| | |
|---|---|
| i.i.d | examples are drawn independently from the same distribution – and this same distribution is used to generate both the training set and the test data. |
| overfitting | when a model learns to replicate noise or spurious patterns in the training data. An overfit model may closely match training data but may fail to generalize to unseen test data. |
| underfitting | when a model is too simple. informed by too few features or regularized too much, and can not adequately capture the underlying structure of the data. |
| Supervised learning | Has a clear objective, and the model's performance can be quantitatively measured using metrics. Overfitting risks and labeled data may be time-consuming to obtain. |
| Unsupervised learning | Does not require labeled data, and may lead to insights that were not seen in supervised learning. Not always have a clear semantic meaning, and may not be suited for tasks that require precise categorization or prediction without labeled examples. |
| Reinforcement Learning | Agents and environments, rewards and punishments. ML paradigm where an agent learns to make decisions by interacting with an environment. |
| Neural Networks | A computational model inspired by the structure and functioning of the human brain. Perceptrons, activation functions, feedforward and backpropagation |
| Deep Learning | Convolutional Neural Networks, Recurrent Neural Networks: a subset of machine learning that involves the use of artificial neural networks with multiple layers |
| Evaluation Metrics | quantitative measures used to assess the performance of a machine learning model |
| Ensemble learning | Boosting(ada boost) |
| Bias and Variance | There is a trade-off between bias and variance, and finding the right balance is crucial for building models that generalize well. |
| Bias | High bias can lead the model to underfit the data, meaning it is too simplistic and fails to capture the underlying patterns in the training data. |
| Variance | High variance can lead to overfitting, where the model performs well on the training data but fails to generalize to new, unseen data. |
| Regularization | L1 and L2 regularization. a technique used in machine learning to prevent overfitting and improve the generalization performance of a model. |
| Clustering Algorithms | K-Means, hierarchical clustering. algorithms are used to group similar data points into clusters. Unsupervised learning |
| PCA(主成分分析) | Dimensionality reduction technique. The goal is to retain the most important information in the data while reducing its dimensionality. |
| Deployment | process of taking a trained machine-learning model and making it available for use in a real-world environment |

## Ch. 2 Bayes' rule

$$P(x|y) = P(y|x)P(x)/\,P(y)$$

| | | | |
|---|---|---|---|
| Posterior | P(x\|y) | Likelihood | P(y\|x) |
| Prior | P(x) | Normalization constant | P(y) |

Assume we have a HHHTTHHT
- Maximum likelihood estimate for h: 5/8
- Prior density for the various values of h∈[0,1] give the formula for the posterior probability density p(h|HHHTTHHT) using the prior p(h)

$$p(h|HHHTTHHT) = \frac{p(HHHTTHHT|h)p(h)}{p(HHHTTHHT)}$$

Marginal likelihood: $p(HHHTTHHT) = \int_0^1 q^5(1-q)^3 p(q)dq$

$$= \frac{h^5(1-h)^3 p(h)}{\int_0^1 q^5(1-q)^3 p(q)dq}$$

Conditional probability:
containing the word "payment", given that the email is a spam email, is 72%. Suppose that the conditional probability of an email being spam, given that it contains the word "payment", is 8%. Find the ratio of the probability that an email is spam to the probability that an email contains the word "payment"
payment = p(a), spam = p(b)
p(a|b) = 0.72 = p(a^b)/p(b), p(b|a) = 0.08 = p(a^b)/p(a)
p(b|a)/p(a|b) = p(b)/p(a) = 0.72/0.08 = 9/1

There are two boxes. Box 1 contains three red and five white balls and box 2 contains two red and five white balls. A box is chosen at random p(box = 1) = p(box = 2) = 0.5 and a ball chosen at random from this box turns out to be red. What is the posterior probability that the red ball came from box 1?

P(box=1|red)=?
P(red|box1) = (3/8)/(1/2) = 3/4
P(box1|red) = p(red|box1)*p(box1)/p(red)=(3/4*1/2)/(37/56)

## Ch. 3 Regularized Least Squares

| | |
|---|---|
| Regularized least squares | Distance metrics using norm. L1 is \|x+y\|= 1 (square), L2 is $\sqrt{(x^2+y^2)}$ (circle) |
| Curse of Dimensionality | The volume of space increases as the dimensionality increase |

## Ch. 4 Loss Function, Linear classification, SGD

Linear classification: the goal is to find the linear decision boundary

| | |
|---|---|
| Loss function | Serves as a quantitative measure of how well the model is performing on a given task by comparing its predictions to the actual (ground truth) values. |

$$MSE = \frac{1}{N}\Sigma_{i=1}^{N}(y_i - \hat{}i)^2$$

$y_i$: actual value for sample i, $\hat{}i$: predicted value

Classification loss functions: Binary Cross-Entropy Loss

$$-\frac{1}{N}\Sigma_{i=1}^{N}[y_i \log(\hat{}i + (1-y_i)\log(1-\hat{}i)]$$

| | |
|---|---|
| Linear Regression | used to predict the value of a variable based on the value of another variable |

