Course number: 80240743

Deep Learning

Xiaolin Hu (胡晓林) & Jun Zhu (朱军)
Dept. of Computer Science and Technology
Tsinghua University

Lecture 2: Math and Machine Learning Basics

Xiaolin Hu
Department of Computer Science and Technology
Tsinghua University

Outline

1. Math basics
2. Machine learning basics
3. Regression and classification
4. Summary

Studied by yourself

Summary of Part 1

- Linear algebra
  - Math objects: Scalars, vectors, matrices, tensors
  - Simple operations: matrix transpose, inverse, product
  - Norms: L_p norm
- Probability theory
  - Random variables: discrete, continuous
  - Prob distribution: PMF and PDF
  - Marginal probability
  - Conditional probability
  - Independence and conditional independence
  - Expectation, variance and covariance
  - Common prob distributions
  - Bayes' rule
- Optimization
  - Gradient descent
  - Critical points
  - Rules in calculus

Outline

1. Math basics
2. Machine learning basics
3. Regression and classification
4. Summary

Learning algorithms

"A computer program is said to learn from experience E w.r.t. some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." ---Tom Mitchell, 1997

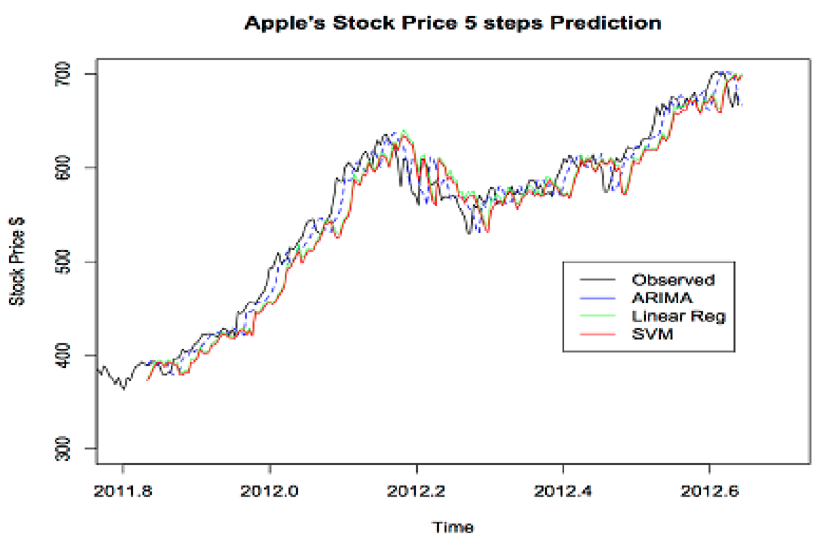
- Machine learning (ML) tasks are usually described in terms of how the ML system should process an example
- An example is a collection of features that have been quantitatively measured from some object or event
  - Features of a bucket: color, diameter, height, material, etc.
  - Features of an animal: size, shape, number of legs, etc.






The tasks T

- Classification
  - Suppose there are k categories. Find a function $f: \mathbb{R}^n \to \{1, \dots, k\}$



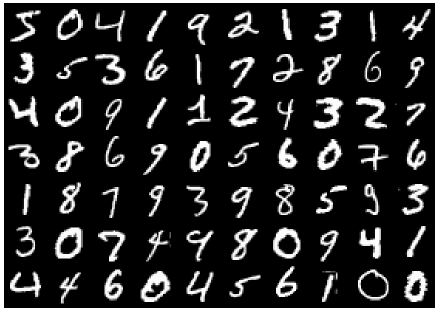
- Regression
  - Find a function $f: \mathbb{R}^n \to \mathbb{R}^m$

Regression results can be converted to classification results

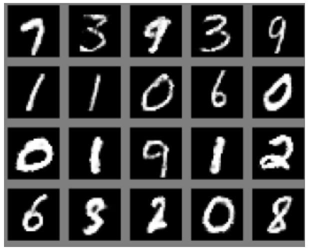
The tasks T

- Synthesis and sampling

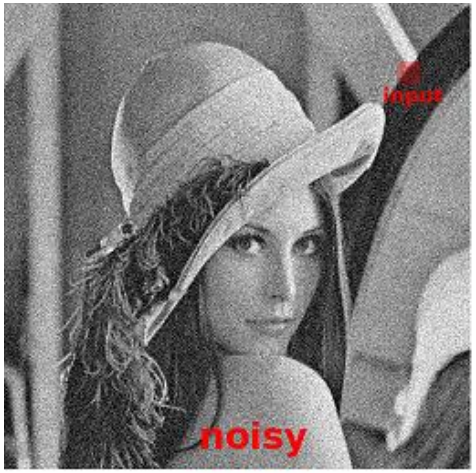
dataset



Synthesized using GAN



- Denoising



The tasks T

- Transcription

Input

Output

Deep learning is cool

- Machine translation

The tasks, T

- Structured output
- Anomaly detection
- Synthesis and sampling
- Imputation of missing values
- Density estimation
- Etc.

The performance measure, P

- To quantitatively evaluate the performance of a ML system
- Usually this measure P is specific to the task T being carried out by the system
  - Classification and transcription: error rate
  - Regression and denoising: distance between the ground-truth and prediction
  - Synthesis, machine translation: difficult and sometimes need human evaluation
- What we are more interested in is the performance measure on a test set of data that is separated from the data used for training the system

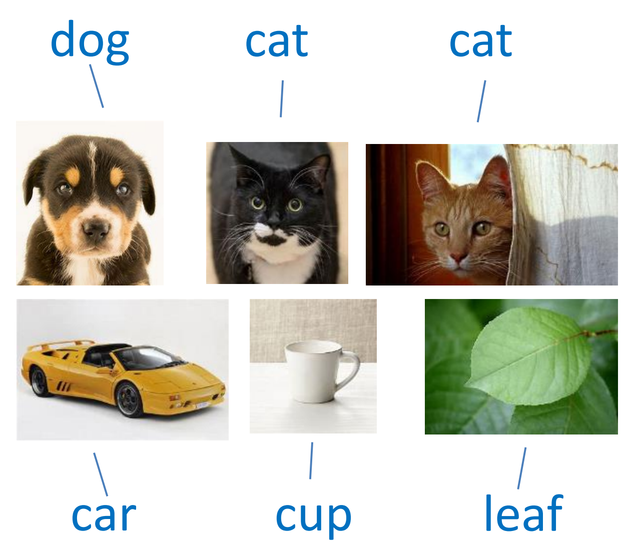
The experience, E

- ML algorithms can be broadly categorized as unsupervised and supervised by what kind of experience they are allowed to have during the learning process
- The algorithms experience a dataset, which is a collection of many examples or data points denoted by $\boldsymbol{x}$
  - We can view examples as samples of a random variable $\mathbf{x}$
- Unsupervised learning

learn $p(\mathbf{x})$

- Supervised learning algorithms

learn $p(\mathbf{y}|\mathbf{x})$



Example: linear regression

x_i: feature
w_i: weight

- Task T: to predict y from x by outputting $\hat{y} = \boldsymbol{w}^\top \boldsymbol{x}$
- Performance P: mean squared error of the model on the test with m test samples $\{(\boldsymbol{x}_i, y_i)\}^{\text{test}}$

$$\text{MSE}_{\text{test}} = \frac{1}{m} \sum_i (\hat{y}_i - y)^2$$

- Experience E: minimize the MSE on the training set of q samples $\{(\boldsymbol{x}_i, y_i)\}^{\text{train}}$

$$\text{MSE}_{\text{train}} = \frac{1}{q} \sum_i (\hat{y}_i - y)^2$$

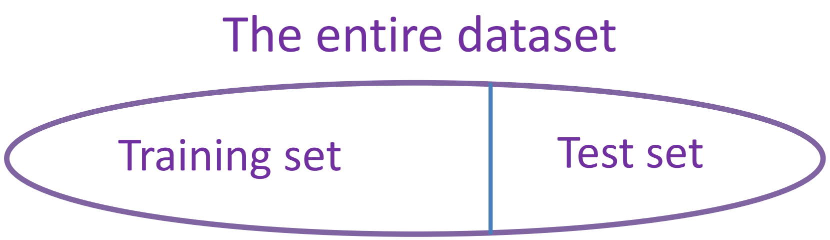
  - Denote $\{(\boldsymbol{x}_i, y_i)\}^{\text{train}}$ collectively by $\left(\boldsymbol{X}^{\text{train}}, \boldsymbol{y}^{\text{train}}\right)$, then

$$\nabla_w \text{MSE}_{\text{train}} = \nabla_w \frac{1}{q} \left|\left|\hat{\boldsymbol{y}}^{\text{train}} - \boldsymbol{y}^{\text{train}}\right|\right|_2^2 = 0$$

$$\Rightarrow \boldsymbol{w} = \left({\boldsymbol{X}^{\text{train}}}^\top \boldsymbol{X}^{\text{train}}\right)^{-\boldsymbol{1}} {\boldsymbol{X}^{\text{train}}}^\top \boldsymbol{y}^{\text{train}}$$

