Julius-Maximilians-UNIVERSITÄT WÜRZBURG

CAIDAS

WüNLP

ALGORITHMS IN AI & DATA SCIENCE 1 (AKIDS 1)

Parametric Classification

Prof. Dr. Goran Glavaš

1.2.2024

Content

- Supervised ML: Categorization
- (Some) Parametric Models
  - Naive Bayes
  - Logistic Regression

Supervised ML

- Two important dimensions of division in supervised ML

1. **Parametric** vs. **Non-parametric** models
2. **Generative** vs. **Discriminative** models

- Today we will see some **parametric** models
  - Naive Bayes: Generative
  - Logistic regression

- Next time we will see some **non-parametric** models
  - Decision Trees, k-Nearest Neighbors

Recap: Supervised ML

Three components of a supervised **machine learning algorithm**

1. Model: a set of functions among which we're looking for the best

$$H = \{ h(\mathbf{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$$

- **hypothesis** = a concrete function obtained for some values $\boldsymbol{\theta}$
- Model is a set of hypothesis

2. Loss function L: used to compute the **empirical error** E on a dataset $D = \{(\mathbf{x}, y)_i\}$

$$E(h|D) = \frac{1}{N}\sum_{i=1}^{N} L(h(\boldsymbol{x}_i|\boldsymbol{\theta}), y_i)$$

3. Optimization procedure: procedure or algorithm with which we find the hypothesis h^* from the model H that **minimizes** the empirical error

- Equivalent to finding parameters $\boldsymbol{\theta}^*$ that minimize E

$$h^* = \mathrm{argmin}_{h \in H} E(h|D)$$

$$\boldsymbol{\theta}^* = \mathrm{argmin}_{\boldsymbol{\theta}} E(h|D)$$

Parametric vs. Non-Parametric

Model: a set of functions among which we're looking for the best

$$H = \{ h(\mathbf{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$$

- Parameters $\boldsymbol{\theta}$ estimated using the annotated dataset $D = \{(\mathbf{x}, y)_i\}$

Parametric vs. non-parametric models

A model $H = \{ h(\mathbf{x}|\boldsymbol{\theta})\}_{\boldsymbol{\theta}}$ is **parametric** if its number of parameters n, $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]$ (estimated in model training) is **fixed** and **does not depend** on the size of the training dataset $D = \{(\mathbf{x}, y)_i\}$ (i.e., number of training examples). Otherwise, the model is **non-parametric**.

Recap: Linear Regression

- **Linear Regression** is arguably the simplest supervised ML models
  - In statistics called „ordinary least squares”, or just „regression”
  - Model output is a linear combination of input features

$$h(\mathbf{x} = [x_1, x_2, \ldots, x_n] \,|\, \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- **Q:** Is linear regression **parametric** or **non-parametric**? Why?

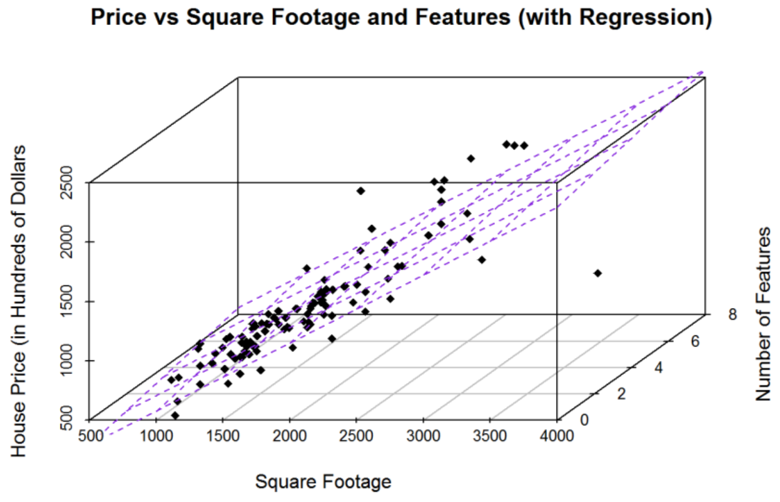


Image from: https://rpubs.com/svoboa/64900

Generative vs. Discriminative Models

Discriminative Models

Discriminative models explicitly model the **decision boundary** between the classes (and nothing else). In other words, the parameters of discriminative models define the decision boundary function.

