CHAPTER 1

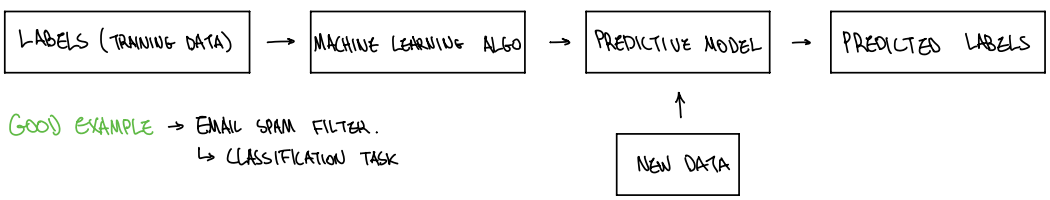
THIS DAY AND AGE → ABUNDANCE OF DATA (STRUCTURED + UNSTRUCTURED)

THREE TYPES OF MACHINE LEARNING

- SUPERVISED LEARNING → LABELLED DATA, DIRECT FEEDBACK, PREDICT OUTCOME / FUTURE.
- UNSUPERVISED LEARNING → UNLABELLED DATA, NO FEEDBACK, FIND HIDDEN STRUCTURE IN DATA.
- REINFORCEMENT LEARNING → DECISION PROCESS, REWARD SYSTEM, LEARN SERIES OF ACTIONS.

SUPERVISED LEARNING

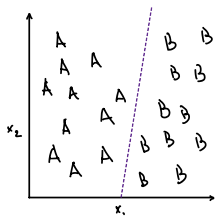
MODEL LEARNS FROM LABELLED TRAINING DATA → MAKE FUTURE PREDICTIONS. → "LABELLED LEARNING"



GOOD EXAMPLE → EMAIL SPAM FILTER.
↳ CLASSIFICATION TASK

CLASSIFICATION

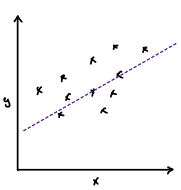
- SUB-CATEGORY OF SUPERVISED LEARNING
- AIM → PREDICT CATEGORIAL CLASS LABELS OF NEW INSTANCES, BASED ON PAST EXAMPLES.
- EMAIL SPAM FILTER → BINARY CLASSIFICATION TASK → SPAM OR NOT SPAM.
- SUPERVISED LEARNING TRAINED MODEL → ASSIGN ANY CLASS LABEL TO NEW DATA
  ↳ MULTICLASS CLASSIFICATION
  - E.G. → HANDWRITTEN CHARACTER RECOGNITION → "A", "B", "C", ...
    ↳ MODEL WASN'T TRAINED ON NUMBERS → WOULDN'T RECOGNISE "0-9"



REGRESSION

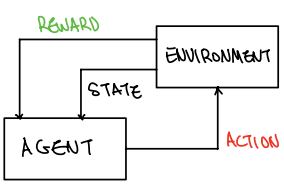
↱ NAME OF VARIABLES IN ML

- FINDING RELATIONSHIP BETWEEN VARIABLES (FEATURES, TARGET VARIABLES) TO PREDICT AN OUTCOME.
- E.G. → FINDING CORRELATION BETWEEN TIME SPENT STUDYING VS. EXAM RESULTS → PREDICT FUTURE SCORES.



REINFORCEMENT LEARNING

- DEVELOP AGENT → IMPROVES PERFORMANCE BASED ON INTERACTION WITH ENV.
- TRIAL-AND-ERROR APPROACH → REWARDS
- E.G. → CHESS PROGRAM → WIN OR LOSE WITH DIFFERENT MOVES
- CHOOSE A SERIES OF ACTIONS TO MAXIMISE REWARDS
  ↳ IMMEDIATE OR DELAYED FEEDBACK

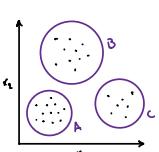


UNSUPERVISED LEARNING

- UNLABELLED DATA ↴
- EXPLORE STRUCTURE OF DATA TO EXTRACT DATA WITHOUT GUIDANCE FROM KNOWN OUTCOME VARIABLES.

