Intro to Deep Learning

Lecture 2:
Machine Learning Fundamentals

Before getting into deep learning…

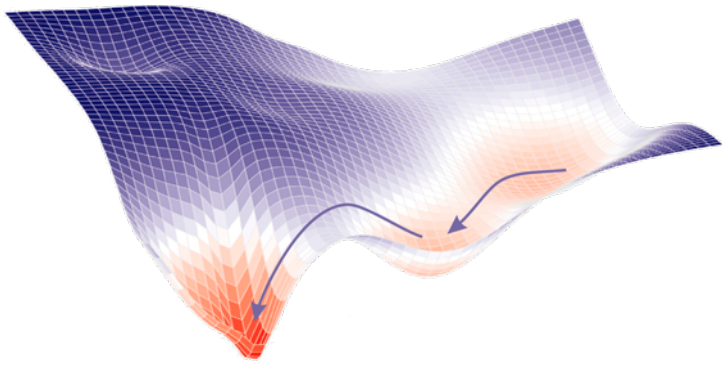
Before getting into deep learning…

Let us recall ML basics...

Course Topics

Machine Learning

Optimization



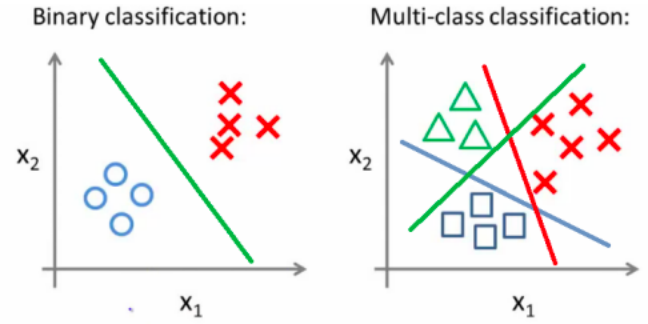
Example: image classification (multi-class)

~15 million data points across ~22,000 classes

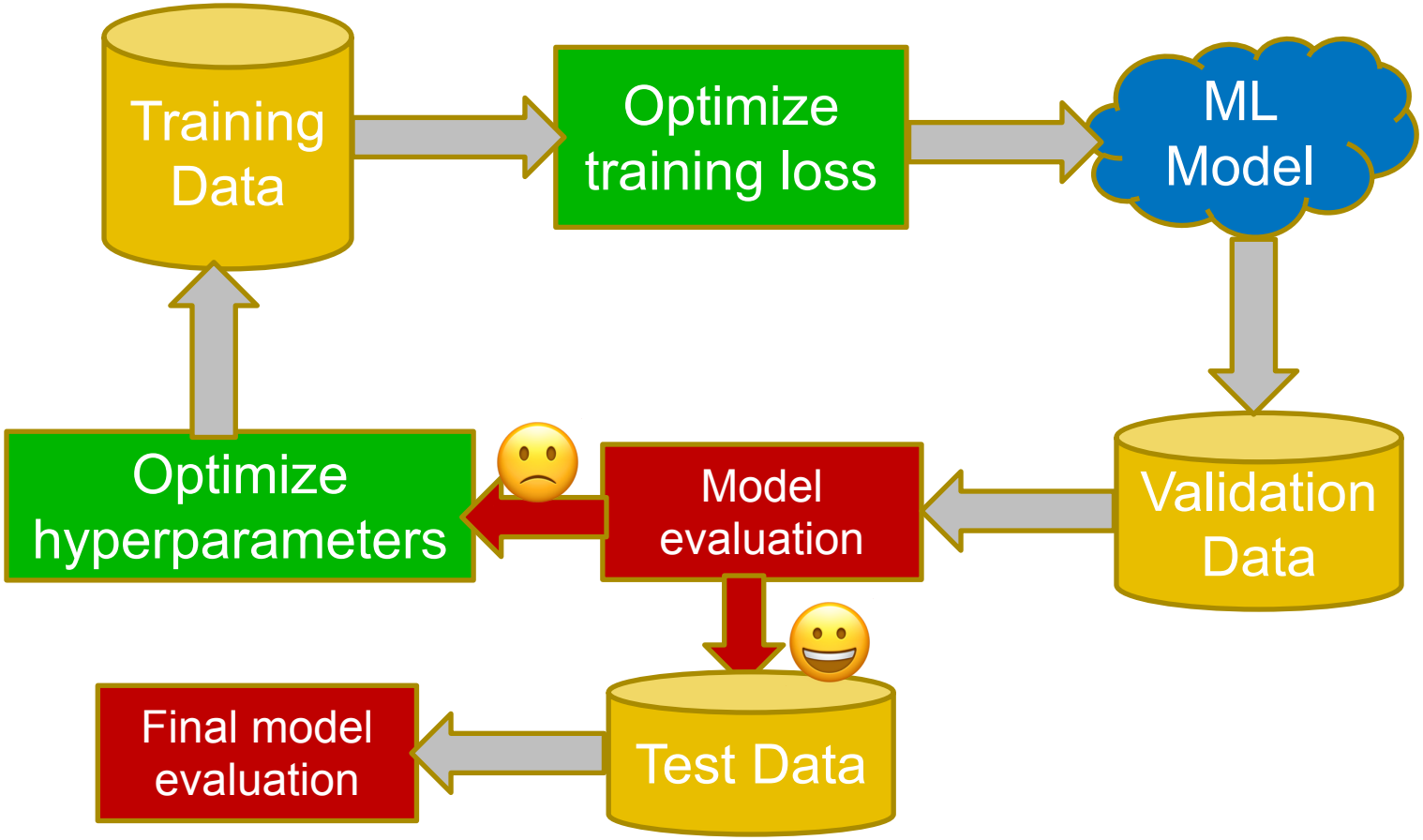
ImageNet figure from vision.standford.edu

A few terminologies

- **Training data:** images given for learning
- **Validation/Test data:** images to evaluate models
- **Binary classification:** classify into two classes
- **Muticlass classification:** classify into >2 classes