Linear regression formula: $y_i = f(x_i, \beta) + e_i$

## Ch. 5 Perceptron

| | |
|---|---|
| perceptron | One of the simplest neural network architectures |

Binary input: x, weight: w, bias; b
Linear decision boundary: z = w*x + b = 0
Passes the weighted sum through an activation function to produce a binary output. Converges on linearly separable data.

| $x_0$ | $x_1$ | $x_2$ | $y$ |
|---|---|---|---|
| -1 | 2 | 1 | -1 |
| 1 | 1 | 1 | +1 |
| -2 | 3 | 2 | -1 |
| 2 | -1 | -1 | +1 |

Initially, $w = (-1, 1, 1)$

Simulate one pass through the following data with the perceptron algorithm described in the lecture and homework. Start with w = (−1, 1, 1) and show the resulting weight vector after each example. (Assume that the perceptron algorithm predicts incorrectly when w · x = 0, and ignore the bias term.)

-first example: w · x = 4, wrong,
w = w + yx = (−1, 1, 1) − (−1, 2, 1) = (0, −1, 0)
-second example w · x = −1, wrong,
w = w + yx = (0, −1, 0) + (1, 1, 1) = (1, 0, 1)
-third example w · x = 0,
wrong, w = w + yx = (1, 0, 1) − (−2, 3, 2) = (3, −3, −1)
-fourth example, w · x = 10, correct no update.

## Ch. 6 SVMs (support vector machines)

The support vectors are the closest points to the boundaries. Are the points that lie on the margin boundaries wx+b=1 and wx+b=-1

| | |
|---|---|
| SVM and perceptron difference | while both perceptrons and SVMs are linear classifiers, SVMs emphasize maximizing the margin and provide more flexibility through the use of kernels for handling nonlinear separation |
| SVM | Classification algorithm that finds the maximum margin separating hyperplane |
| Optimization for SVM | Maximizing γ. Constraints will be that all training instances are correctly classified. |
| Properties | w is a linear combination of the support vectors/ pos and neg examples contribute equally to w |
| Hard margin | Hard constraint ↑ w (and b) depend only on support vectors |
| Soft margin | Allow it to make some errors but it will have to pay a price for each error…this price is slack |

## Ch. 7 Kernels

$$x = < x_1, x_2 > \rightarrow \Phi(x) = < x_1^2, \sqrt{2}x_1x_2, x_2^2 >$$
$$\Phi(x) * \Phi(z) = (x * z)^2$$

This is calculated by inserting the method inside the given function
Kernelized SVM Prediction: (Duality)
Functional margin: y(w*x + b)
Geometric margin: y(w*x + b)/||w||₂ →maximizing this is equiv to

$\min_{w,b} ||w||_2$ subject to $y_i(w * x_i + b) \geq 1$ for all i, and
$\min_{w,b} \frac{1}{2}(w * w)$ subject to $1 - y_i(w * x_i + b) \leq 0$ for all i
$w^* = \Sigma_{i=1}^{n} \alpha_i^* y_i \boldsymbol{x_i}$ means **w** is a weighted sum of examples(if without*)

$$b = -\frac{1}{2}(\min_{i:y_i=+1}(w^* * x_i) + \max_{j:y_i=-1}(w^* * x_j))$$

Support vectors: (1,0) ,(0,1), (1,1)
Equation for maximizing margin: y=wx + b, w=-1 so x+y-b = 0
Therefore |x+y-b|/√ (1+1), b = 3/2 → y = -x + 3/2
Geometric margin: |2-2/3|/√ (2) = √ (2)/4

| | |
|---|---|
| Mercer's condition | ensures the positive definiteness of a kernel function. a necessary and sufficient condition for a function to be a valid kernel. |

## Ch. 8 Naïve Bayes

| | |
|---|---|
| Naïve Bayes | Assume that the features are conditionally independent given t the class. |

| $x_1$ | $x_2$ | y (label) |
|---|---|---|
| 1 | 1 | + |
| 1 | 0 | + |
| 0 | 1 | + |
| 0 | 0 | − |
| 0 | 0 | − |
| 1 | 1 | − |
| 1 | 1 | − |

$p(x_1, x_2|y) = p(x_1|y)\, p(x_2|y)$

Naive Bayes estimates:
p(y = +) = 3/7; p(y = −) = 4/7;
p(x1=1|y=+)=2/3;p(x1=1|y=-)=1/2;
p(x1=0|y=+)=1/3; p(x1=0|y=−) = 1/2;
p(x1=1, x2=0|y=+) = 2/3 · 1/3 = 2/9
p(x1 = 1, x2 = 0|−) = 1/2 · 1/2 = 1/4
Thus Naive Bayes estimates
p(+|x) ∝ 3/7·2/9=2/21, and
p(−|x)∝4/7·1/4=3/21