CLUSTERING

- ORGANISE INFORMATION INTO SUBGROUPS (CLUSTERS) → WITHOUT PRIOR INFORMATION ABOUT GROUP.
- SOMETIMES CALLED "UNSUPERVISED CLASSIFICATION"
- GOOD TECHNIQUE FOR STRUCTURING INFORMATION + FINDING RELATIONSHIPS FROM DATA.



DIMENSIONAL REDUCTION

- OFTEN WE WORK WITH HIGH DIMENSIONALITY - LARGER NUMBER OF FEATURES.
- COMPRESSES DATA ONTO SMALLER DIMENSIONAL SUBSPACE WHILE RETAINING MOST OF THE RELEVANT INFORMATION.
- COMMONLY USED IN FEATURE PREPROCESSING TO REMOVE NOISE FROM DATA.
- CAN BE USEFUL FOR DATA VISUALISATION (SOMETIMES)

TERMINOLOGY

- TRAINING EXAMPLE - ROW IN A TABLE REPRESENTING DATASET USED TO TEACH A MODEL HOW TO MAKE PREDICTIONS.
- TRAINING - MODEL FITTING
- FEATURE (x) - INPUT / VARIABLE
- TARGET (y) - OUTCOME / OUTPUT
- LOSS FUNCTION - COST FUNCTION / ERROR FUNCTION.
  - ↳ "LOSS" - LOSS MEASURED FOR SINGLE DATA POINT
  - ↳ "COST" - COMPUTE LOSS OVER THE ENTIRE DATASET. (AVERAGE OR SUMMED)

MACHINE LEARNING SYSTEMS

PREPROCESSING

- ONE OF THE MOST CRUCIAL STEPS → SHAPE DATA FOR LEARNING ALGORITHMS.
- SELECTED FEATURES → SAME SCALE
- DIMENSIONAL REDUCTION → HIGHLY CORRELATED FEATURES, THEREFORE REDUNDANT TO CERTAIN DEGREE.
  - ↳ ADVANTAGE → LESS STORAGE SPACE REQUIRED + ALGORITHM CAN RUN FASTER.
- DIVIDE DATA SET TO TEST NEW DATA AGAINST TRAINED DATA.
  - ↳ INDUSTRY STANDARD: 80 : 20 - 80 TEST, 20 VALIDATION.

TRAINING, SELECTING AND EVALUATING MODELS

- ONE MODEL ISN'T NECESSARILY GOOD FOR ALL APPLICATIONS
- ESSENTIAL TO TRAIN MULTIPLE MODELS AND TRAIN THEM TO EVALUATE WHICH WORKS BEST.
- COMMON METRIC TO MEASURE PERFORMANCE = CLASSIFICATION ACCURACY.
- "HOW DO WE KNOW WHICH MODEL PERFORMS WELL ON FINAL DATASET ?!" → CROSS-VALIDATION
- CROSS-VALIDATION → FURTHER DIVIDE DATASET INTO TRAINING + VALIDATION SUBSET.
- FREQUENT USE → HYPERPARAMETER FINE-TUNING.
- GENERALIZATION ERROR → HOW ACCURATELY WILL THE TRAINED MODEL PERFORM WITH TEST DATASET (PREDICT OUTCOME).
- SAME PROCEDURES APPLIED TO TRAINING DATASET APPLIED TO TEST DATASET → OTHERWISE MODEL WILL OVERFIT
- OVERFITTING → MACHINE LEARNING MODEL LEARNS TRAINING DATA TOO WELL

CHAPTER 2

ML ALGO FOR CLASSIFICATION → PERCEPTRON + ADAPTIVE LINEAR NEURONS

ARTIFICIAL NEURONS

- FIRST COMPUTATIONAL MODEL OF A NEURON → MCCULLOCH-PITTS → WARREN MCCULLOCH + WALTER PITTS. 1943
- FIRST CONCEPT OF PERCEPTON LEARNING RULE BASED ON MCP → FRANK ROSENBLATT 1957
  - ↳ PROPOSED AN ALGORITHM → LEARN OPTIMAL WEIGHT → MULTIPLIED WITH INPUT FEATURES → ORDER TO MAKE DECISION IF NEURON FIRES OR DOESN'T.
  - ↓
  - CURRENT DAY → IN REGRESSION + CLASSIFICATION
    - ↳ THIS ALGORITHM COULD BE USED TO PREDICT WHETHER A NEW DATA POINT BELONGS TO ONE CLASS OR OTHER
- BINARY CLASSIFICATION → 0 AND 1
  - ↳ $z = w_1x_1 + w_2x_2 \dots w_mx_m$ UNIT STEP FUNCTION:

$$w = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \; x = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \longrightarrow \sigma(z) = \begin{cases} 1 & \text{IF } z \geq \theta \\ 0 & \text{OTHERWISE} \end{cases}$$

LINEAR ALGEBRA

- ABBREVIATE THE SUM OF PRODUCTS OF VALUE x AND w → VECTOR DOT PRODUCT.