Capacity, overfitting and underfitting

- A ML algorithm must perform well on new, previously unseen inputs—not just on which it was trained
  - This ability is called generalization



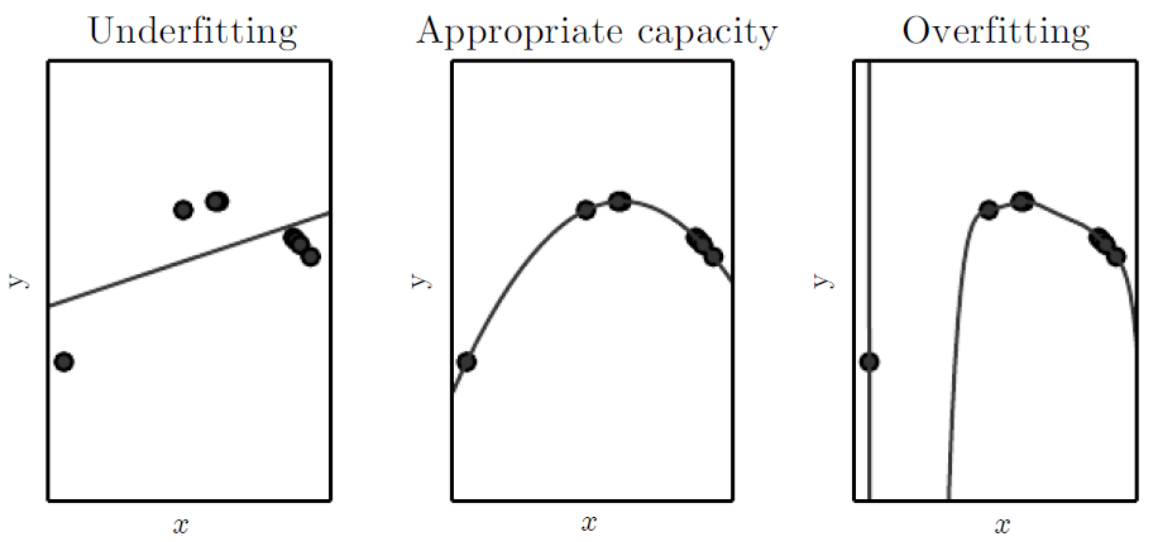
- Smaller training error → higher model capacity
  - If the training error is too large, the model is underfitting the training set
- Smaller test error or generalization error → higher generalization ability
  - If the training error is very small but the test error is very large, the model is overfitting the training set

Example: polynomial regression

- Consider a regression problem in which the input x and output y are both scalars. Find a function $f: \mathbb{R} \to \mathbb{R}$ to fit the data
  - $f(x) = b + wx$
  - $f(x) = b + w_1 x + w_2 x^2$
  - $f(x) = b + \sum_{i=1}^{9} w_i x^i$

MSE training:

$$\min_{w} \frac{1}{N} \sum_{n=1}^{N} \left\| f\left(x^{(n)}\right) - y^{(n)} \right\|_2^2$$



General principles

- Increase the model capacity
  - Make the training error small
- Increase the generalization ability
  - Make the gap between training error and test error small

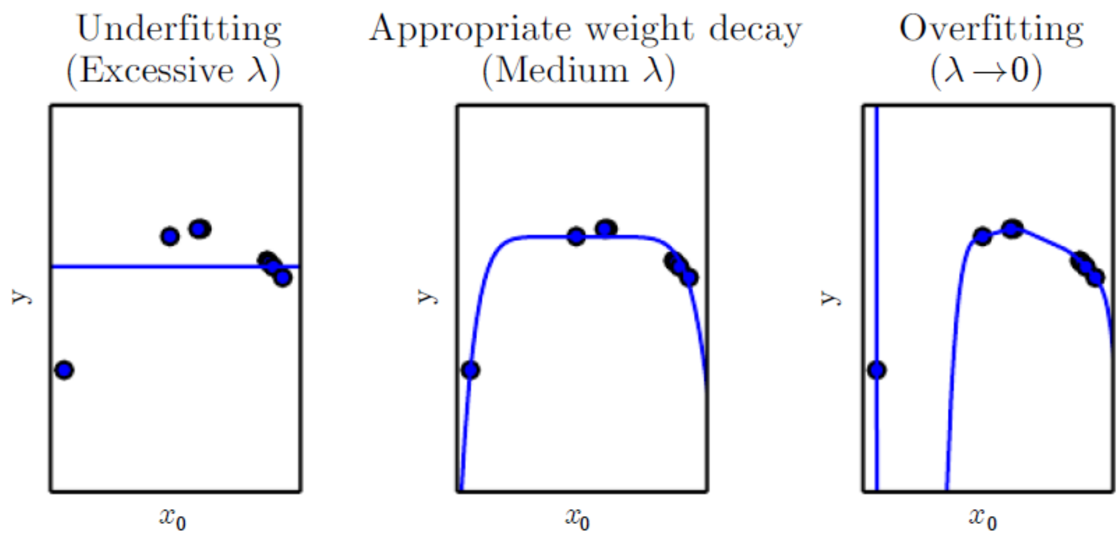
Regularization

- To carry out a specific task, we often build a set of preferences into the learning algorithm, which is embodied by a regularizer Ω
- E.g., for polynomial regression, the total cost function becomes

$$J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^{\top}\boldsymbol{w}$$

Weight decay

where $\lambda > 0$ is a constant.



A high-degree polynomial regression example

- Here $\Omega(\boldsymbol{w}) = \boldsymbol{w}^{\top}\boldsymbol{w}$
- There are many regularizers

Regularization

Regularization is any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error

Hyperparameters

- Machine learning algorithms usually have two sets of parameters:
  - Hyperparameters: control the algorithm's behavior and are not adapted by the algorithm itself. They often determines the capacity of the model
  - Learnable parameters ("learnable" is often omitted): can be learned from data
- The polynomial regression algorithm $J(\boldsymbol{w}) = \mathrm{MSE}_{\mathrm{train}} + \lambda \boldsymbol{w}^\top \boldsymbol{w}$
  - Hyperparameters: λ
  - Learnable parameters: $\boldsymbol{w}$

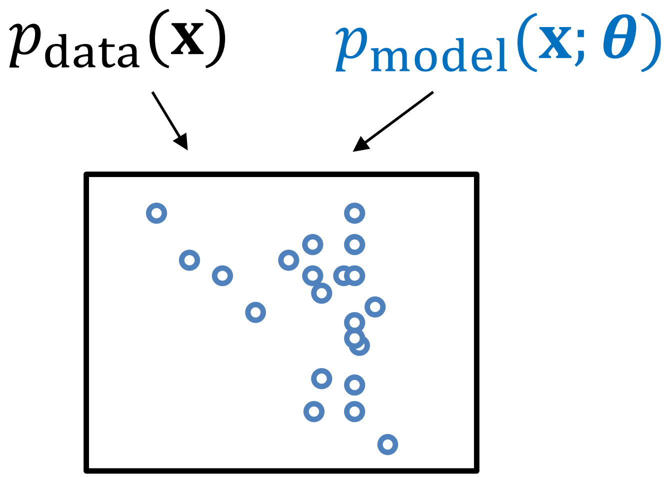
Validation sets

- How to choose the hyperparameters considering that we cannot see the test set?
  - Set them such that the training error is as small as possible?
- We need another set on which the model is not trained on
  - Make the error on this set as small as possible
  - This is called the validation set
- How do we obtain a validation set?