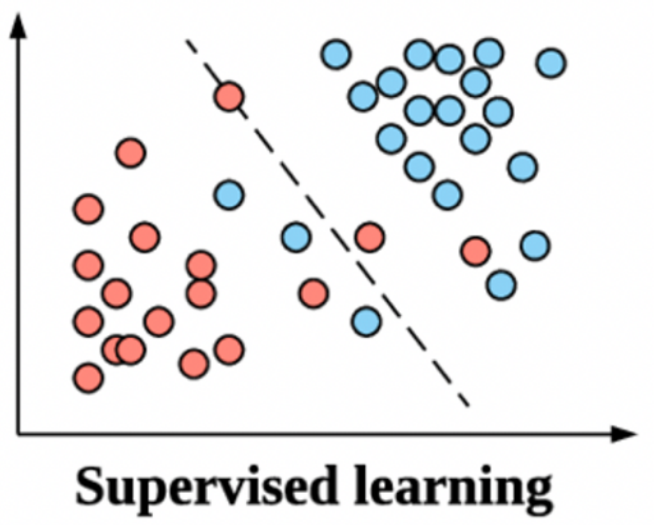


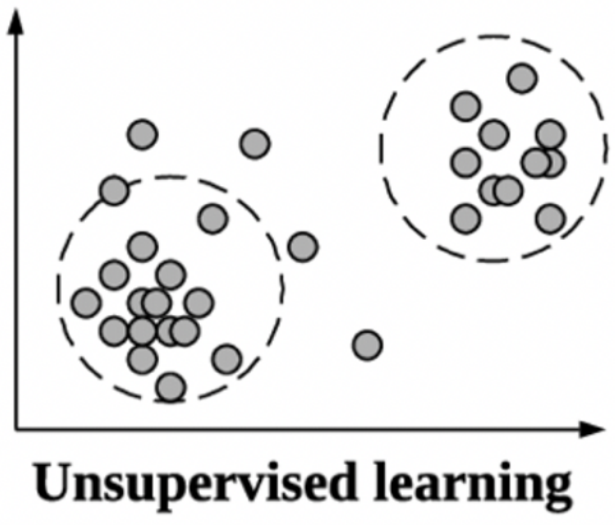
Machine Learning Pipeline



Ingredient 1: Data

Data



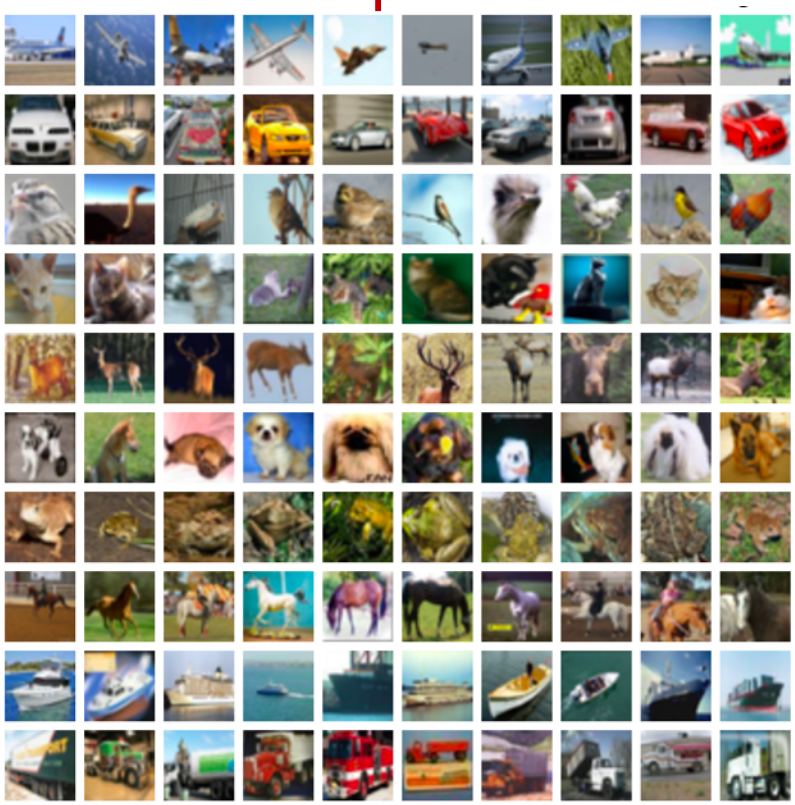


Data

Labels

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck

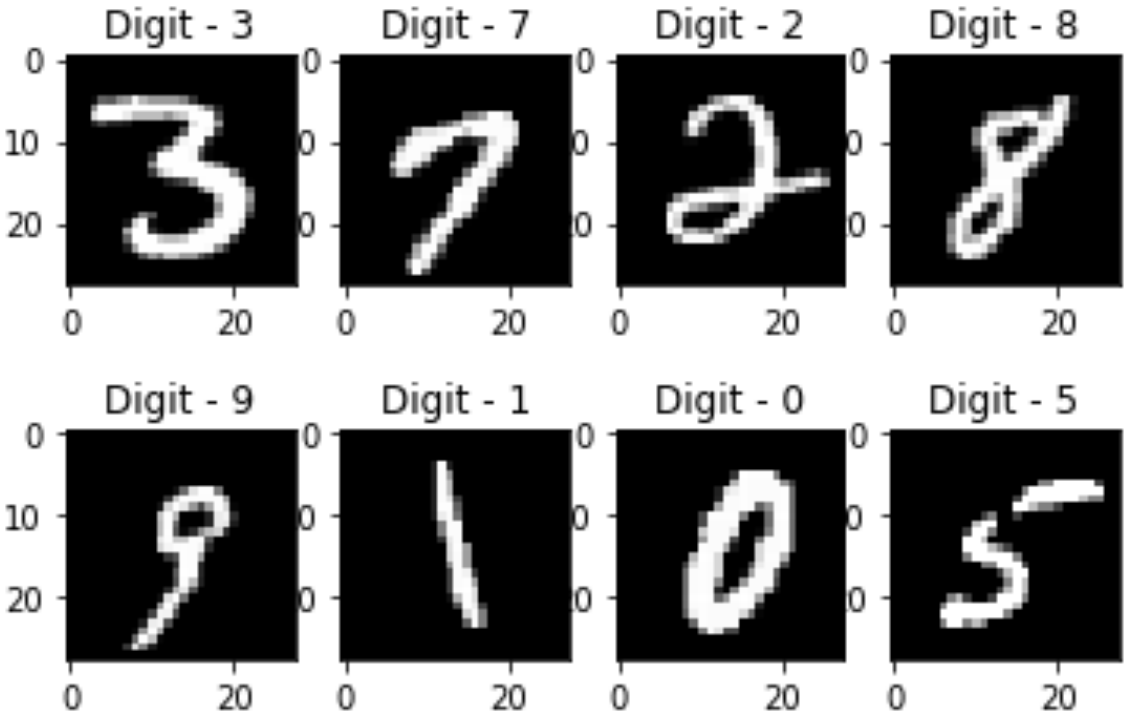
Inputs



Inputs are
32×32 color images

Notation: A dataset of size n is denoted as $(x_i, y_i)_{i=1}^n$

MNIST data


Digit - 3
Digit - 7
Digit - 2
Digit - 8
Digit - 9
Digit - 1
Digit - 0
Digit - 5
0
10
20