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \; b = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \rightarrow \text{TRANSPOSE } a \text{ TO } a^T \rightarrow a^T = [a_1 \; a_2 \; a_3] \rightarrow \text{DOT PRODUCT } a^Tb = \sum_i a_ib_i = a_1 \cdot b_1 + a_2 \cdot b_2 + a_3b_3$$

$$\text{e.g. } \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

PERCEPTRON LEARNING RULE

- REDUCTIONIST APPROACH TO MIMIC HOW SINGLE NEURON WORKS IN THE BRAIN.
- ALGORITHM SUMMARISED:
  1. INIT WEIGHT + BIAS → UNTIL 0 OR SMALL NUMBER
  2. FOR EACH TRAINING EXAMPLE $x^{(i)}$:
     a. COMPUTE OUTPUT $\hat{y}^{(i)}$
     b. UPDATE WEIGHT + BIAS UNIT.
- OUTPUT IS CLASS LABEL PREDICTED BY UNIT STEP FUNCTION.

w_j = x_j IN DATASET.
η = LEARNING RATE (TYPICALLY CONSTANT BETWEEN 0.0 AND 1.0)
$y^{(i)}$ = TRUE CLASS LABEL OF ith TRAINING EXAMPLE
$\hat{y}^{(i)}$ = PREDICTED CLASS LABEL

$$w_j := w_j + \Delta w_j \quad \text{and} \quad b := b + \Delta b \quad \text{UPDATED} \quad \Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad \text{and} \quad \Delta b = \eta(y^{(i)} - \hat{y}^{(i)})$$

- BIAS UNIT + WEIGHT → CONSTANTLY UPDATED SIMULTANEOUSLY →

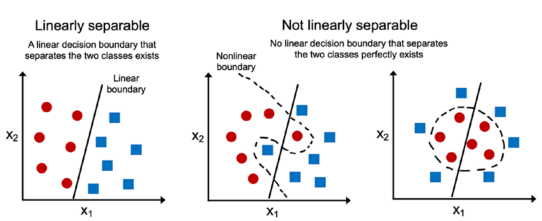
$$\Delta w_1 = \eta(y^{(i)} - \text{output}^{(i)})x_1^{(i)}$$
$$\Delta w_2 = \eta(y^{(i)} - \text{output}^{(i)})x_2^{(i)}$$
$$\Delta b = \eta(y^{(i)} - \text{output}^{(i)})$$

- PERCEPTRON PREDICTS CORRECTLY → BIAS UNIT + WEIGHT REMAIN UNCHANGED → VALUES ARE 0:

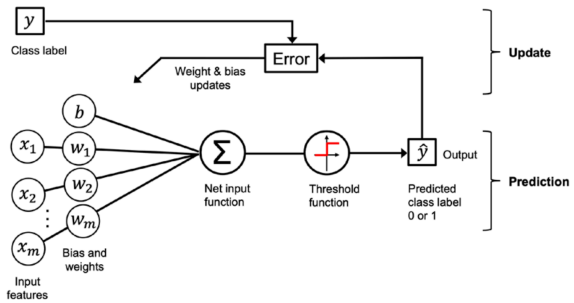
$$y^{(i)} = 0, \; \hat{y}^{(i)} = 0, \; \Delta w_j = \eta(0-0)x_j^{(i)} = 0, \; \Delta b = \eta(0-0) = 0$$
$$y^{(i)} = 1, \; \hat{y}^{(i)} = 1, \; \Delta w_j = \eta(1-1)x_j^{(i)} = 0, \; \Delta b = \eta(1-1) = 0$$

