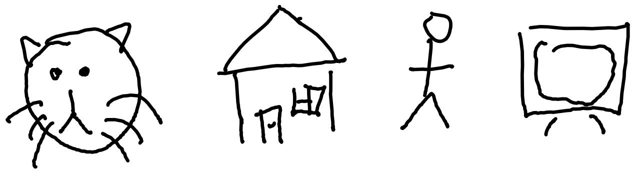
0 0 1 1 8 8 7 7

Feature: A measurement we can make about the input.

 # dark pixels

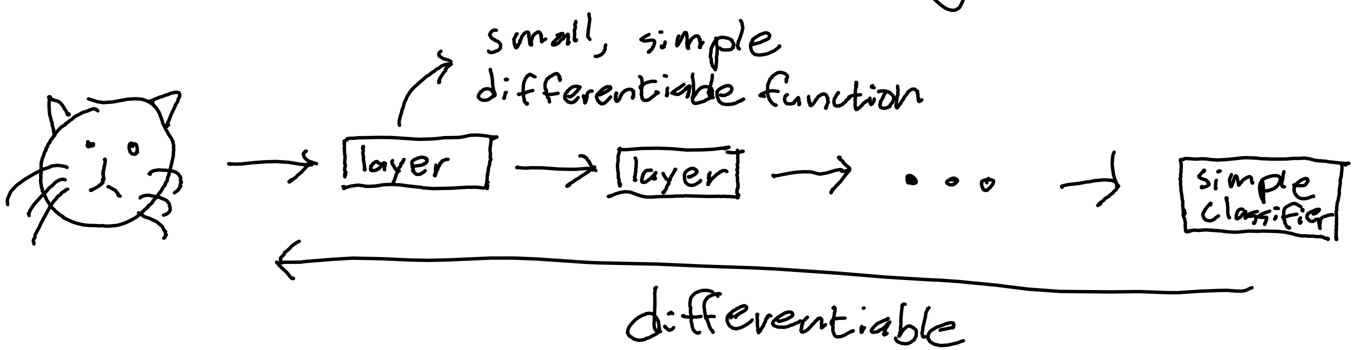


does the image have this shape?

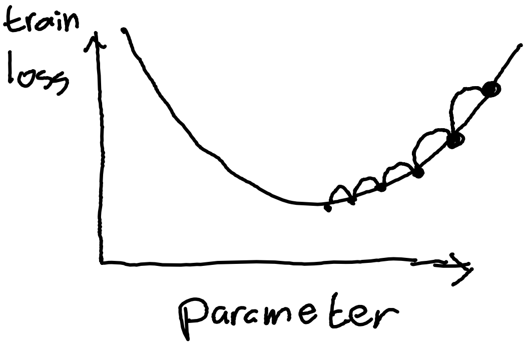


... (many classes)

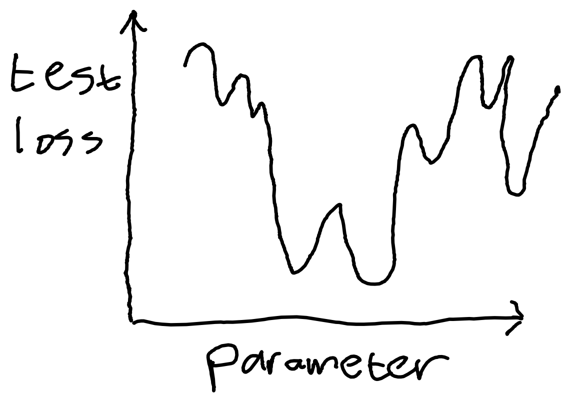
Deep learning: Learn features from data using big complex differentiable models that can be trained end-to-end with gradient descent.



Old-fashioned ML



Deep learning



regularization, inductive bias, pre-training...

Linear Regression

Linear model that predicts a scalar value from some inputs.

ex) Predict home value from

age sqft # bedrooms ...

$$\text{price} = \underbrace{w_1 \, \text{age} + w_2 \, \text{sqft} + w_3 \, \text{nbr} + b}_{\text{parameters}}$$

$$x = [\text{age}, \text{sqft}, \text{nbr}]$$

$$w = [w_1, w_2, w_3]$$

$$\hat{y} = \text{price}$$

$$\hat{y} = w^T x + b$$

$$\hat{y} = w^T x + b$$

$$\hat{Y} = Xw + b$$

$\hat{Y} \in \mathbb{R}^{ndata}$ $\quad X \in \mathbb{R}^{n\_data \times n\_features}$

Goal: fit the parameters (w, b) using the training data

How well are we doing?

How closely are predicted $\hat{y}$ to true values y in the training set?

Loss function:

$$L = \frac{1}{2}(\hat{y} - y)^2$$ squared error

Goal: Change parameters to minimize L over the training dataset.

solve? $\cancel{w^* = (X^T X)^{-1} X^T y}$

use gradient descent!

Gradient descent:

1. Initialize parameters Θ (w, b)
2. Repeat:

$$\underbrace{\Theta}_{\text{new param values}} \leftarrow \underbrace{\Theta}_{\text{old param values}} - \underbrace{\eta}_{\text{learning rate}} \underbrace{\nabla_\Theta L}_{\text{gradient of } L \text{ w.r.t. } \Theta}$$

until some criterion is met.

ex loss is zero. (rare)

ex loss stops decreasing.

ex performance on held-out data stops improving

ex Run out of compute or patience.