MNIST data

Input dimension
$28 \times 28 = 784$

```
digit 7
pixls 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 84 185 159 151 60 36 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 222
254 254 254 254 241 198 198 198 198 198 198 198 198 170 52 0 0 0 0 0 0 0 0 0 0 0 0 67 114 72 114 163 227 254 225 254 254 254 250 229 254 254 140 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 17 66 14 67 67 67 59 21 236 254 106 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 83 253 209 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 22 233 255 83 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 129 254 238 44 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 59 249 254 62 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 133 254 187 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9 205 248 58 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 126 254
182 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 75 251 240 57 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 221 254 166 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 3 203 254 219 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 254 254 77 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 224 254 115 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 133 254 254 52 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 61 242 254 254 52 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 121 254 254 219 40 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 121 254 207 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0
```

```
digit 2
pixls 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 116 125 171 255 255 150 93 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 169 253 253 253 253 253 253 218 30 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 169 253 253 253 213 142 176 253 253 122 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 52 250 253 210 32 12 0 6 206 253 140 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 77
251 210 25 0 0 0 122 248 253 65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 18 0 0 0 0 209 253 253 65 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 117 247 253 198
10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 76 247 253 231 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 128 253 253 144 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 176 246 253 159 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 234 253 233 35 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 198 253 253 141 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 78 248 253 189 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 19 200 253 253 141 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 134 253 253 173 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 248 253 253 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 248 253 253 43 20