Generative Models

Generative models model the (probability) **distributions of examples** in **classes.** Learning the distributions of examples within classes, they can then easily derive the **decision boundary** post-hoc from those distributions. So generative models model **more** than just the decision boundary between the classes.

- Generative models typically have **more parameters** than discriminative and are typically data „hungrier”: require **more data** for training

Recap: Linear Regression

- **Linear Regression** as a classifier
  - Model output is a linear combination of input features

$$h(\mathbf{x} = [x_1, x_2, \ldots, x_n] \mid \boldsymbol{\theta}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

- Linear regression is not really suitable as a classifier (it's a regression model), but assuming we use it as such...

- **Q:** is it a **generative** or **discriminative** model? Why?

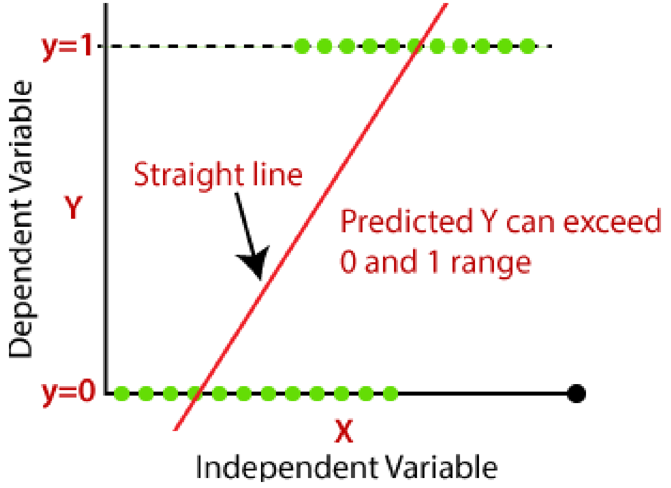


Image from: https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning

Content

- Supervised ML: Categorization
- (Some) Parametric Models
  - Naive Bayes
  - Logistic Regression

Naive Bayes

- **Naive Bayes** is a **generative** classification algorithm based on:
  - **Bayes' rule** and a
  - **„Naive" assumption** that input features $x_1, x_2, \dots, x_n$ are mutually independent (conditionally independent, when conditioned on classes)

- **Bayes rule**: $P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$

- Or, in our **classification** case:

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y) * P(y)}{P(\mathbf{x})}$$

This is called **posterior** (probability of the class, having seen the „evidence", which is our example **x**)we need to compute for **every class** y in order to make a decision which class is most likely for some input **x**

We will estimate these from the training set. These probabilities are **parameters** of the NB.

- $P(\mathbf{x}|y)$ – **likelihood** (that **x** is „generated" if y is the class)
- $P(y)$ – **prior** (probability of the class, without knowing anything about the example)

Naive Bayes

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y) * P(y)}{P(\mathbf{x})}$$

- Note that the denominator does not depend on y: $P(\mathbf{x})$ is going to be the same for all classes y

- For **classification** alone, we don't need to compute $P(\mathbf{x})$
  - Whichever class has the largest value for $P(\mathbf{x}|y) * P(y)$ is also going to have the largest value for the full $P(y|\mathbf{x})$ (as $P(\mathbf{x})$ is the same for all classes)

$$P(y|\mathbf{x}) \propto P(\mathbf{x}|y) * P(y)$$

  - So, in order to classify $\mathbf{x}$ we „only" need to compute **likelihoods** $P(\mathbf{x}|y)$ and **priors** $P(y)$ for each class y

Naive Bayes

- Let Y be the set of classes we're classifying into
  - $y \in Y$ denotes one (any) class from that set
- **Bayes classification model** is then given with

$$h(\mathbf{x}) = \text{argmax}_{y \in Y} P(\mathbf{x} | y) * P(y)$$

- **Key question**: how to compute $P(y)$ and $P(\mathbf{x}|y)$ – that is our model parameters, using the training dataset $D = \{(\mathbf{x}, y)_i\}$?