Maximum likelihood estimation (MLE)

Problem definition

- Given a set of N examples $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$ drawn independently from the true but unknown data-generating distribution $p_{\text{data}}(\mathbf{x})$
- Find a prob distribution $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ to approximate $p_{\text{data}}(\mathbf{x})$
- The task is to find optimal $\boldsymbol{\theta}$



Assumption: the observed data samples $\mathbb{X}$ are generated from $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ with the *maximum probability* over all possible $\boldsymbol{\theta}$

$$p_{\text{model}}(\mathbb{X}; \boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \Pi_{i=1}^{N} p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Maximum likelihood estimation (MLE)

Problem definition

- Given a set of N examples $\mathbb{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(N)}\}$ drawn independently from the true but unknown data-generating distribution $p_{\text{data}}(\mathbf{x})$
- Find a prob distribution $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ to approximate $p_{\text{data}}(\mathbf{x})$
- The task is to find optimal $\boldsymbol{\theta}$

- The MLE for $\boldsymbol{\theta}$ is defined as

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\text{model}}(\mathbb{X}; \boldsymbol{\theta}) = \arg\max_{\theta} \Pi_{i=1}^{N} p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

- We usually use

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log p_{\text{model}}(\boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

$$= \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(\boldsymbol{x}; \boldsymbol{\theta})$$

Log-likelihood

  - where $\hat{p}_{\text{data}}$ is the empirical distribution

Conditional log-likelihood

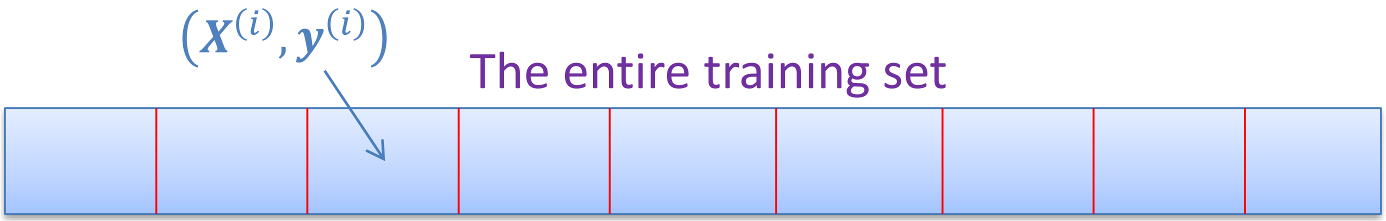
- Estimate a conditional probability $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ in order to predict $\mathbf{y}$ given $\mathbf{x}$
  - E.g. For classification, $\mathbf{y}$ is a (discrete) random variable representing label of an input $\mathbf{x}$
- If $\boldsymbol{X}$ represents all inputs and $\boldsymbol{Y}$ all observed targets, then the conditional maximum likelihood estimator is

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} P_{\mathrm{model}}(\boldsymbol{Y}|\boldsymbol{X};\boldsymbol{\theta})$$

- If the examples are assumed to be i.i.d., then this can be decomposed into

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log P_{\mathrm{model}}(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)};\boldsymbol{\theta})$$

Stochastic gradient decent (SGD)



- Minimizing the cost function over the entire training set is computationally expensive
- We often decompose the training set into smaller subsets or minibatches and optimize the cost function defined over individual minibatches $\left(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}\right)$ and take the average

$$J(\boldsymbol{\theta}) = \frac{1}{N'} \sum_{i=1}^{N'} L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta})$$

$$\boldsymbol{g} = \frac{1}{N'} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{N'} L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}, \boldsymbol{\theta})$$

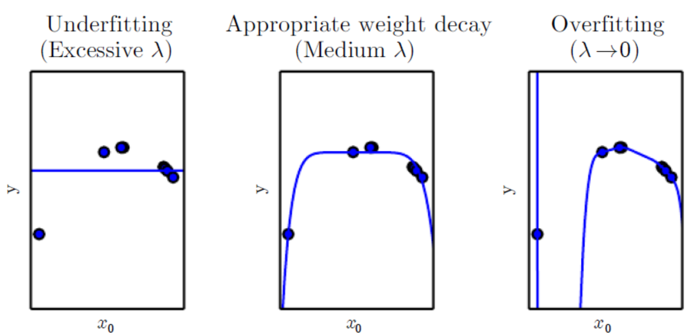
$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \boldsymbol{g}$$

- A total of N' minibatches
- The batchsize ranges from 1 to a few hundreds

Summary of Part 2

- Machine learning basics
  - Task T
    - Classification, regression, synthesis...
  - Performance P
    - Training set, test set
    - Accuracy, error rate, AUC, MAP, human evaluation...
  - Experience E
    - Supervised, unsupervised

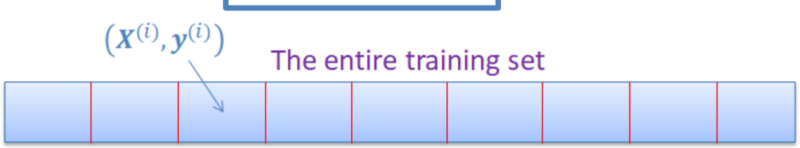
Model capacity



MLE

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} p_{\mathrm{model}}(\mathbb{X}; \boldsymbol{\theta})$$

SGD



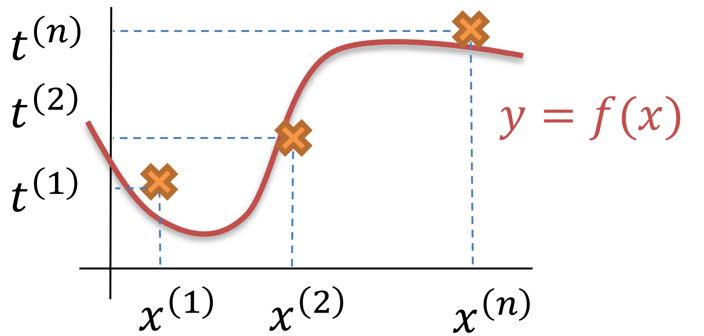
Outline

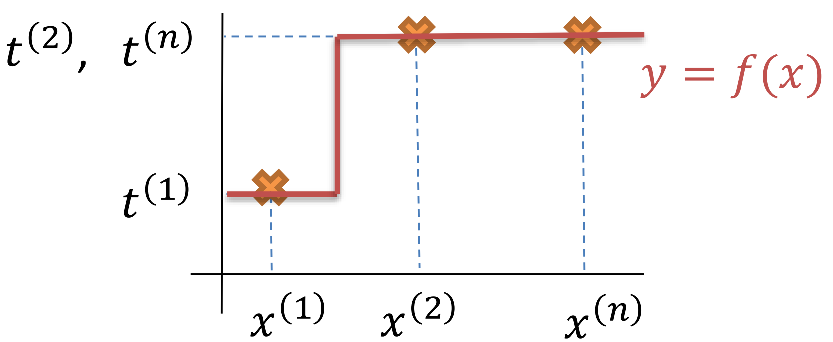
1. Math basics
2. Machine learning basics
3. Regression and classification
4. Summary

Regression and classification

Given a set of data points $\boldsymbol{x}^{(n)} \in R^m$ and the corresponding labels $t^{(n)} \in \Omega$: $\{(\boldsymbol{x}^{(1)}, t^{(1)}), (\boldsymbol{x}^{(2)}, t^{(2)}), \dots, (\boldsymbol{x}^{(N)}, t^{(N)})\}$, for a new data point $\boldsymbol{x}$, predict its label

- The goal is to find a mapping
$$f: R^m \rightarrow \Omega$$
- If Ω is a continuous set, this is called regression
- If Ω is a discrete set, this is called classification