20 20 20 5 0 5 20 20 37 150 150 150 147 10 0 0 0 0 0 0 0 0 0 248 253 253 253 253 253 253 253 168 143 166 253 253 253 253 253 253 253 123 0 0 0 0 0 0 0 0 0 174
253 253 253 253 253 253 253 253 253 253 253 249 247 247 169 117 117 57 0 0 0 0 0 0 0 0 0 0 118 123 123 123 166 253 253 253 155 123 123 41 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
digit 1
pixls 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 38 254 109 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 87 252 82 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 135 241 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 45 244 150 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 84 254 63 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 202 223 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 32 254 216 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 95 254 195 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 140 254 77 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 57 237 205 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 124 255 165 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 171 254 81 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 24 232 215 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 120 254 159 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 151 254
142 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 228 254 66 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 61 251 254 66 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 141 254 205 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 215 254 121 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 198 176 10 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

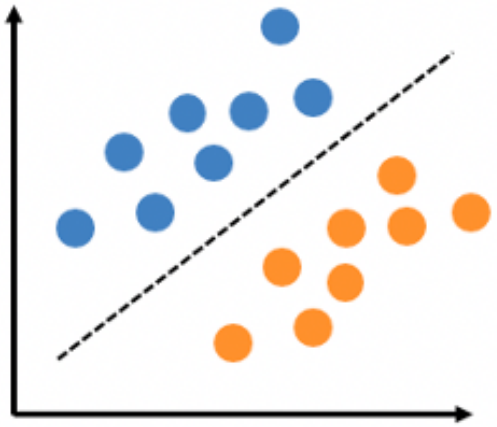
Machine Learning Model

An **ML Model** is a function that outputs a prediction for a given input.

Ex: Linear classifier

Let $\hat{y}_f$ be the label output by f

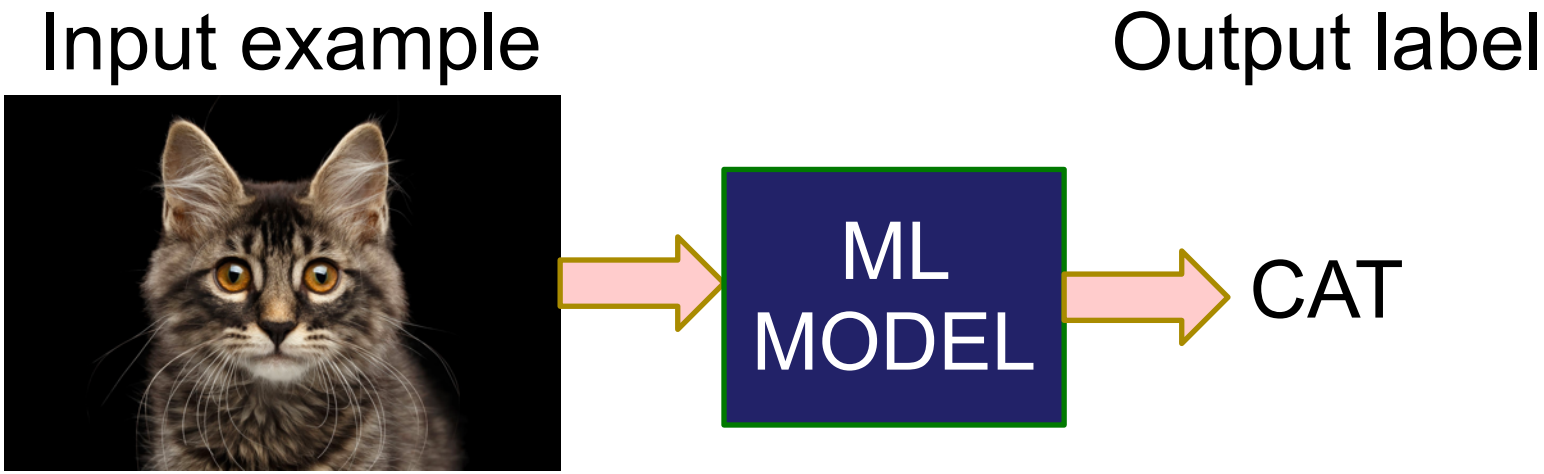
- $\hat{y}_f = 1$ if $f(x) \geq 0$
- $\hat{y}_f = -1$ else



In other words, $\hat{y}_f = \text{sign}(f(x))$

Goal (in supervised ML): "Build a model that can predict the label of the input examples."

Goal (in supervised ML): "Build a model that can predict the label of the input examples."



Goal (in supervised ML): "Build a model that can predict the label of the input examples."