- WRONG PREDICTION → LEANS POSITIVE OR NEGATIVE TARGET CLASS:

$$y^{(i)} = 1, \; \hat{y}^{(i)} = 0, \; \Delta w_j = \eta(1-0)x_j^{(i)} = \eta x_j^{(i)}, \; \Delta b = \eta(0-0) = \eta$$
$$y^{(i)} = 0, \; \hat{y}^{(i)} = 1, \; \Delta w_j = \eta(0-1)x_j^{(i)} = -\eta x_j^{(i)}, \; \Delta b = \eta(1-1) = -\eta$$



- CONVERGANCE OF PERCEPTRON ONLY GUARANTEED → TWO CLASSES ARE LINEARLY SEPERABLE.
- IF TWO CLASSES CAN'T BE SEPARATED BY LINEAR BOUNDARY → SET MAXIMUM NUMBER OF PASSES OVER TRAINING DATASET (EPOCH)



GENERAL CONCEPT OF PERCEPTRON

- INPUT EXAMPLE (x) + BIAS (b) + WEIGHTS (w) → COMPUTE NET INPUT.
  - ↳ HELPS SHIFT DECISION BOUNDARY
- THRESHOLD FUNCTION → CHECKS IF SUM z CROSSES CERTAIN THRESHOLD
  DECIDES OUTPUT = $\begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$
- LEARNING PHASE → WHEN PREDICTION IS WRONG → ADJUSTS WEIGHTS TO GET CLOSER TO CORRECT OUTPUT
  - CONTINUES UNTIL PERCEPTRON CLASSIFIES ALL TRAINING EXAMPLES CORRECTLY.

PERCEPTRON ON IRIS DATASET

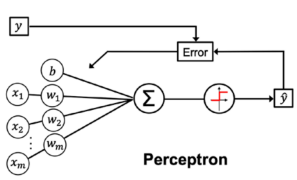
- ONLY USE TWO FLOWER CLASSES → PERCEPTRON IS A BINARY CLASSIFIER.
- PERCEPTRON ALGORITHM CAN BE EXTENDED → ONE-VERSUS-ALL (OvA) TECHNIQUE.
- ONE-VERSUS-ALL → MULTI-CLASS CLASSIFICATION
  - ↳ SOMETIMES CALLED ONE-VERSUS-REST (OvR)
  - ↳ EXTEND BINARY CLASSIFIER → MULTI-CLASS PROBLEM.
  - ↳ TRAIN ONE CLASSIFIER PER CLASS → TREATED AS POSITIVE + EXAMPLES FROM ALL OTHER CLASSES = NEGATIVE.
  - ↳ NEW UNLABELLED DATA → USE n CLASSIFIERS, n = NUMBER OF CLASS LABELS → ASSIGN TO HIGHEST CONFIDENCE TO PARTICULAR INSTANCE.
  - ↳ IN CASE OF PERCEPTRON → OvA → CHOOSE CLASS LABEL THAT IS ASSOCIATED WITH LARGEST ABSOLUTE NET INPUT VALUE.

PERCEPTRON CONVERGENCE

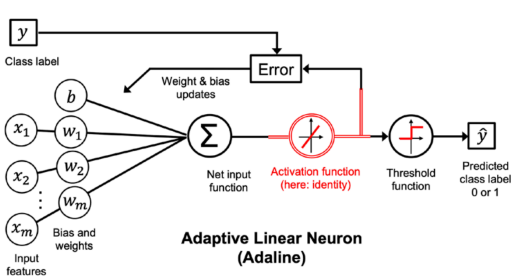
- CONVERGENCE → BIGGEST PROBLEM FOR THE PERCEPTRON.
- IF CLASSES CANNOT BE SEPARATED → WEIGHTS WILL NEVER STOP UPDATING UNLESS WE SET MAXIMUM NUMBER OF EPOCHS.