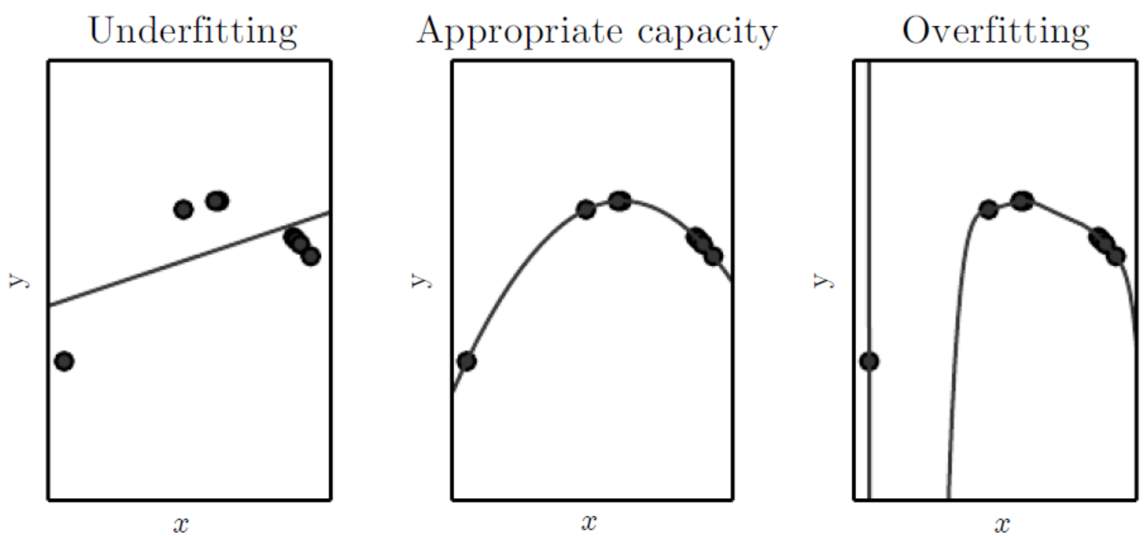


Recall: polynomial regression

- Consider a regression problem in which the input x and output y are both scalars. Find a function $f: \mathbb{R} \rightarrow \mathbb{R}$ to fit the data
  - $f(x) = b + wx$
  - $f(x) = b + w_1 x + w_2 x^2$
  - $f(x) = b + \sum_{i=1}^{9} w_i x^i$

MSE training:

$$\min_{w} \frac{1}{N} \sum_{n=1}^{N} \left\| f\left(x^{(n)}\right) - y^{(n)} \right\|_2^2$$



Linear regression

- $f(\boldsymbol{x})$ is linear

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$$

bias

b can absorbed into a new vector θ and $f(\boldsymbol{x}) = \boldsymbol{\theta}^\top \boldsymbol{x}$

where $\boldsymbol{w} \in R^m, b \in R$.

- Choose the cost function as the mean squared error (MSE)

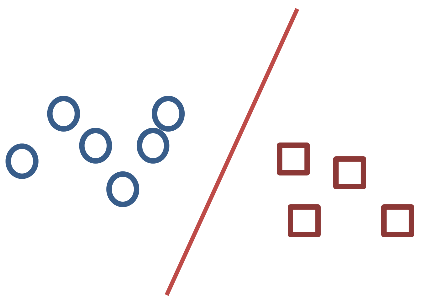
$$E = \frac{1}{2N}\sum_{n=1}^{N}\left(f\left(\boldsymbol{x}^{(n)}\right) - t^{(n)}\right)^2 = \frac{1}{2N}\sum_{n=1}^{N}\left(\boldsymbol{w}^\top \boldsymbol{x}^{(n)} + b - t^{(n)}\right)^2$$

- Find optimal $\boldsymbol{w}^*$ and $\boldsymbol{b}^*$ by minimizing the cost function

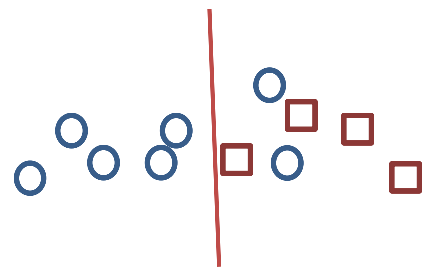
$$\left.\begin{aligned} \nabla_{\boldsymbol{w}} E &= \sum_{n=1}^{N}\left(\boldsymbol{w}^\top \boldsymbol{x}^{(n)} + b - t^{(n)}\right)\boldsymbol{x}^{(n)} = 0 \\ \nabla_b E &= \sum_{n=1}^{N}\left(\boldsymbol{w}^\top \boldsymbol{x}^{(n)} + b - t^{(n)}\right) = 0 \end{aligned}\right\} \quad \boldsymbol{w}^*, b^*$$

Linear classification

- In the feature space, a linear classifier corresponds to a hyperplane



Linearly separable

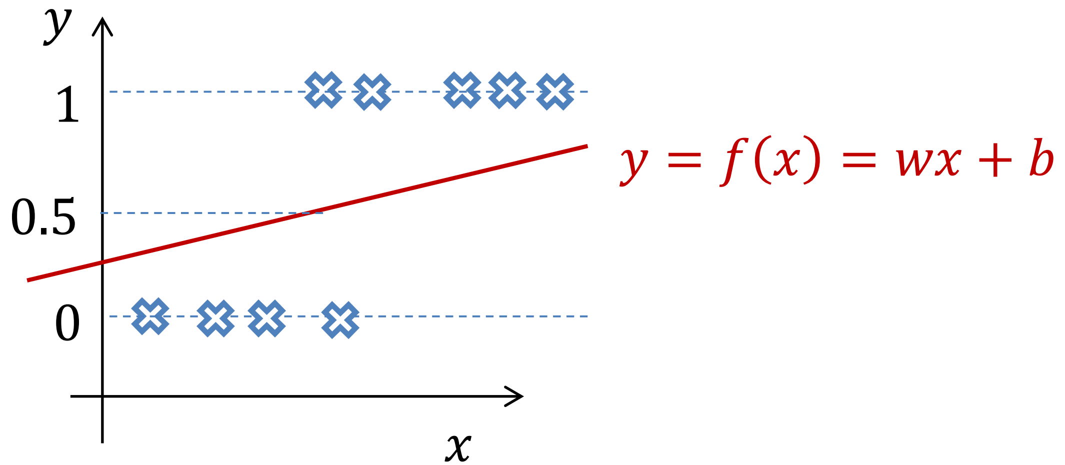


Linearly nonseparable

- Two typical linear classifiers
  - Perceptron
  - SVM

Do binary classification using linear regression

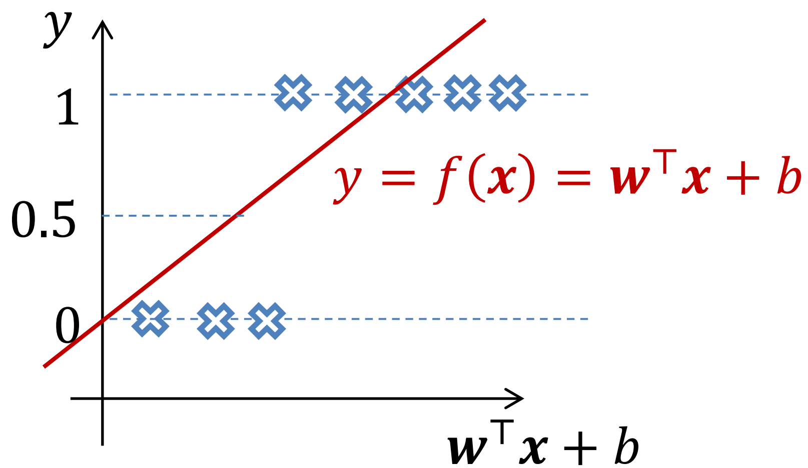
- Suppose $t \in \{0,1\}$. Consider the 1D feature case



- Regression
  - Prediction $y = f(x)$ which is continuous
- Classification
  - Prediction $y = \begin{cases} 1, \text{if } f(x) \geq 0.5 \\ 0, \text{if } f(x) < 0.5 \end{cases}$

How about high dimensional input?