- Given training data $(x_i, y_i)_{i=1}^n$
- Find $f : x \mapsto y$ using training data
- Such that f is (approximately) correct on **unseen** test data

Ingredient 2: Machine Learning Model

An **ML Model** is a function f that outputs a prediction for a given input.

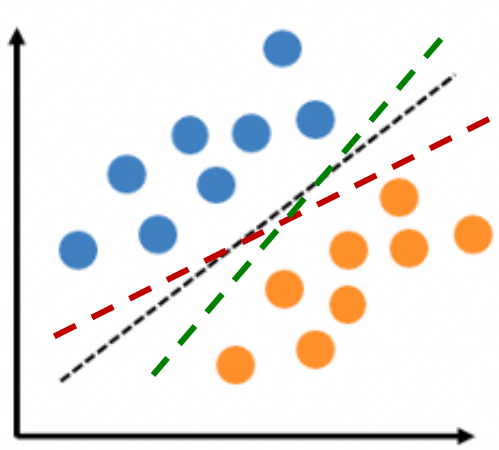
Ex: Input example: $x \in \mathbb{R}^d$

A linear model with weights $w \in \mathbb{R}^d$ outputs

$$f_\theta(x) := \langle w, x \rangle = w^\top x \in \mathbb{R}$$

Ingredient 3: Loss function

There are many ML models



Q: How do you find a good ML model?

A: Minimize a loss function ℓ

Loss function

ℓ takes a prediction $f(x)$ and a label y as input

- **Ex: Squared loss**

$$\ell\big(f(x), y\big) = \big(f(x) - y\big)^2$$

- **Ex: Cross-entropy loss (multi-class)**

$$\ell\big(f(x), y\big) = -\sum_{i \in [C]} y_i \log(p_i), \quad \text{where} \quad p_i = \frac{e^{f_i(x)}}{\sum_{j=1}^{n} e^{f_j(x)}}$$

- **Ex: 0/1 loss** (0-1 loss)

$$\ell\big(f(x), y\big) = \mathbb{I}\big(\text{sign}(f(x)) \neq y\big)$$

1 if $y \neq \text{sign}(f(x))$

0 if $y = \text{sign}(f(x))$

More examples to come...

Finding the best model

- **Statistical learning:** Data is generated from a distribution

$$(x, y) \sim \mathscr{D}$$

- The model f has the performance

$$\mathscr{L}(f) = \mathbb{E}_{\mathscr{D}}\left[\ell(f(x), y)\right]$$

Observation: $\mathbb{E}_{\mathscr{D}}\left[\mathbb{I}\left(y \neq \operatorname{sign}(f(x))\right)\right] = \mathbb{P}\left(y \neq \operatorname{sign}(f(x))\right)$

- We can find best model from a hypothesis set $\mathscr{F}$ via

$$f^\star = \arg\min_{f \in \mathscr{F}} \mathscr{L}(f)$$

Hypothesis sets

- The model f has the performance

$$\mathscr{L}(f) = \mathbb{E}_{\mathscr{D}}\left[\ell(f(x), y)\right]$$

Observation: $\mathbb{E}_{\mathscr{D}}\left[\mathbb{I}\left(y \neq \operatorname{sign}(f(x))\right)\right] = \mathbb{P}\left(y \neq \operatorname{sign}(f(x))\right)$

- We can find best model from a hypothesis set $\mathscr{F}$ via

$$f^\star = \arg\min_{f \in \mathscr{F}} \mathscr{L}(f)$$

- Example hypothesis sets
  - Linear classifiers
  - Neural networks
  - Decision trees
  - ...

Training Loss

Our aim is typically finding an accurate model i.e.

$$f^\star = \arg\min_{f \ in \mathcal{F}} \mathscr{L}(f) \quad \text{where} \quad \mathscr{L}(f) = \mathbb{E}_{\mathscr{D}}\left[\ell(f(x), y)\right]$$

- **Challenge 1:** We don't have $\mathscr{D}$. Instead, we only have training data sampled from $\mathscr{D}$

$$\left(x_i, y_i\right)_{i=1}^n \sim \mathscr{D}$$

- **Solution:** Minimize the training loss to approximate $\mathscr{L}(f)$