ADAPTIVE LINEAR NEURONS (ADALINE)

- SINGLE-LAYER NEURAL NETWORK → ADAPTIVE LINEAR NEURON (ADALINE) → BERNARD WIDROW + TEDD HOFF 1960
  - ↳ INTERESTING → KEY CONCEPTS OF DEFINING AND MINIMIZING CONTINUOUS LOSS FUNCTIONS.
    - ↳ LAY THE GROUND WORK FOR UNDERSTANDING OTHER MACHINE LEARNING ALGORITHMS FOR CLASSIFICATION.
      - LOGISTIC REGRESSION
      - SUPPORT VECTOR MACHINES
      - MULTI-LAYER NEURAL NETWORKS
      - ALSO → LINEAR REGRESSION MODEL
- KEY DIFFERENCE WITH ROSENBLATT'S PERCEPTRON → WEIGHTS UPDATED BASED ON LINEAR ACTIVATION FUNCTION VS. UNIT STEP FUNCTION
- LINEAR ACTIVATION FUNCTION = $\sigma(z)$ → IDENTITY FUNCTION OF NET INPUT $\sigma(z) = z$



Perceptron



Adaptive Linear Neuron (Adaline)

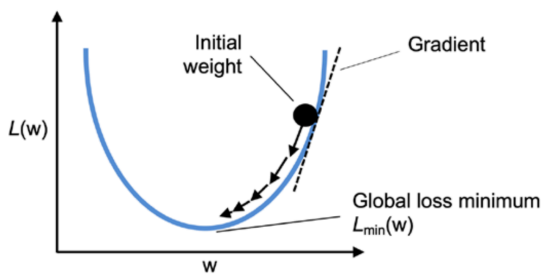
MAIN CONCEPT OF ADALINE

MINIMISING LOSS FUNCTION WITH GRADIENT DESCENT

- KEY INGREDIENT FOR SUPERVISED LEARNING → OBJECTIVE FUNCTION
- OBJECTIVE FUNCTION → LOSS OR COST FUNCTION THAT WE WANT TO MINIMISE.
- ADALINE → LOSS FUNCTION (L) → LEARN MODEL PARAMETERS AS MEAN SQUARE ERROR (MSE) → BETWEEN CALCULATED + TRUE CLASS LABEL

$$L(w,b) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \sigma(z^{(i)})\right)^2$$

- $\frac{1}{2}$ FOR OUR OWN CONVENIENCE → EASIER TO DERIVE THE GRADIENT OF LOSS FUNCTION
- ADVANTAGE:
  - ↳ LOSS FUNCTION BECOMES DIFFERENTIABLE
  - ↳ IT'S CONVEX → CAN USE GRADIENT DESCENT → FIND WEIGHTS THAT MAXIMISE LOSS FUNCTION TO CLASSIFY EXAMPLES IN A DATASET.



GRADIENT DESCENT

- "CLIMBING DOWN HILL" → UNTIL LOSS OR GLOBAL LOSS MINIMUM IS REACHED.
- EACH ITERATION → STEP OPPOSITE DIRECTION OF GRADIENT
  - ↳ STEP DETERMINED BY LEARNING RATE + SLOPE OF GRADIENT.
- OPPOSITE DIRECTION OF $\nabla L(w,b)$: $w: w + \Delta w$, $b: b + \Delta b$
- Δw, Δb DEFINED AS NEGATIVE GRADIENT MULTIPLIED BY LEARNING RATE η

$$\Delta w = -\eta \nabla_w L(w,b), \quad \Delta b = -\eta \nabla_b L(w,b)$$

- GRADIENT OF LOSS FUNCTION → PARTIAL DERIVATIVE OF LOSS FUNCTION → RESPECT TO EACH WEIGHT, w_j

$$\frac{\partial L}{\partial w_j} = -\frac{2}{n}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right) x_j^{(i)}$$

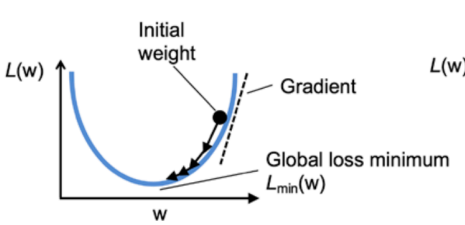
- PARTIAL DERIVATIVE OF LOSS RESPECTED TO BIAS