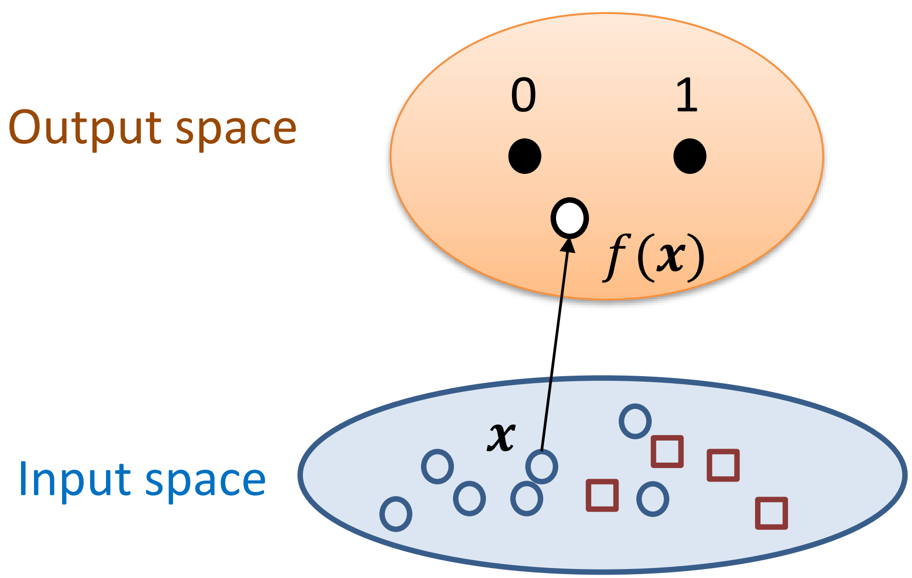
- Suppose $t \in \{0,1\}$. Consider $\boldsymbol{x} \in R^m$, can we use linear regression to do binary classification?



- Regression
  - Prediction $y = f(\boldsymbol{x})$ which is continuous
- Classification
  - Prediction $y = \begin{cases} 1, \text{if } f(\boldsymbol{x}) \geq 0.5 \\ 0, \text{if } f(\boldsymbol{x}) < 0.5 \end{cases}$

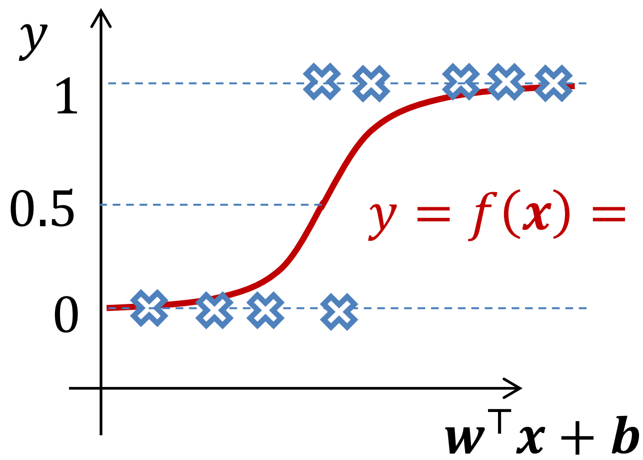
Input-output mapping

Use a linear function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$ to map the input $\boldsymbol{x} \in R^m$ to the 1D output space $\{0,1\}$



Do binary classification using nonlinear regression

- $f(x)$ can be a nonlinear functions, e.g., the logistic sigmoid function



$$y = f(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x} - b)}$$

- Regression
  - Prediction $y = f(\boldsymbol{x})$ which is continuous
- Classification
  - Prediction $y = \begin{cases} 1, \text{if } f(\boldsymbol{x}) \geq 0.5 \\ 0, \text{if } f(\boldsymbol{x}) < 0.5 \end{cases}$

Train the nonlinear regression model

- $f(\boldsymbol{x})$ is nonlinear

$$f(x) = h(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

  where $\boldsymbol{w} \in R^m, b \in R$.

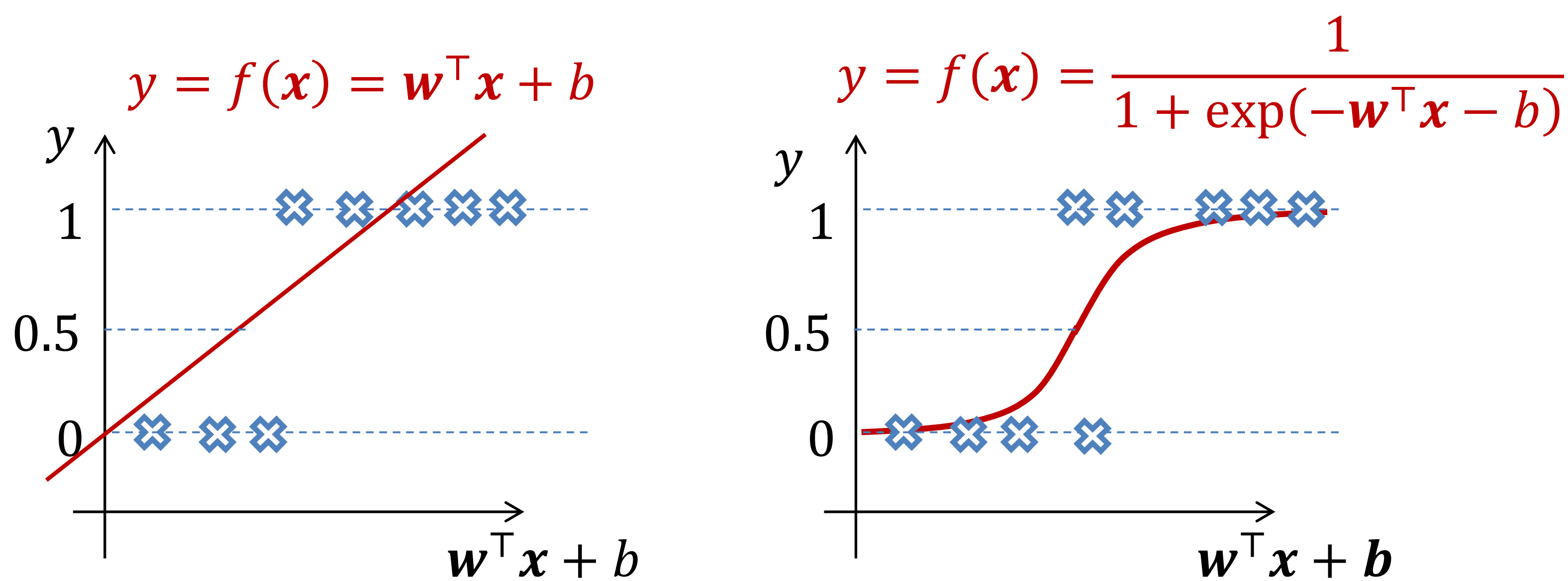
- Choose the cost function as the mean squared error (MSE)

$$E = \frac{1}{2N} \sum_{n=1}^{N} \left(f\left(\boldsymbol{x}^{(n)}\right) - t^{(n)} \right)^2$$

- Find optimal $\boldsymbol{w}^*$ and $\boldsymbol{b}^*$ by minimizing the cost function

$$\left.\begin{aligned} \nabla_{\boldsymbol{w}} E &= \sum_{n=1}^{N} \left(h\left(\boldsymbol{w}^\top \boldsymbol{x}^{(n)} + b\right) - t^{(n)} \right) \boldsymbol{x}^{(n)} = 0 \\ \nabla_b E &= \sum_{n=1}^{N} \left(h\left(\boldsymbol{w}^\top \boldsymbol{x}^{(n)} + b\right) - t^{(n)} \right) = 0 \end{aligned}\right\} \boldsymbol{w}^*, b^*$$

Linear regression VS nonlinear regression



For binary classification, which one do you prefer? Why?