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \widehat{\mathscr{L}}(f) \quad \text{where} \quad \widehat{\mathscr{L}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x), y)$$

Hopefully $\hat{f}$ will be a good proxy for $f^\star$! Rigorous proof via concentration inequalities!

Machine Learning Pipeline

1. Data: Collect training dataset
2. Model: Fix hypothesis set $\mathscr{F}$ and loss ℓ
3. Optimize: Minimize the training loss

Example ML Pipeline

1. Data: Collect $(x_i, y_i)_{i=1}^n$

2. Model: Choose a neural network architecture and pick cross-entropy loss

3. Optimize

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \widehat{\mathscr{L}}(f) \quad \text{where} \quad \widehat{\mathscr{L}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x), y)$$

Optimization Challenge

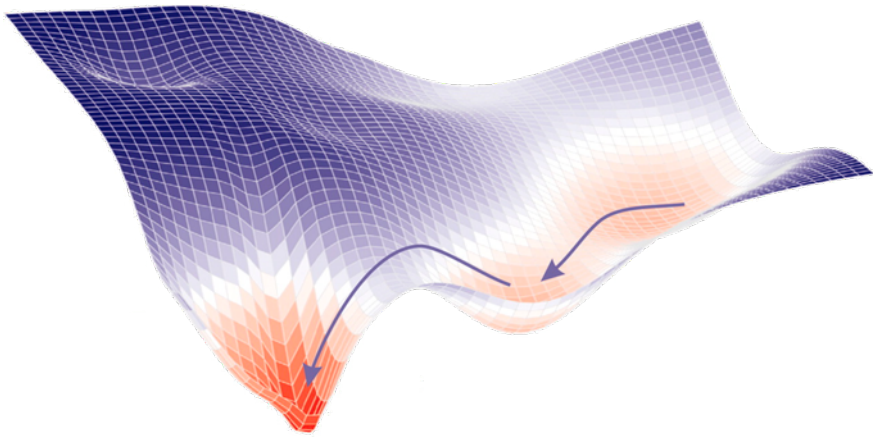
Our aim is typically finding an accurate model i.e.

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \widehat{\mathscr{L}}(f) \quad \text{where} \quad \widehat{\mathscr{L}}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x), y)$$

- Challenge 2: How do you solve $\min_{f \in \mathcal{F}} \widehat{\mathscr{L}}(f)$?

Optimization

Goal: $\min_{f \in \mathcal{F}} \mathcal{L}(f)$



😀 Some problems admit closed form solution (e.g. linear regression)

😕 Modern ML problems are highly non-convex and non-linear.

Solution: **Gradient descent**

1. Workhorse of modern optimization
2. Optimize iteratively: Starting from an initial model f_0

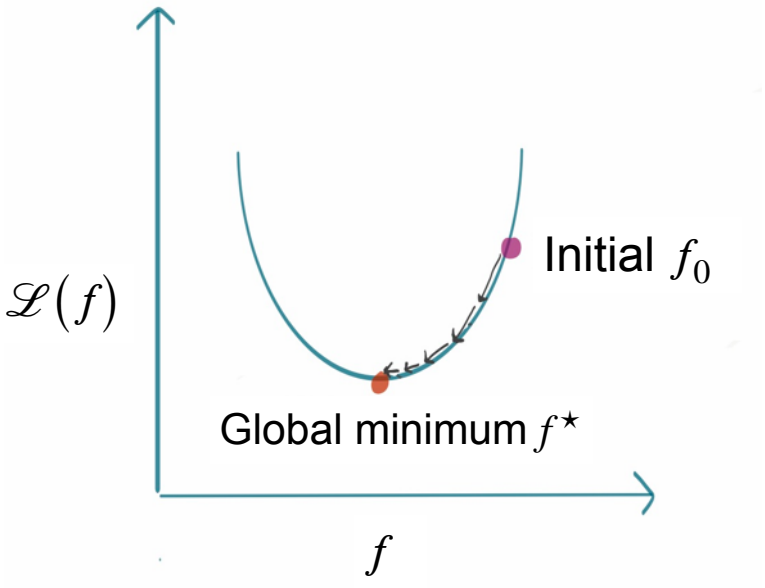
$$f_{i+1} = f_i - \eta \nabla \mathcal{L}(f_i)$$

Optimization

- Gradient descent iterations

$$f_{i+1} = f_i - \eta \nabla \mathscr{L}(f_i)$$

New model | Current model | Learning rate | Gradient

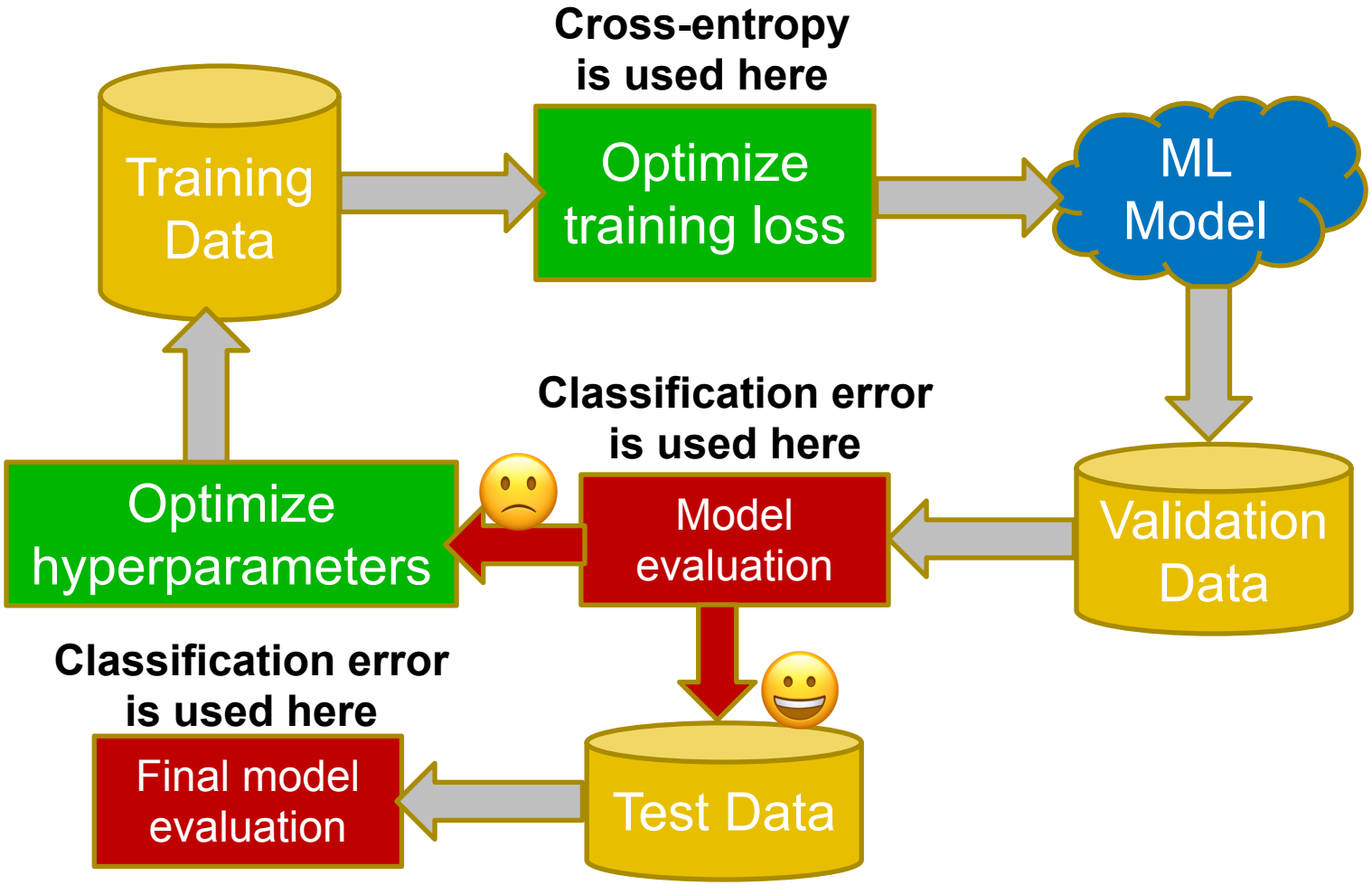


Back to loss functions

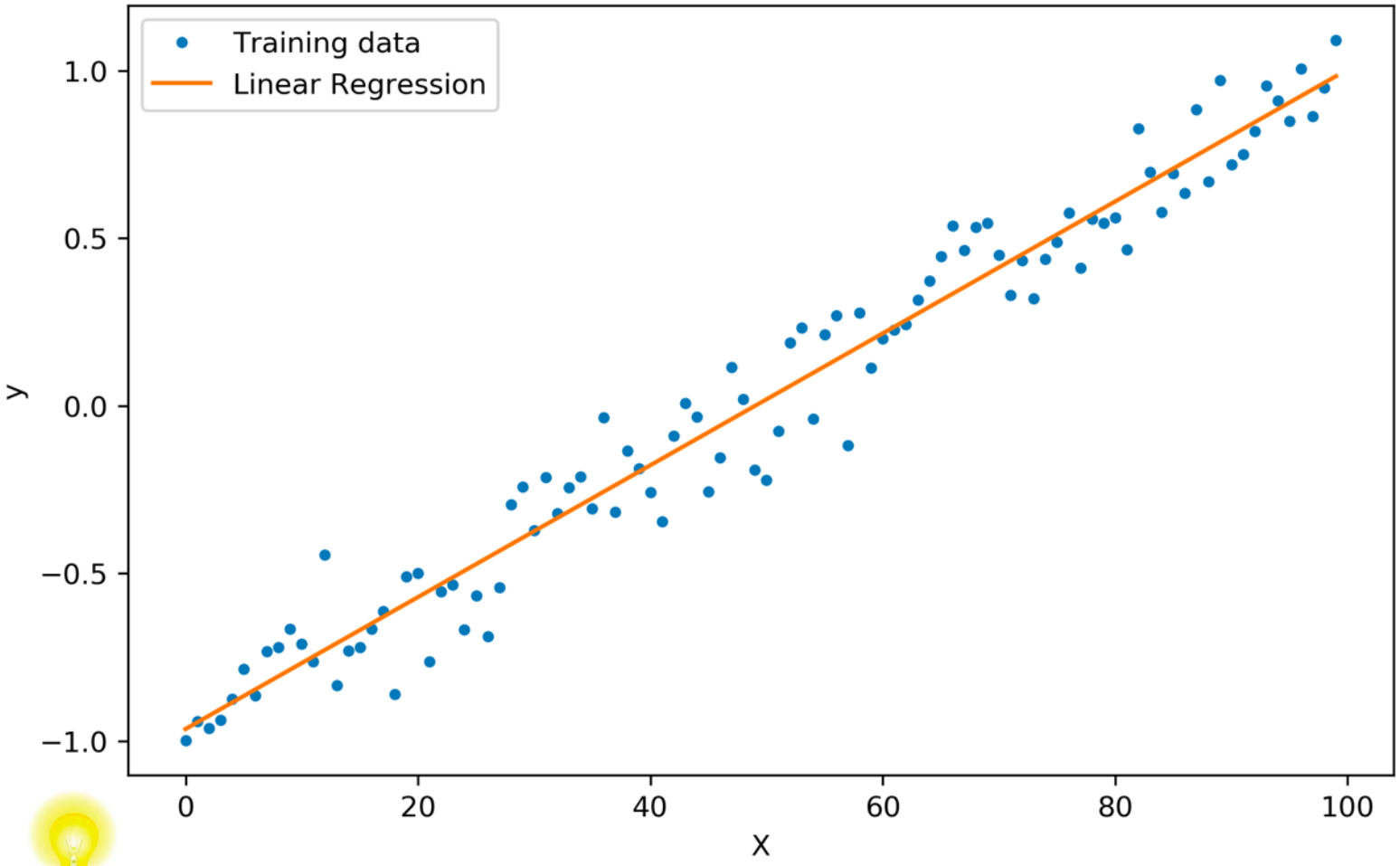
- Gradient descent requires gradient calculation
- Classification error $\mathbb{I}(\text{sign}(f(x)) \neq y)$ is not differentiable.
- Idea: Use a surrogate loss function instead
  1. Cross-entropy (logistic loss)
  2. Squared-loss
  3. ...
- In short: Training performance metric (training loss) may differ from test performance metric (test error).

Break

ML Pipeline


Cross-entropy is used here
Training Data
Optimize training loss
ML Model
Classification error is used here
Optimize hyperparameters
Model evaluation
Validation Data
Classification error is used here
Final model evaluation
Test Data

Optimization Fundamentals



Let us understand linear regression before deep learning.

Linear regression

1. Data: $\left(x_i, y_i\right)_{i=1}^n$, $x_i \in \mathbb{R}^d, y_i \in \mathbb{R}$