$\hat{y} = w^T x + b$ (x, y) is a single training datapoint

$L = \frac{1}{2}(\hat{y} - y)^2$

We need $\nabla_\theta L$ ($\nabla_w L$ and $\nabla_b L$)

Use the chain rule!

Functions: $L(\hat{y})$, $\hat{y}(w)$ (or $\hat{y}(b)$)

$$\frac{dL}{dw} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{dw}$$

$$\frac{dL}{db} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{db}$$

$$\hat{y} = w^T x + b$$

$$L = \frac{1}{2}(\hat{y} - y)^2$$

$$\frac{dL}{d\hat{y}} = \hat{y} - y$$

$$\frac{d\hat{y}}{dw} = x$$

$$\frac{d\hat{y}}{db} = 1$$

$$\frac{dL}{dw} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{dw} = (\hat{y} - y)x$$

$$\frac{dL}{db} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{db} = \hat{y} - y$$

$$w \leftarrow w - \eta \nabla_w L$$

SGD

use $\underline{1}$ example → stochastic gradient descent

"batch" X, a matrix of examples

$x \in \mathbb{R}^{\text{n-features}}$ $X \in \mathbb{R}^{\text{n-data} \times \text{n-features}}$

$$w \leftarrow w - \frac{\eta}{\text{n-data}} \sum_{\text{n-data}} \nabla_w L$$

"minibatch" SGD

↳ random subset of the training dataset

$$w \leftarrow w - \frac{\eta}{n_{data}} \sum_{n_{data}} \nabla_w L$$

Hyperparameters:

Values set "by hand" instead of learned

1. η learning rate
2. "batch size" (n_data)
3. stopping criterion
4. Initialization scheme

Softmax (Logistic) Regression

Classification: Predict one of k classes

- not ordered

"cat" $\ngtr$ "dog"

- might want class probabilities

<u>ex</u> 93% cat
7% dog

ex Predict whether a home will sell above, below, or not from age, sqft, nbr, price

"one-hot" vectors

$[1, 0, 0]$ below 1

$[0, 1, 0]$ above 2

$[0, 0, 1]$ not 3

Model predicts $k = 3$ scores

larger score: "I think its this class"

$$\underset{\text{score}}{O_1} = W_{1,1} x_1 + W_{1,2} x_2 + \dots + b_1$$

class 1

$$O = Wx + b \quad W \in \mathbb{R}^{3 \times 4},\ b \in \mathbb{R}^3,\ x \in \mathbb{R}^4,\ o \in \mathbb{R}^3$$

have unnormalized scores
want probabilities:

1. non-negative
2. Sum to one

$\hat{y} = \mathrm{softmax}(o) \quad o \in \mathbb{R}^k \quad \hat{y} \in \mathbb{R}^k$

$$\mathrm{softmax}(o)_j = \frac{\exp(o_j)}{\sum_k \exp(o_k)}$$

1. $\exp(\ldots) \geq 0$
2. $\sum_j \mathrm{softmax}(o)_j = \frac{\sum_j \exp(o_j)}{\sum_k \exp(o_k)} = 1$

Softmax regression

$$p(\underbrace{y}_{\text{class label}} \mid \underbrace{x}_{\text{input}}) = \text{softmax}(\underbrace{Wx + b}_{\text{params}})$$

"linear" because the scores o are a linear function of the input.

Want to find parameters that maximize $p(y|x)$ over our training data

↳ ground-truth class

$$\prod_i p(y^{(i)}|x^{(i)}) \qquad (x^{(i)}, y^{(i)} \; i^{th} \text{ training example})$$

$$\sum_i \log p(y^{(i)}|x^{(i)}) \qquad (\text{because log monotonic})$$

$$-\sum_i \log p(y^{(i)}|x^{(i)}) \qquad (\text{because we minimize})$$

$$-\sum_i \sum_j x_j^{(i)} \log \operatorname{softmax}(Wx^{(i)} + b)_j$$

"cross-entropy"

Measures how "wrong" a probability distribution is

$$-\sum_i \sum_j y_j^{(i)} \log \hat{y}_j^{(i)}$$

$$\hat{y}_j^{(i)} = \text{softmax}(Wx^{(i)} + b)_j$$

For one particular example

$$L(y, \hat{y}) = -\sum_j y_j \log \frac{\exp(o_j)}{\sum_k \exp(o_k)}$$

$$= \sum_j y_j \log \sum_k \exp(o_k) - \sum_j y_j o_j$$

$$= \log \sum_k \exp(o_k) - \sum_j y_j o_j$$

$$\frac{dL(y,\hat{y})}{do_j}$$ only new term in chain rule

$$d_{o_j}[L(y,\hat{y})] = d_{o_j}[\log \Sigma_k \exp(o_k) - \Sigma_j y_j o_j]$$

$$= \frac{d_{o_j} \Sigma_k \exp(o_k)}{\Sigma_k \exp(o_k)} - y_j$$

$$= \frac{\exp(o_j)}{\Sigma_k \exp(o_k)} - y_j$$

$$= \text{softmax}(o)_j - y_j$$