Normal distribution assumption

- Assume the label follows a normal distribution with mean $f(\boldsymbol{x}) = h(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$:

$$p(\mathrm{t}|\mathbf{x}) = \mathcal{N}(\mathrm{t}; f(\mathbf{x})) \propto \exp\left(-||\mathrm{t} - f(\mathbf{x})||_2^2\right)$$

(sometimes we may not distinguish between plain and italic type faces)

- Given a dataset $\left\{\left(\boldsymbol{x}^{(1)}, t^{(1)}\right), \ldots, \left(\boldsymbol{x}^{(N)}, t^{(N)}\right)\right\}$. View t and $\mathbf{x}$ as random variables. The conditional data likelihood function (independence assumption)

$$P\left(t^{(1)}, \ldots, t^{(N)} \middle| \boldsymbol{X}\right) = \Pi_{n=1}^{N} P(t^{(n)} | \boldsymbol{x}^{(n)})$$

- $\max \log P\left(t^{(1)}, \ldots, t^{(N)} \middle| \boldsymbol{X}\right)$ is equivalent to

$$\min E = \frac{1}{2N} \sum_{n=1}^{N} \left\|f\left(\boldsymbol{x}^{(n)}\right) - t^{(n)}\right\|_2^2$$

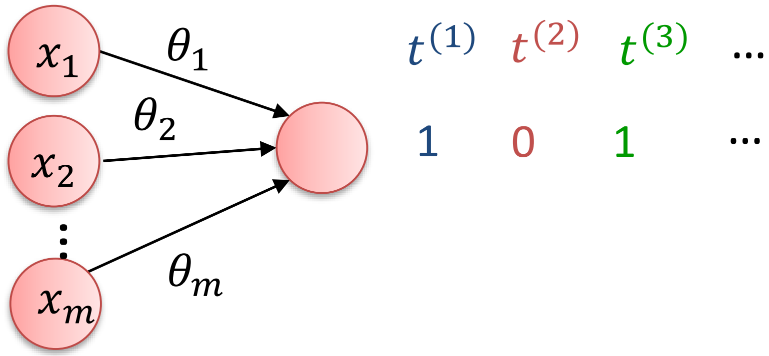
Bernoulli distribution assumption for 2-class classification

$$p(\mathrm{t}|\mathbf{x}) = \mathcal{N}(\mathrm{t}; f(\mathbf{x})) \propto \exp\left(-\left|\left|\mathrm{t} - f(\mathbf{x})\right|\right|_2^2\right)$$

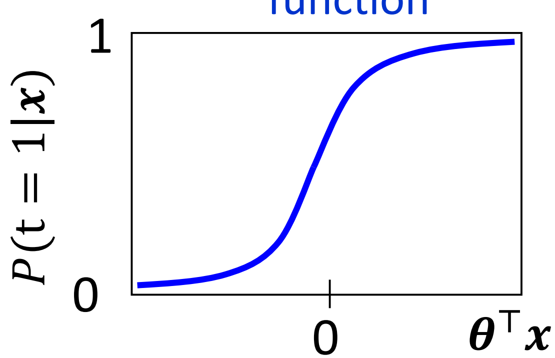
- For regression (t is continuous), the normal distribution assumption is natural
- For classification (t is discrete), it is strange
- We have more suitable assumptions on the data distribution for classification
  - Bernoulli distribution

Logistic regression

- For 2-class problems, one 0-1 unit is enough for representing a label



logistic sigmoid function



- We try to learn a conditional probability (we've absorbed b in θ)

$$P(\mathrm{t}=1|\boldsymbol{x}) = \frac{1}{1+\exp(-\boldsymbol{\theta}^\top \boldsymbol{x})} \triangleq h(\boldsymbol{x})$$

$$P(\mathrm{t}=0|\boldsymbol{x}) = 1 - P(t=1|\boldsymbol{x}) = 1 - h(\boldsymbol{x})$$

$P(\mathrm{t}=1|\boldsymbol{x})$ is a Bernoulli distribution

where $\boldsymbol{x}$ is input and t is label

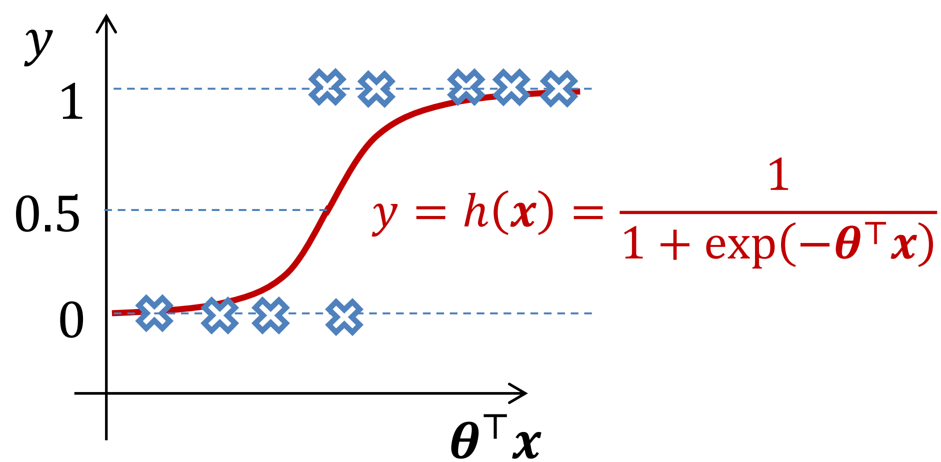
Logistic regression

- Our goal is to search for a value of $\boldsymbol{\theta}$ so that the probability $P(\text{t} = 1|\boldsymbol{x}) = h(\boldsymbol{x})$ is
  - large when $\boldsymbol{x}$ belongs to class 1 and
  - small when $\boldsymbol{x}$ belongs to class 0 (so that $P(t = 0|\boldsymbol{x})$ is large)
- Classification:

$$y = \begin{cases} 1, \text{if } h(\boldsymbol{x}) \geq 0.5 \\ 0, \text{if } h(\boldsymbol{x}) < 0.5 \end{cases}$$

Or equivalently

$$y = \begin{cases} 1, \text{if } \boldsymbol{\theta}^\top \boldsymbol{x} \geq 0 \\ 0, \text{if } \boldsymbol{\theta}^\top \boldsymbol{x} < 0 \end{cases}$$



Maximum conditional data likelihood

- *Recall* the maximum conditional likelihood estimation:
  - 1. write down the conditional likelihood function
  - 2. take log and maximize
- Given a dataset $\{(\boldsymbol{x}^{(1)}, t^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, t^{(N)})\}$ where $t^{(n)} \in \{0,1\}$
- View t as a Bernoulli variable and $P(\mathrm{t} = 1|\boldsymbol{x}) = h(\boldsymbol{x}; \boldsymbol{\theta})$. The conditional likelihood function

$$P(t^{(1)}, \ldots, t^{(N)}|X; \boldsymbol{\theta}) = \Pi_{n=1}^{N} h(\boldsymbol{x}^{(n)})^{t^{(n)}} (1 - h(\boldsymbol{x}^{(n)}))^{1-t^{(n)}}$$

- Maximizing the likelihood is equivalent to minimizing

$$E(\boldsymbol{\theta}) = -\frac{1}{N} \ln P(t^{(1)}, \ldots, t^{(N)})$$

Cross-entropy (CE) function

$$= -\frac{1}{N} \sum_{n=1}^{N} \left(t^{(n)} \ln h(\boldsymbol{x}^{(n)}) + (1 - t^{(n)}) \ln(1 - h(\boldsymbol{x}^{(n)})) \right)$$

Exercise: Calculate the gradient

Cross-entropy (CE) function

$$E(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N}\left(t^{(n)}\ln h(\boldsymbol{x}^{(n)}) + (1-t^{(n)})\ln(1-h(\boldsymbol{x}^{(n)}))\right)$$

$$\nabla E(\boldsymbol{\theta}) = ?$$

$$h(z) = \frac{1}{1+\exp(-z)}$$

$$\frac{\partial h}{\partial z} = h(1-h)$$

Poll 1 answer(s) at most

Setting

Is your result correct?