2. Model: Linear model $f(x; w) = x^\top w$
3. Squared Loss: $\ell(f(x), y) = \frac{1}{2}\left(f(x) - y\right)^2$
4. Optimize

$$\widehat{w} = \arg\min_{w \in \mathbb{R}^d} \widehat{\mathscr{L}}(w) \quad \text{where} \quad \widehat{\mathscr{L}}(w) = \frac{1}{2n} \sum_{i=1}^n (x_i^\top w - y_i)^2$$

Linear regression

- Let us move to matrix notation

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \quad n \times d \text{ matrix} \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \quad n \times 1 \text{ vector}$$

- The loss function becomes

$$\widehat{\mathscr{L}}(w) = \frac{1}{2n} \sum_{i=1}^{n} (x_i^\top w - y_i)^2 = \frac{1}{2n} \left\| y - Xw \right\|_2^2$$

Linear regression

Optimization becomes

$$w^\star = \arg\min_{w \in \mathbb{R}^d} \frac{1}{2} \left\| y - Xw \right\|_2^2$$

Solution can be found by differentiation.

Convex analysis

- Set the gradient to zero: $\nabla \mathscr{L}(w) = 0$

$$\nabla \mathscr{L}(w) = X^\top X w - X^\top y$$

- Solution $w^\star$ satisfies: $X^\top X w^\star = X^\top y$

The normal equation

Linear regression

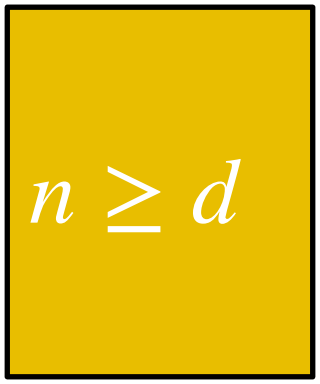
Solution satisfies: $X^\top X w^\star = X^\top y$

- If $X^\top X$ is invertible (which requires $n \geq d$),
  - Unique solution $w^\star = \left(X^\top X\right)^{-1} X^\top y$
  - **Intuition:** More equations than unknowns!
- Otherwise, infinitely many solutions.
  - Which solution to pick?
  - More on this later

Pseudo-inverse

A general solution: $w^\star = X^\dagger y$ where $X^\dagger$ is pseudo-inverse

- $X^\dagger = \left(X^\top X\right)^{-1} X^\top$ if columns of X are linearly independent



Over-determined problem

- $X^\dagger = X^\top \left(XX^\top\right)^{-1}$ if rows are linearly independent

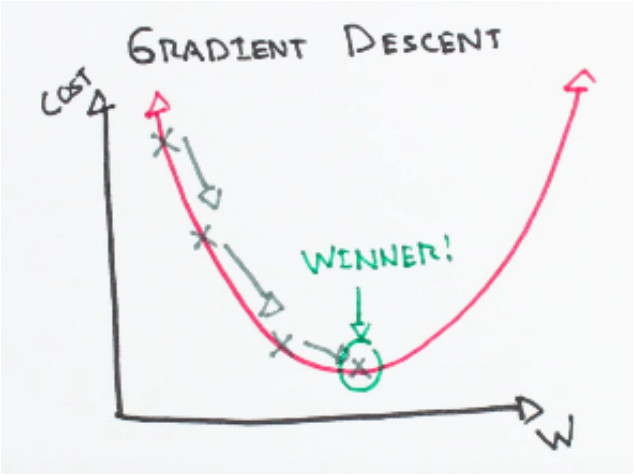
Over-parameterized problem

Linear regression

- Pseudo-inverse is simple and closed-form

- But matrix inversion is an expensive operation
  - $w^\star = \left(X^\top X\right)^{-1} X^\top y$ requires $O(d^3 + dn)$ operations naively

    Matrix inversion: $O(d^3)$ Matrix-vector multiplication: $O(nd)$
- How to solve linear regression without inversion?



Linear regression

- Recall the gradient:

$$\nabla \mathscr{L}(w) = X^\top X w - X^\top y$$

  - Fix some initial weights w_0
  - Fix a learning rate $\eta > 0$

- Gradient iterations are given by

$$w_{t+1} = w_t - \eta \nabla \mathscr{L}(w_t)$$

Linear regression

- Gradient iterations are given by

$$w_{t+1} = w_t - \eta \nabla \mathscr{L}(w_t)$$

- Plugging in the gradient formula

$$w_{t+1} = w_t - \eta \left(X^\top X w_t - X^\top y \right)$$

- Computational cost of each iteration

$$O(nd)$$