Naive Bayes (Discrete Features): Example

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes

- How to compute $P(y)$ using the training dataset $D = \{(\mathbf{x}, y)_i\}$?
- **Maximum Likelihood Estimation** = estimate probabilities based on what is most likely according to the training data
- $P(y) = \mathbf{count}(y) / N$ (size of the training set)
- $P(y = \text{No}) = 5/14$
- $P(y = \text{Yes}) = 9/14$
- $P(y = \text{No}) + P(y = \text{Yes}) = 1!$
  - When we sum the probabilities for a random variable over all possible values it always **has to sum up to 1**

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes

- How to compute $P(\mathbf{x}|y)$ using the training dataset $D = \{(\mathbf{x}, y)_i\}$?
- **Maximum Likelihood Estimation** = estimate probabilities based on what is most likely according to the training data
- Let $\mathbf{x} = [x_1 = \text{rain}, x_2 = \text{hot}, x_3 = \text{high}, x_4 = \text{weak}]$
  $y = \text{No}$
- $P(\mathbf{x}|y) = \text{count}((\mathbf{x}, y)) / N = \mathbf{0}/14 = \mathbf{0}$
  - Would also be 0 if y = Yes
- **Problem**: we don't make predictions for examples seen in the training data!
  - Test/inference examples are, in principle, **unseen** in training data

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes

- Naive Bayes solves this issue by **„naively"** factorizing P($\mathbf{x}$|y) into a **product of class-conditional probabilities** of features P(x_i|y)

$$P(\mathbf{x} = [x_1, x_2, ..., x_n] \mid y) = P(x_1|y) * P(x_2|y) * ... * P(x_n|y)$$

- The above equation is only true if features x1, x2, ..., xn are all **mutually independent**: no correlation between their values
- In practice, this is almost never the case and the product is merely a **naive approximation** of the likelihood P($\mathbf{x} = [x_1, x_2, ..., x_n] \mid y$)
  - But we are much more likely to successfully conditional probability for individual feature values P(x_i|y) values on the training dataset than the conditional probability of the whole example P($\mathbf{x} = [x_1, x_2, ..., x_n] \mid y$)

Naive Bayes

- Let Y be the set of classes we're classifying into
  - y ∈ Y denotes one (any) class from that set
- **Naive Bayes model** is then given with

$$h(\mathbf{x}) = \text{argmax}_{y \in Y} \, P(y) \; * \prod_{i=1}^{n} P(x_i | y)$$

- **Loss function**? Not obvious, but it's actually a **0-1 loss**!
  - But that's not differentiable?
  - **Doesn't matter**, as **we're not using any numerical optimization**!
- **Optimization** (i.e., parameter estimation)?
  - **Maximum likelihood estimation**
    (somewhat different for numerical than for discrete feats, as we'll see in a bit)

Naive Bayes: Parameter Estimation

- **Priors**: $P(y = \text{No}) = 5/14$, $P(y = \text{Yes}) = 9/14$

- **Likelihoods**:
  - $P(x_1\text{=sunny}|\text{No}) = 3/5$; $P(x_1\text{=rain}|\text{No}) = 2/5$; $P(x_1\text{=overcast}|\text{No}) = \mathbf{0}/5$
  - $P(x_1\text{=sunny}|\text{Yes}) = 2/9$; $P(x_1\text{=rain}|\text{Yes}) = 3/9$; $P(x_1\text{=overcast}|\text{Yes}) = 4/9$
  - $P(x_2\text{=hot}|\text{No}) = 2/5$; $P(x_2\text{=mild}|\text{No}) = 2/5$; $P(x_2\text{=cool}|\text{No}) = 1/5$
  - $P(x_2\text{=hot}|\text{Yes}) = 2/9$; $P(x_2\text{=mild}|\text{Yes}) = 4/9$; $P(x_2\text{=cool}|\text{Yes}) = 3/9$
  - $P(x_3\text{=high}|\text{No}) = 4/5$; $P(x_3\text{=normal}|\text{No}) = 1/5$
  - $P(x_3\text{=high}|\text{Yes}) = 3/9$; $P(x_3\text{=normal}|\text{Yes}) = 6/9$
  - $P(x_4\text{=weak}|\text{No}) = 2/5$; $P(x_3\text{=strong}|\text{No}) = 3/5$
  - $P(x_4\text{=weak}|\text{Yes}) = 6/9$; $P(x_3\text{=strong}|\text{Yes}) = 3/9$