A Yes

B No

Submit

Training and testing

$$E(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N}\left(t^{(n)}\ln h(\boldsymbol{x}^{(n)}) + (1-t^{(n)})\ln(1-h(\boldsymbol{x}^{(n)}))\right)$$

- Calculate the gradient

$$\nabla E(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n}\boldsymbol{x}^{(n)}\left(h(\boldsymbol{x}^{(n)}) - t^{(n)}\right)$$

- Some regularization term can be incorporated into the cost function

$$J(\boldsymbol{\theta}) = E(\boldsymbol{\theta}) + \lambda\left|\left|\boldsymbol{\theta}\right|\right|^2/2$$

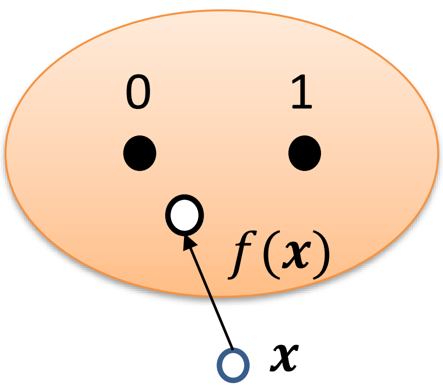
- Training: learn $\boldsymbol{\theta}$ to minimize the cost function

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha\nabla J(\boldsymbol{\theta})$$

  where α is the learning rate

- Testing: for a new input $\boldsymbol{x}$, if $P(t=1|\boldsymbol{x}) > P(t=0|\boldsymbol{x})$ then we predict the input as class 1, and 0 otherwise

Summary
Regression for 2-class classification

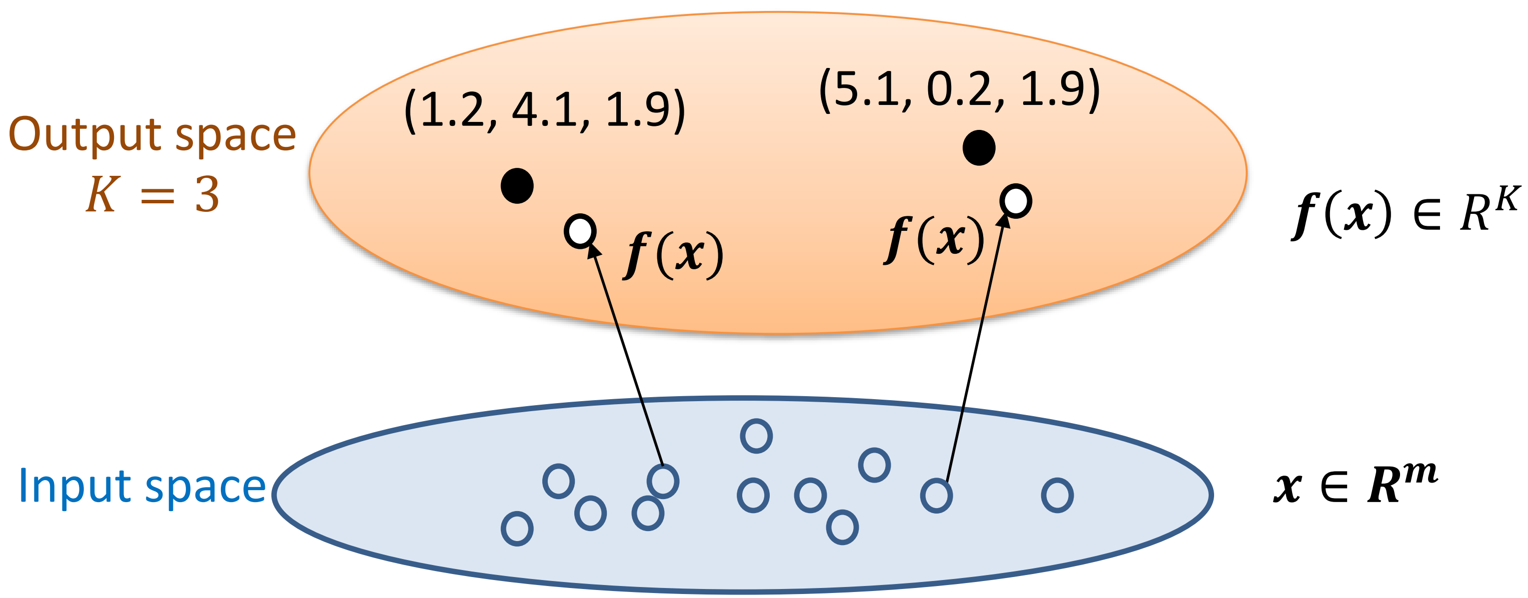


	Linear regression	Nonlinear regression
	$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$	$f(\boldsymbol{x}) = g(\boldsymbol{w}^\top \boldsymbol{x} + b)$ where g is nonlinear 1. MSE always applies 2. If g is the sigmoid function and the CSE is used, it is logistic regression

How about more than 2 classes?

Linear regression for vectors

- If $t \in R^K$ is a continuous vector, then use a linear function $f_k(\boldsymbol{x})$ to regress t_k for $k = 1, \dots, K$: $f_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \boldsymbol{x} + b_k$, where $\boldsymbol{w_k} \in R^m, b_k \in R$



Linear regression for vectors

- $f_k(\boldsymbol{x})$ is linear for $k = 1, \ldots, K$: $f_k(\boldsymbol{x}) = \boldsymbol{w}_k^\top \boldsymbol{x} + b_k$, where $\boldsymbol{w_k} \in R^m, b_k \in R$.
- Choose the cost function as the mean squared error (MSE)

$$E = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(f_k\left(\boldsymbol{x}^{(n)}\right) - t_k^{(n)} \right)^2$$

- Find optimal $\boldsymbol{w}_k^*$ and $\boldsymbol{b}_k^*$ by solving the linear system

$$\left.\begin{aligned}
\nabla_{\boldsymbol{w_k}} E &= \frac{1}{N} \sum_{n=1}^{N} \left(\boldsymbol{w}_k^\top \boldsymbol{x}^{(n)} + b_k - t_k^{(n)} \right) \boldsymbol{x}^{(n)} = 0 \\
\nabla_{b_k} E &= \frac{1}{N} \sum_{n=1}^{N} \left(\boldsymbol{w}_k^\top \boldsymbol{x}_k^{(n)} + b_k - t_k^{(n)} \right) = 0
\end{aligned}\right\} \boldsymbol{w}_k^*, b_k^*$$

Nonlinear regression for vectors

- $f_k(\boldsymbol{x})$ is nonlinear for $k = 1, \dots, K$

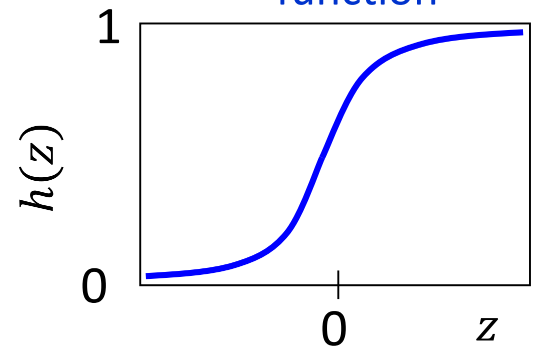
$$f_k(\boldsymbol{x}) = h\left(\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right)$$

where $\boldsymbol{w}_k \in R^m$, $b_k \in R$, and

$$h(z) = \frac{1}{1 + \exp(-z)}$$

Or other functions

logistic sigmoid function



- Choose the cost function as the MSE

$$E = \frac{1}{2N} \sum_{n=1}^{N} \sum_{k=1}^{K} \left(f_k\left(\boldsymbol{x}^{(n)}\right) - t_k^{(n)} \right)^2$$

Local sensitivity or local gradient

- Denote $u_k = \boldsymbol{w}_k^\top \boldsymbol{x} + b_k$ and $\boxed{\delta_k = \frac{\partial E}{\partial u_k},}$ then $\frac{\partial E}{\partial \boldsymbol{w}_k} = \delta_k \frac{\partial u_k}{\partial \boldsymbol{w}_k}$ and $\frac{\partial E}{\partial b_k} = \delta_k \frac{\partial u_k}{\partial b_k} = \delta_k$

$$\delta_k = (f(u_k) - t_k) f'(u_k)$$

Vector-matrix form

- Define

$$\boldsymbol{W} = \begin{pmatrix} w_{11} & \cdots & w_{1m} \\ \vdots & \vdots & \vdots \\ w_{K1} & \cdots & w_{Km} \end{pmatrix} \quad \frac{\partial E}{\partial \boldsymbol{W}} = \begin{pmatrix} \partial E/\partial w_{11} & \cdots & \partial E/\partial w_{1m} \\ \vdots & \vdots & \vdots \\ \partial E/\partial w_{K1} & \cdots & \partial E/\partial w_{Km} \end{pmatrix}$$

m: the number of inputs; K: the number of outputs

- Output: $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$, where $\boldsymbol{f}, \boldsymbol{h}, \boldsymbol{b} \in R^K$, $\boldsymbol{x} \in R^m$
- Error function: $E = \frac{1}{2N}\sum_{n=1}^{N} \left\| \boldsymbol{f}\left(\boldsymbol{x}^{(n)}\right) - \boldsymbol{t}^{(n)} \right\|_2^2$
- Gradient