Only need matrix-vector multiplication

Linear regression

Plugging in the gradient formula

$$w_{t+1} = w_t - \eta\left(X^\top X w_t - X^\top y\right)$$

Questions:

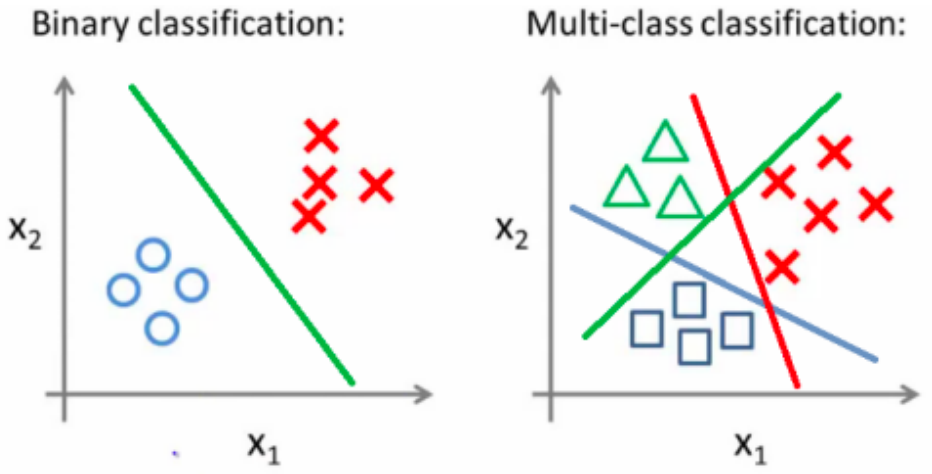
- How many iterations needed?
- Which learning rate?
- Can we speed up further?

To be discussed in future lectures

Answers are important for **deep learning** as well.

Multiclass Classification

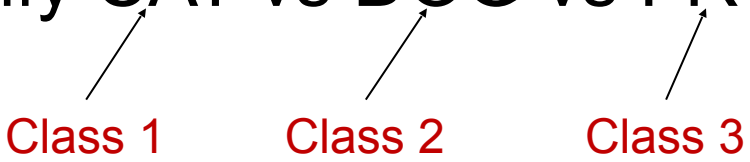
- So far, we assumed label is a scalar
- Modern ML models can classify 100s of objects



- How do we handle multiple classes?

One-hot encoding

- For a K-class problem, use K dimensional labels
- Ex: Classify CAT vs DOG vs FROG.

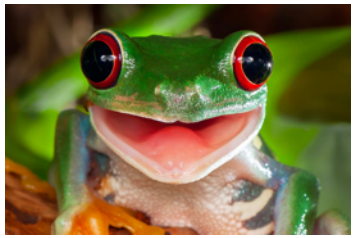


  - CAT receives label $[1\,0\,0]$
  - DOG receives label $[0\,1\,0]$
  - FROG receives label $[0\,0\,1]$

Example Dataset

Inputs x_i

Labels y_i



$[0\,0\,1]$

$[0\,1\,0]$

$[1\,0\,0]$

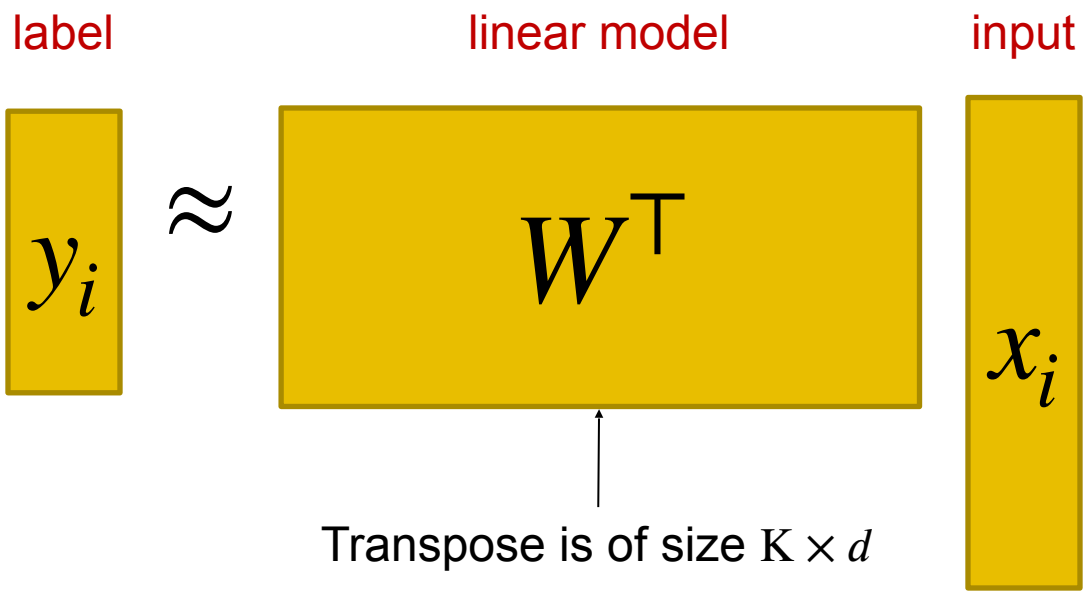
$[0\,1\,0]$

Multiclass Classifier

- Suppose $x \in \mathbb{R}^d$, $y \in \mathbb{R}^K$
- Need a ML model $f : \mathbb{R}^d \to \mathbb{R}^K$
- What does this mean for linear model?

Multiclass Linear Model

- Multiclass linear model
  - a matrix W rather than vector w
  - dimensions $d \times K$



Multiclass Optimization

- Back to linear regression with squared-loss

$$\widehat{\mathscr{L}}(W) = \frac{1}{2}\sum_{i=1}^{n} \left\| y_i - W^\top x_i \right\|_2^2$$

The L_2 distance between label vector and prediction

Multiclass Optimization

- Back to linear regression with squared-loss

$$\widehat{\mathscr{L}}(W) = \frac{1}{2}\sum_{i=1}^{n} \left\| y_i - W^\top x_i \right\|_2^2$$

$$= \frac{1}{2}\left\| Y - XW \right\|_F^2$$

$$\left\| Z \right\|_F = \sqrt{\sum_{i,j} Z_{i,j}^2}$$

Multiclass Optimization

- Back to linear regression with squared-loss

$$\widehat{\mathscr{L}}(W) = \frac{1}{2}\sum_{i=1}^{n} \left\| y_i - W^\top x_i \right\|_2^2 = \frac{1}{2}\left\| Y - XW \right\|_F^2$$

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \quad n \times d \text{ matrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \quad n \times K \text{ matrix}$$

Optimizing with Gradient Descent

- Gradient is given by

$$\nabla \mathcal{L}(W) = X^T(XW - Y)$$

- Plugging in the gradient formula

$$W_{t+1} = W_t - \eta\left(X^T X W_t - X^T Y\right)$$

- Computational cost of each iteration

$$O(K \times Nd)$$

- The equations remain the same! 😀

- To be discussed: **cross-entropy loss**

Summary

1. Machine learning fundamentals

2. Gradient descent basics
   - Linear regression

3. Multiclass models & one-hot encoding

4. Next class: Getting started with neural nets