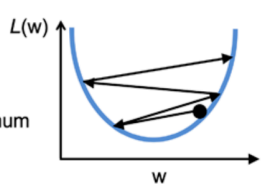
$$\frac{\partial L}{\partial w_j} = \frac{\partial}{\partial w_j}\frac{1}{n}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right)^2 = \frac{1}{n}\frac{\partial L}{\partial w_j}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right)^2$$

$$= \frac{2}{n}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right)\frac{\partial}{\partial w_j}\left(y^{(i)} - \sigma(z^{(i)})\right)$$

$$= \frac{2}{n}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right)\frac{\partial}{\partial w_j}\left(y^{(i)} - \sum_i (w_j x_j^{(i)} + b)\right)$$

$$= \frac{2}{n}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right)\left(-x_j^{(i)}\right) = -\frac{2}{n}\sum_i \left(y^{(i)} - \sigma(z^{(i)})\right) x_j^{(i)}$$





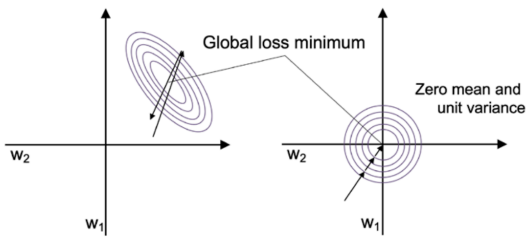
- LEFT IMAGE = WELL-CHOSEN LEARNING RATE (HYPERPARAMETER) WHERE LOSS DECREASES GRADUALLY
- RIGHT IMAGE = OVERSHOT GLOBAL MINIMUM → LEARNING RATE IS TOO LARGE.

STANDARDIZATION

- STANDARDIZATION → NORMALIZATION THAT HELPS GRADIENT DESCENT LEARNING TO CONVERGE MORE QUICKLY. → DATASET NOT NORMALLY DISTRIBUTED
  - SHIFTS THE MEAN OF EACH FEATURE → CENTERED AT ZERO + EACH FEATURES STANDARD DEVIATION = 1

$$x_j' = \frac{x_j - \mu_j}{\sigma_j}$$

  - HELPS GRADIENT DESCENT LEARNING → EASIER TO FIND LEARNING RATE THAT WORKS WELL FOR WEIGHTS + BIAS.
  - GREAT FOR FEATURES WITH DIFFERENT SCALES. → STABILISES TRAINING → OPTIMISER CAN GO THROUGH FEWER STEPS FOR OPTIMAL SOLUTION.



- LEFT UNSCALED FEATURES.
- RIGHT STANDARDIZED FEATURES.

STOCHASTIC GRADIENT DESCENT (SGD)

- ALSO KNOWN AS = ITERATIVE OR ONLINE GRADIENT DESCENT.
- PARAMETERS ARE UPDATED INCREMENTALLY FOR EACH TRAINING EXAMPLE OVER SUM OF ACCUMULATED ERRORS OVER ALL TRAINING EXAMPLES.

$$\Delta w_j = \eta\left(y^{(i)} - \sigma(z^{(i)})\right)x_j^{(i)}, \quad \Delta b = \eta\left(y^{(i)} - \sigma(z^{(i)})\right)$$

- SGD CONSIDERED AN APPROXIMATION TO GRADIENT DESCENT → TYPICALLY REACHES CONVERGENCE FASTER → MORE FREQUENT WEIGHT UPDATES.
- ERROR SURFACE IS NOISIER THAN GRADIENT DESCENT. → ADVANTAGE FOR SGD → ESCAPE SHALLOW MINIMA MORE READILY → NONLINEAR LOSS FUNCTION.
- SGD → PRESENT TRAINING DATA IN RANDOM ORDER. → SHUFFLE DATA EVERY EPOCH → PREVENTS CYCLES.
- SGD - ADAPTIVE LEARNING RATE → DECREASES OVER TIME.

$$\frac{c_1}{[\text{number of iterations}] + c_2}$$

- ADVANTAGE FOR SGD → ONLINE LEARNING
  - ↳ MODEL TRAINED WITH NEW DATA ON THE FLY.
  - ↳ USEFUL FOR LARGE AMOUNTS OF DATA.
  - ↳ SYSTEM CAN IMMEDIATELY ADAPT TO CHANGES.

MINI-BATCH GRADIENT DESCENT

- FULL BATCH GRADIENT DESCENT TO SMALLER SUBSETS OF TRAINING DATA
- ADVANTAGE OVER FULL BATCH → CONVERGENCE REACHED FASTER. (FINDS OPTIMAL WEIGHTS AND BIASES FASTER)
-