Linear Regression

$y = w \cdot x + b$

$y = w_1 \cdot x_1 \dots w_n \cdot x_n + b$ } prediction y

$$Loss = \frac{1}{m} \sum_{i=1}^{m} (y_i - t_i)^2$$ } compare y with t and calculate loss

sum of all errors → squared

m → number of examples

While Loss > precicion :

$w_i = w_i - lr\,(y - t)\,x_i$ } update the weights until the loss is close enough to zero

Binary Logistic Regression

$y = w_1 \cdot x_1 \dots w_n \cdot x_n + b$ } prediction y

$\hat{y} = \sigma(y) = \frac{1}{1 + e^{-y}}$ } apply the sigmoid to y
↳ z = probability value between 0 and 1

$\hat{y}' = \sigma'(y) = \hat{y} \cdot (1 - \hat{y})$ } calculate the derivative of the sigmoid

cross-entropy loss

$$\text{loss} = \frac{1}{m}\sum_{i=1}^{m}\left(t_i \cdot \log(\hat{y}_i) + (1-t_i)\cdot \log(1-\hat{y}_i)\right)$$

* calculate the gradients of the loss function by taking the partial derivatives of the loss function with respect to each parameter

While loss > precision :

$$w_i = w_i - lr\,(y-t)x_i$$

} update the weights until the loss is close enough to zero

Multinomial Logistic Regression

$$y = w_1 \cdot x_1 \dots w_n \cdot x_n + b$$ } prediction y

$$\hat{y} = \frac{e^{y_j}}{\sum_{k=1}^{n} e^{y_k}}$$

} take the exponential of the weighted sum and normalize
↳ divide each exponential of the weighted sum by the sum of all the exponentials

→ softmax

cross-entropy loss

$$\text{loss} = \frac{1}{m}\sum_{i=1}^{m}\left(t_i \cdot \log(\hat{y}_i) + (1-t_i)\cdot \log(1-\hat{y}_i)\right)$$

While loss > precision :

$$w_i = w_i - lr\,(y-t)x_i$$

} update the weights until the loss is close enough to zero

Perceptron

for each neuron j :

$$y_j = w_1 \cdot x_1 \dots w_n \cdot x_n$$

↙ compute activation function

$$y_j = \begin{cases} 1 \text{ if } \sum_{i=0}^{m} w_{ij} \cdot x_i > 0 \\ 0 \text{ if } \sum_{i=0}^{m} w_{ij} \cdot x_i \leq 0 \end{cases}$$

$$w_{i,j} = w_{ij} - lr\,(y_j - t_j)\,x_i$$ } update weights individually

Neural Networks

① Multiply input matrix X with a weight matrix W, and add a bias

$$y = w_1 \cdot x_1 \dots w_n \cdot x_n + b = \sum_{i=0}^{m} w_{ij} \cdot x_i$$

② Use a differentiable activation function (element-wise)

$$\hat{y} = \sigma(y) = \frac{1}{1+e^{-y}} \quad \text{OR} \quad \hat{y} = \max(y, 0)$$

↳ Returns a new matrix

③ Repeat step ① and ② for each layer

(4) At the final layer, use the sigmoid for binary neural networks and multi-label neural networks and the softmax for multi-class neural networks.

(5) Apply a loss function → MSE or Cross-entropy
↳ to the output of the activation function from the final layer (sigmoid or softmax)

(6) Compute the delta terms in the output layer and update the weights between the output layer and hidden layer

delta term at the final node j

$$\delta_o(k_j) = y - t \quad \text{(assumes cross-entropy loss)}$$

(7) Compute the delta terms in the hidden layer

output after activation of node j; delta term at output node i; weight from hidden node j to output node i; for $j = 1, \ldots, k$

$$\delta(\text{hidden}_j) = a_j(1 - a_j) \cdot \sum_{i=1}^{n} \delta_o(k_i) \cdot w_{j,i}$$