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes: Parameter Estimation

- **Problem**: we might still have unseen combinations of parameter values and classes
  - Like in the example: x_1 = overcast, y = No
  - Effectively prevents us from making a prediction for any example for which the value of x_1 is „overcast”

$P(y=No \mid \mathbf{x} = [x_1 = overcast, x_2 = cool, x_3 = normal, x_4 = strong]) \propto$

$P(y=No) * P(x_1=overcast \mid No) * P(x_2=cool \mid No) * P(x_3=normal \mid No) * P(x_4=strong \mid No)$

$= 5/14 * \mathbf{0}/5 * 1/5 * 1/5 * 3/5$

$= \mathbf{0}$

- Single unseen feature-class combination pushes the whole posterior to 0.

Naive Bayes: Smoothing

- **Smoothing**: artificial reassignment of probability mass – to give some of it to **unseen events**, in order to prevent 0 probabilities in products

- **Additive (or Laplace) smoothing**
  - Add some small quantity α (for example 1 or 0.5) to each count of feature value (conditioned on each of the classes)
  - $P(x_i = \text{value} \mid y) = \dfrac{\textbf{count}(x_i = \text{value} \mid y) + \alpha}{\textbf{count}(y) + \alpha * V_{xi}}$
  - If **count**(x_i = value | y) is 0, the nominator is α (so not zero!)
  - V_{xi} is the number of different values x_1 can have (in our example, $V_{xi} = 3$)
    - Since we add α to the nominator of each of those values, adding $V_{xi} * \alpha$ to the denominator **ensures** that $P(x_i \mid y)$ is **still a probability distribution**

Naive Bayes: Parameter Estimation with Smoothing

- **Priors**: P(y = No) = 5/14, P(y = Yes) = 9/14

- **Likelihoods (**with smoothing, α = 0.5**)**:
  - $P(x_1=sunny|No) = (3+0.5)/(5+3*0.5)$; ...
  - $P(x_1=sunny|Yes) = (2+0.5)/(9 +3*0.5)$; ...
  - $P(x_2=hot|No) = (2+0.5)/(5+3*0.5)$; ...
  - $P(x_2=hot|Yes) = (2+0.5)/; (5+3*0.5)$; ...
  - $P(x_3=high|No) = (4+0.5)/(5+2*0.5)$...
  - ...

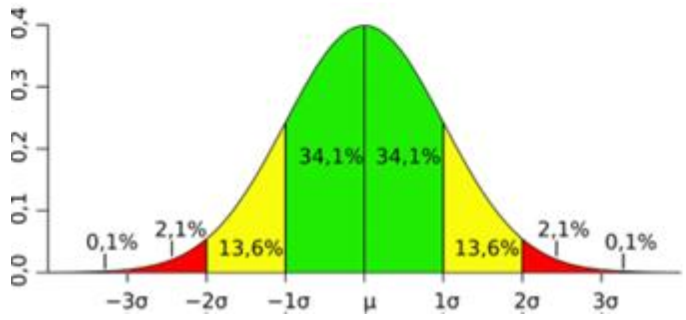
Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Naive Bayes with Numeric Features

- How do we estimate $P(x_i|y)$ if x_i is a **numeric feature**?
  - In this case we cannot use counting

- **Maximum likelihood estimation** for **numeric variables** requires an assumption of a distribution of the „continous random variable"
  - Those distributions are defined with **probability density functions**
  - **Normal (or Gaussian distribution):**