## Ch. 9 K-Nearest-Neighbors

Is a non-parametric method. Normalize dimensions(distance metric)
Decision boundaries change with k(ex, change distance metrics)
Noise can cause problems; 10 % chance of fail

## Ch. 10 Unsupervised learning(k-means)

Steps to finding k
- Randomly initialize center for a kth cluster
- assign example n to the closest center
- points assigned to cluster k
- re-estimate center of cluster k
- return cluster assignments

## Ch. 11 Gaussian Distribution, Expectation Maximization

| | |
|---|---|
| E step (P2) | Find the expectation of where the new point would be assigned to $\mathbb{P}(\mathbf{Z}|\mathbf{X}, \pi^{\text{current}})$ |
| M step (P1) | Change clusters to a different location $\pi^{\text{new}} = \text{argmax}_\pi \sum_{\mathbf{Z}} \mathbb{P}(\mathbf{Z}|\mathbf{X}, \pi^{\text{current}}) \cdot \ln \mathbb{P}(\mathbf{Z}, \mathbf{X}|\pi)$ |

## Ch. 12 Supervised learning (Entropy)

| | |
|---|---|
| Entropy | An amount of information in something |
| Decision Tree | Small and simple trees are better |

$$H(s) = -p_+ \log(p_+) - p_- \log(p_-)$$

## Ch. 13 Reinforcement Learning: Markov Decision Processes (MDPs)

| | |
|---|---|
| Utilities | How much it is worth can be different in people; additive or discounted utility |
| Optimal utility | Bellman Equations; values change over time but the rewards don't change. |
| ModelBased Learning | algorithm builds a model of the underlying structure or patterns in the data during the training process |
| Model Free learning | learning optimal strategies or policies directly from observed data, rather than creating an internal representation of the system. |

Optimal value function

$$V^*(s) = \max_a Q^*(s,a)$$

Optimal value action function

$$Q^*(s,a) = \Sigma_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Optimal policy

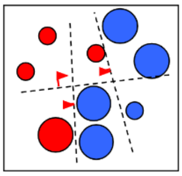
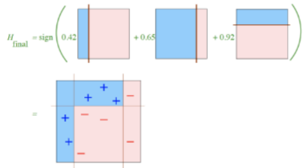
$$\pi^*(s) = \underset{a}{\text{argmax}}\, Q^*(s,a)$$

Value iteration update equations

$$V_{i+1}(s) \max_a \Sigma_{s'} T(s,a,s')[R(s,a,s') + \gamma V_i(s')]$$

- Transition probability function: T(s,a,s')
  Probability of transitioning from s to s' under a
- Reward function: R(s,a,s')
  Immediate reward obtained when trasitioning from s to s' by a
- Discount factor: $\gamma$
  Parameter to determine the importance of future rewards in decision making process
- Action taken by an agent in a given state: a
- Current state: s
- Q-learning: Q(s,a)=(1− α ) · Q(s,a)+ α · (R(s,a,s')+ γ max_a Q(s',a))

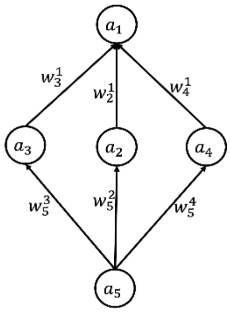
## Ch. 14 Ensemble Methods and Boosting

| | |
|---|---|
| Ensemble Methods | avoid overfitting, increase weights of the misclassified dots |
| AdaBoost  | Give misclassified instances a higher weight. Add up the weak classifiers  |

## Ch. 15 Deep Learning

Multilayer perceptron: neurons are arranged in layers (input, hidden, output)

| | |
|---|---|
| Backpropagation | update parameters by propagating the error backward through the network |




$$\frac{dL}{dW_2} = \frac{dL}{dZ_1} * \frac{dz_1}{dw_2}$$

$$\delta_1 = \frac{dL}{da_1} = \frac{dL}{dz_1} * \frac{dz_1}{da_1} = \frac{dL}{dz_1} * f(a_1)(1 - f(a_1))$$

$$\delta_2 = \frac{dL}{da_2} = \frac{dL}{dz_1} * \frac{dz_1}{dz_2} * \frac{dz_2}{da_2} = f'(a_2) * w_2^1 * \delta_1$$

$$\delta_5 = \delta_2 * w_5^2 + \delta_3 * w_5^3 + \delta_4 * w_5^4$$

$$a_2 = w_5^2 * z_5, z_2 = f(a_2), a_1 = w_2^1 sigmoid\,(z_2) + \cdots + w_4^1 sigmoid\,(z_4)$$

$$z = \Sigma_i sigmoid(a_i)\ where\ i = [2,3,4]$$

$$z = sigmoid(w_2^1 sigmoid\left(f(w_5^2 * z_5)\right) + \cdots + w_4^1 sigmoid\left(f(w_5^4 * z_5)\right))$$