* For one instance

$$\delta(h_j) = (y - t) \cdot a_j(1 - a_j) \cdot w_{ji}$$

(8) Update the weights by the deltas in both layers

$\delta_o(kj)$

$$w_{i,j} = w_{ij} - lr \cdot (y - t) \cdot a_j$$

$$v_{i,j} = v_{ij} - lr \cdot \delta(hidden_j) \cdot x_i$$

Scaling, Regularization and Euclidean Distance

min-max-scaling → for each feature, find the minimum and maximum value and transform the feature to numbers in the interval 0,1.

$$\frac{x - min}{max - min}$$

Normalizer

$$\frac{x - mean}{std}$$

Regularization → prevents overfitting

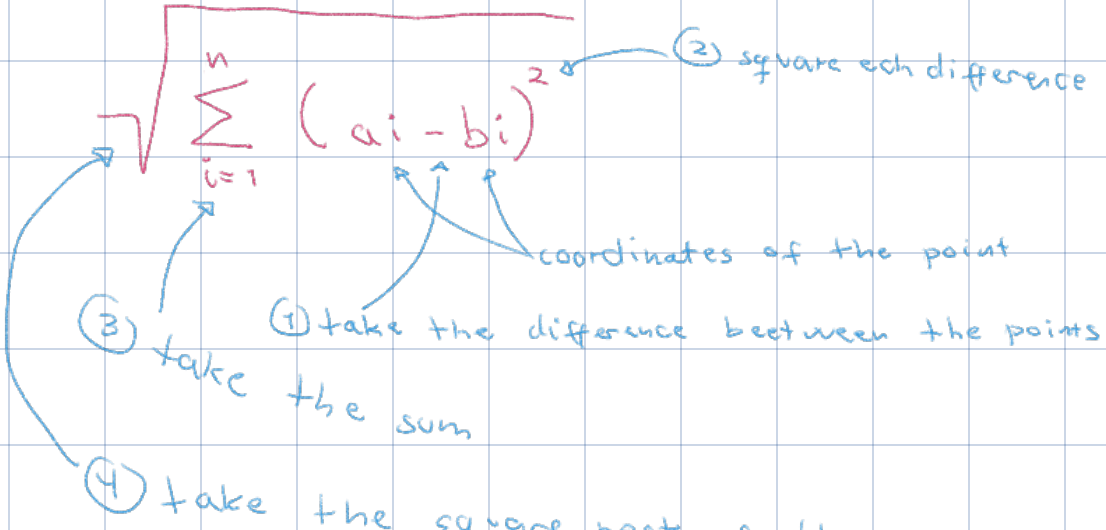
↳ adds a penalty term to the model's loss function

parameter that decides the magnitude of the punishment

$$loss(X, t, w) - \alpha \cdot R(w)$$

$$R(w) = \vec{w} \cdot \vec{w}$$

Euclidean distance → measure of the straight-line distance between two points in a multi-dimensional space.



Reinforcement Learning

greedy action selection
↳ $\max Q(s,a)$ → action with the highest average reward value in the Q-table for state s.

ε-greedy action selection:
↳ initialize a probability $\varepsilon \in [0,1]$
select $\max Q(s,a)$ with prob $1-\varepsilon$
select random a with prob ε

Update rules:

* SARSA *

reward for current action

new state value (assuming current policy)