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}$$

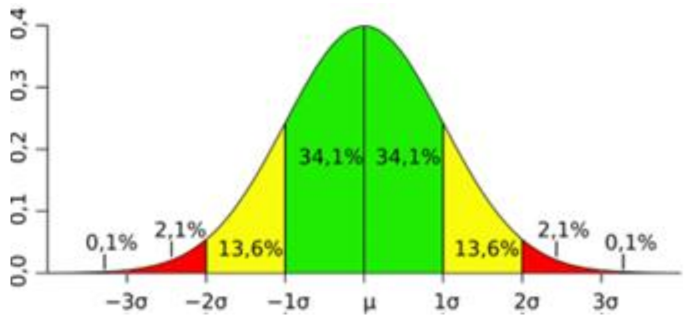


For a concrete value of x, p(x)
gives the „probability density" for x, which we treat as „probability" for all intents and purposes (the notation difference is lower-cased p)

Naive Bayes with Numeric Features

- How do we estimate $P(x_i|y)$ if x_i is a numeric feature?
  - In this case we cannot use counting
  - **Normal (or Gaussian distribution):**

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}}\, e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}$$



  - But to be able to compute p(x) for some x, we need to estimate μ and σ from our training set
  - $p(x|y)$ – **each likelihood** (for each class) is assumed to be one normal distr.
  - μ = mean (average) of values x across all instances of the class y
  - $\sigma = \sqrt{\sum_i (xi - \mu)^2 / N}$ (N is the number of examples/instances)

Naive Bayes with Numeric Features: Example

x_1	x_2	y
-0.17	1.54	Yes
-0.89	1.18	Yes
-2.61	0.87	Yes
-0.91	0.58	Yes
1.17	0.16	Yes
1.54	2.07	No
1.22	3.58	No
1.85	2.77	No
2.44	2.88	No
0.90	3.64	No

- We have four normal distributions to estimate
  - $p(x_1|Yes)$, $p(x_1|No)$
  - $p(x_2|Yes)$, $p(x_2|No)$
  - This means computing μ and σ for each of the four

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)}$$

- Example
  - $\mu_{x1|Yes} = (-0.17 - 0.89 - 2.61 - 0.91 + 1.17) / 5 = -0.68$
  - $\sigma_{x1|Yes} = \sqrt{[(-0.17+0.68)^2 + (-0.89+0.68)^2 + (-2.61+0.68)^2 + (-0.91+0.68)^2 + (1.17+0.68)^2]/5}$
    $= 1.225$
  - For some **new** value of x_1 we compute $p(x_1|Yes)$ by simply putting that value and $\mu_{x1|Yes}$ and $\sigma_{x1|Yes}$ into the Gaussian formula above
- NB can seamlessly combine **numeric** and **discrete features**!

Content

- Supervised ML: Categorization
- (Some) Parametric Models
  - Naive Bayes
  - Logistic Regression

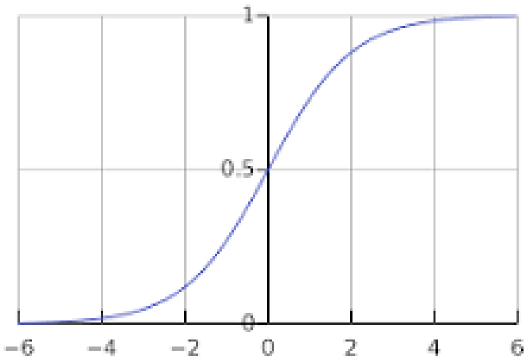
Logistic Regression

- **Logistic regression** is a discriminative and parametric supervised machine learning algorithm for **binary classification**
  - Despite the name, it's a classification and not a regression algorithm!
- **Model**: $h(\mathbf{x}|\mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w})$

$$= \frac{1}{1+\exp(-\mathbf{x}^T\mathbf{w})}$$

$$= \frac{1}{1+\exp(-(w_0 + w_1 * x_1 + \ldots + w_n * x_n))}$$

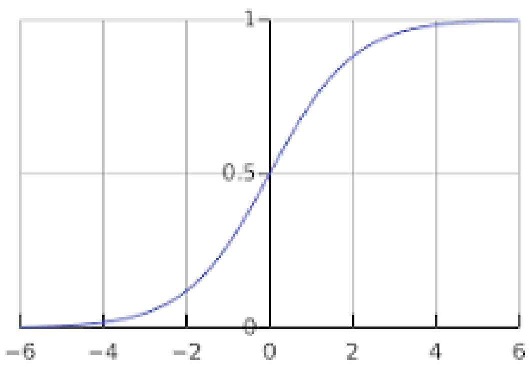
$\sigma(x) = 1/(1+e^{-x})$



- σ is the so-called sigmoid function
- $\mathbf{w} = [w_0, w_1, \ldots, w_n]$ is the vector of **parameters** of logistic regression