Q-table value | learning rate | current value | discount rate

$$Q[s,a] = (1-\mu) \cdot Q[s,a] + \mu \cdot (r + \gamma \cdot Q[s',a'])$$

* Q-learning *

new state value, assuming a greedy policy

$$Q[s,a] = (1-\mu) \cdot Q[s,a] + \mu \cdot (r + \gamma \cdot \max Q[s',:])$$

Distance measures

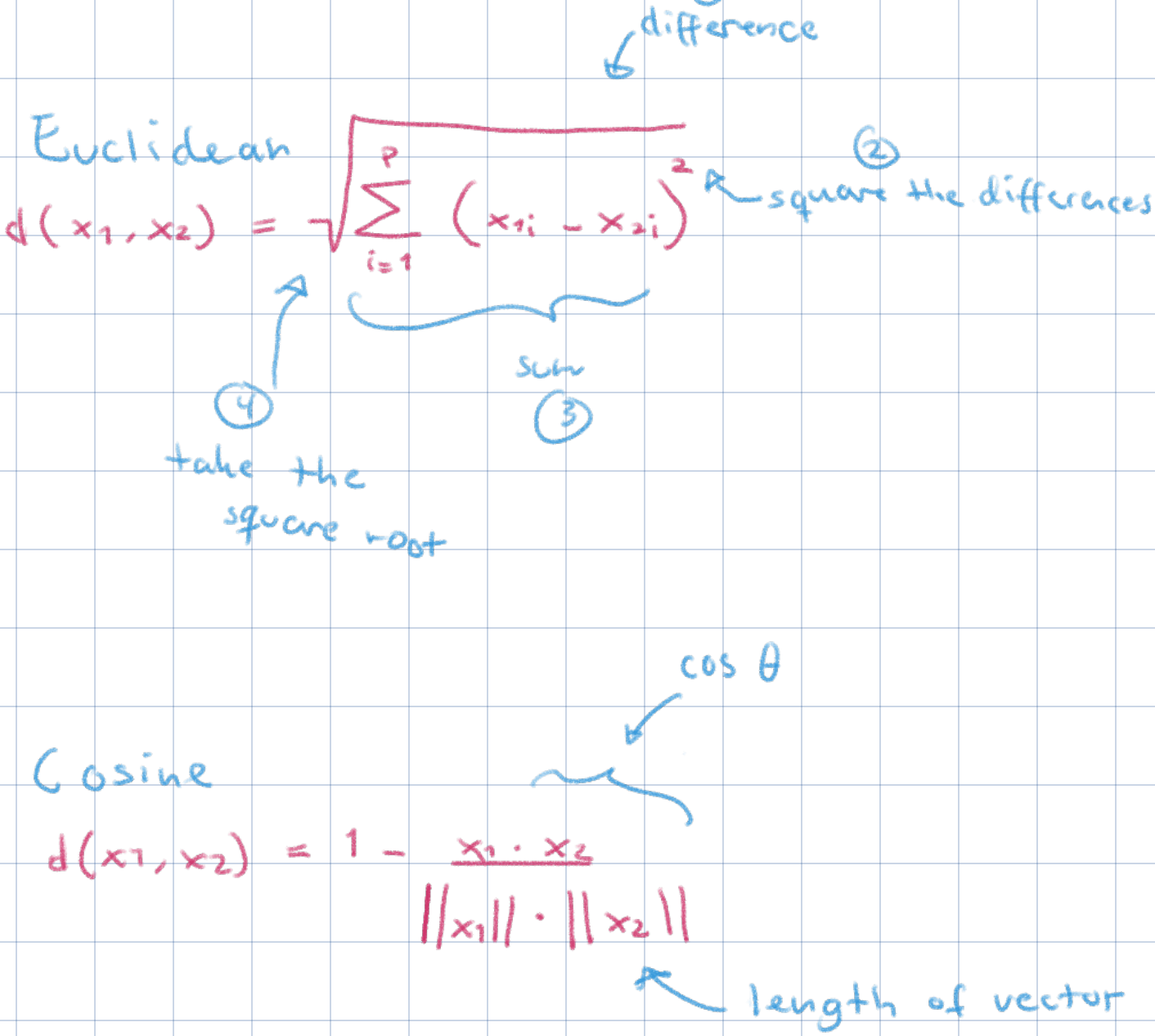
Manhatten

$$d(x_1, x_2) = \sum_{i=1}^{p} |x_{1i} - x_{2i}|$$

absolute difference between each datapoint

Two datapoints

sum of all differences


difference
Euclidean
$d(x_1, x_2) = \sqrt{\sum_{i=1}^{p} (x_{1i} - x_{2i})^2}$
2
square the differences
sum
3
4
take the square root

cos θ
Cosine
$d(x_1, x_2) = 1 - \frac{x_1 \cdot x_2}{\|x_1\| \cdot \|x_2\|}$
length of vector

EVALUATION

True Positive	False Positive
False Negative	True Negative

Recall

$$\frac{TP}{TP + FN}$$

Precision

$$\frac{TP}{TP + FP}$$

Accuracy

$$\frac{TP + TN}{TP + FP + TN + FN}$$

F-Score $\rightarrow$

$$\frac{2 \cdot P \cdot R}{P + R}$$