Logistic Regression

$\sigma(x) = 1/(1+e^{-x})$



- **Model**: $h(\mathbf{x}|\mathbf{w}) = \sigma(\mathbf{x}^T\mathbf{w})$
- **Loss function**: cross-entropy error
  - $L_{CE}(h(\mathbf{x}_i|\mathbf{w}), y_i) = -[\ y_i * \ln h(\mathbf{x}_i|\mathbf{w}) +$
    $(1 - y_i) * \ln (1 - h(\mathbf{x}_i|\mathbf{w}))\ ]$
- True label y_i is either 0 or 1 (cannot be both :)
  - If $y_i = 0$ then only $(1 - y_i) * \ln (1 - h(\mathbf{x}_i|\mathbf{w}))$ „survives"
  - If $y_i = 1$ then only $y_i * \ln h(\mathbf{x}_i|\mathbf{w})$ „survives"
- **Optimization**: find **w** that minimize empirical error on the training set

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} \frac{1}{N}\sum_{i=1}^{N} L(\,h(\boldsymbol{x}_i|\mathbf{w}), y_i)$$

$$\mathbf{w}^* = \operatorname{argmin}_{\mathbf{w}} -\frac{1}{N}\sum_{i=1}^{N} [\ y_i * \ln h(\mathbf{x}_i|\mathbf{w}) + (1 - y_i) * \ln (1 - h(\mathbf{x}_i|\mathbf{w}))\ ]$$

Logistic Regression

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}}\ \mathrm{E}$$

$$\text{Minimize per } \mathbf{w}: -\frac{1}{N}\sum_{i=1}^{N}[\ y_i * \ln h(\mathbf{x}_i|\mathbf{w}) + (1 - y_i) * \ln(1 - h(\mathbf{x}_i|\mathbf{w}))\ ]$$

- **Q:** How do we find the minimum of a continuous function?
  - We compute the gradient and solve the equation „gradient = 0"

$$\nabla_{\mathbf{w}}\mathrm{E} = 0$$

$$\nabla_{\mathbf{w}}\left[-\frac{1}{N}\sum_{i=1}^{N}[\ y_i * \ln h(\mathbf{x}_i|\mathbf{w}) + (1 - y_i) * \ln(1 - h(\mathbf{x}_i|\mathbf{w}))\ ]\right] = 0$$

  - Unlike for linear regression, this equation has no closed form solution.
  - **Q:** What do we do then? Hint: Cross-entropy loss is differentiable per **w**

Logistic Regression: Features

- From the formula of the logistic regression, it's obvious it works only with numbers – only allows **numeric features**

- For many problems, we also have (good, indicative) **discrete features**
- **Q:** How to turn discrete features into numeric features?

  X_1 has 3 possible values: sunny, overcast, rain?
  - **Q:** How about: sunny → 1, overcast → 2, rain → 3?
  - This is not good: introduces ordering between feature values

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

- **One-hot-encoding**: if a feature has V possible values, each value is converted into V binary features
  - [is_sunny, is_overcast, is_rain]
  - sunny -> [1, 0, 0], overcast = [0, 1, 0], rain = [0, 0, 1]

Questions?

Awọn ibeere?
Turite klausimų?
ਸਵਾਲ?
Küsimusi?
Pitanja?
Sorusu olan?
Dúvidas?
質問は？
有问题吗？
Fragen?
¿Preguntas?
Frågor?
Questions?
Pytania?
Domande?
Питання?
Vragen?
Ερωτήσεις;
Porandukuéra?
คำถาม?
أسئلة؟