Elementwise product

$$\nabla_{\boldsymbol{W}} E = \frac{1}{N}\sum_{n=1}^{N} \left(\boldsymbol{f}\left(\boldsymbol{x}^{(n)}\right) - \boldsymbol{t}^{(n)} \right) \odot \boldsymbol{f}'\left(\boldsymbol{x}^{(n)}\right) \left(\boldsymbol{x}^{(n)}\right)^{\top} \in R^{K \times m}$$

$$\nabla_{\boldsymbol{b}} E = \frac{1}{N}\sum_{n=1}^{N} \left(\boldsymbol{f}\left(\boldsymbol{x}^{(n)}\right) - \boldsymbol{t}^{(n)} \right) \odot \boldsymbol{f}'\left(\boldsymbol{x}^{(n)}\right) \qquad \in R^{K}$$

Representation of class labels

- For classification, given $\{(\boldsymbol{x}^{(1)}, t^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, t^{(N)})\}$, the goal is to find a mapping from $\boldsymbol{x}^{(n)}$ to $t^{(n)}$

$$f: R^m \to \Omega$$

where Ω is a discrete set

- $t^{(n)}$ can be a (discrete) scalar or vector

Suppose there are 5 classes in total

Rarely used

Scalar representation

$$t^{(1)} = 1$$
$$t^{(3)} = 3$$

Vector representation

$$\boldsymbol{t}^{(1)} = (1, 0, 0, 0, 0)^\mathsf{T}$$
$$\boldsymbol{t}^{(3)} = (0, 0, 1\ 0, 0)^\mathsf{T}$$

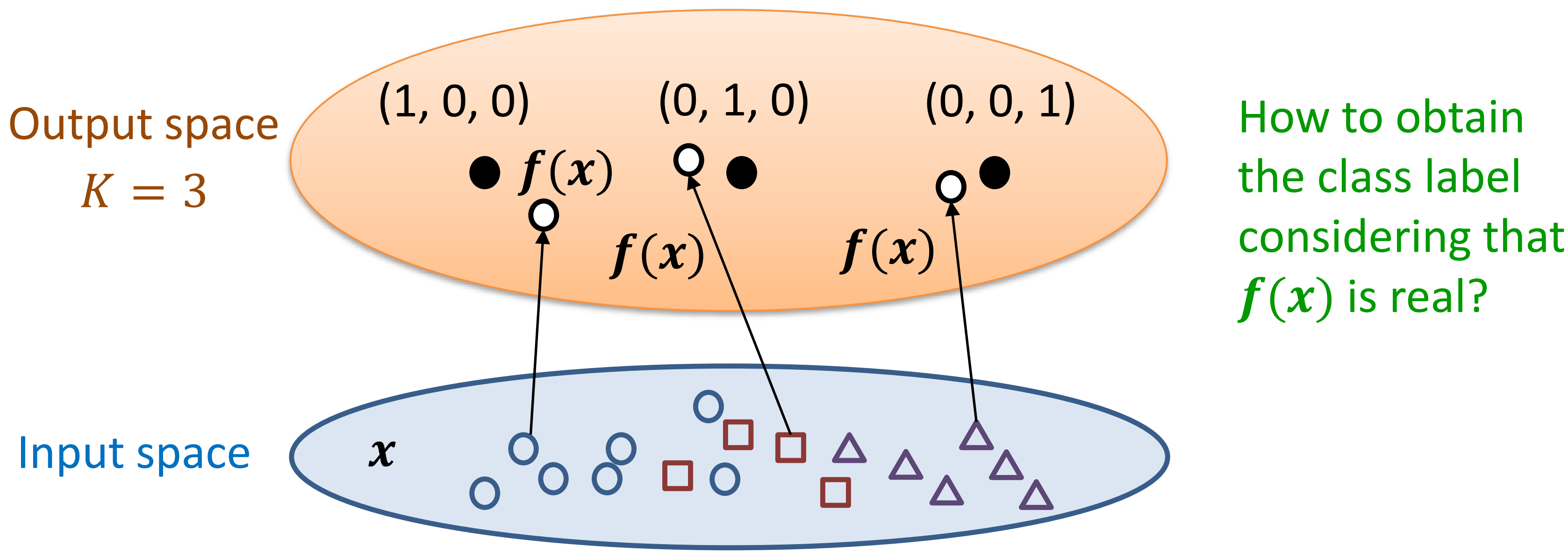
Usually used

- 1-of-K representation
- Property: $t_k^{(n)} \in \{0,1\}$; $\sum_k t_k^{(n)} = 1$

Any problem with scalar representation?

Do multilabel classification using regression

- Using the 1-of-K representation for class labels, one can also do classification using linear regression



- Both linear and nonlinear regression discussed before can be applied

Normal distribution assumption

- Assume the label follows a normal distribution with mean $\boldsymbol{f}(\mathbf{x}) = \boldsymbol{h}(\boldsymbol{W}\mathbf{x} + \boldsymbol{b})$:

$$p(\mathbf{t}|\mathbf{x}) = \mathcal{N}(\mathbf{t}; \boldsymbol{f}(\mathbf{x}), \boldsymbol{I}) \propto \exp\left(-\left|\left|\mathbf{t} - \boldsymbol{f}(\mathbf{x})\right|\right|_2^2\right)$$

(sometimes we may not distinguish between plain and italic type faces)

- Given a dataset $\left\{\left(\boldsymbol{x}^{(1)}, \boldsymbol{t}^{(1)}\right), \dots, \left(\boldsymbol{x}^{(N)}, \boldsymbol{t}^{(N)}\right)\right\}$. View $\mathbf{t}$ and $\mathbf{x}$ as random variables. The conditional data likelihood function (independence assumption)

$$P\left(\boldsymbol{t}^{(1)}, \dots, \boldsymbol{t}^{(N)} \middle| \boldsymbol{X}\right) = \Pi_{n=1}^{N} P(\boldsymbol{t}^{(n)} | \boldsymbol{x}^{(n)})$$

- $\max \log P\left(\boldsymbol{t}^{(1)}, \dots, \boldsymbol{t}^{(N)} \middle| \boldsymbol{X}\right)$ is equivalent to

$$\min E = \frac{1}{2N} \sum_{n=1}^{N} \left|\left|\boldsymbol{f}\left(\boldsymbol{x}^{(n)}\right) - \boldsymbol{t}^{(n)}\right|\right|_2^2$$

Multinoulli distribution assumption for classification

$$p(\mathbf{t}|\mathbf{x}) = \mathcal{N}(\mathbf{t}; \boldsymbol{f}(\mathbf{x}), \boldsymbol{I}) \propto \exp\left(-\left|\left|\mathbf{t} - \boldsymbol{f}(\mathbf{x})\right|\right|_2^2\right)$$

- For regression (t is continuous), the normal distribution assumption is natural
- For classification (t is discrete), it is strange
- We have more reasonable assumptions on the data distribution for classification
  - Bernoulli distribution for $K = 2$
  - Multinoulli or categorical distribution for $K > 2$

Recap: multinoulli or categorical prob distributions

- The prob distribution over a single discrete variable t with K different states where K is finite

$$P(\mathrm{t} = k|\boldsymbol{p}) = p_k$$

where $\boldsymbol{p} \in [0,1]^K$ and $\sum_{k=1}^{K} p_k = 1$

- With one-hot representation $\mathbf{t} = (0, \dots, 1, \dots, 0)^\top$, then $P(\mathbf{t}) = \Pi_{k=1}^{K} P(\mathrm{t}_k = 1)^{t_k}$

A random variable A value 0 or 1

Scalar: 1,2,3

$P(\mathrm{t} = 1) = 0.2;$
$P(\mathrm{t} = 2) = 0.5;$
$P(\mathrm{t} = 3) = 0.3$

One-hot: $(1,0,0)^\top, (0,1,0)^\top, (1,0,0)^\top$

$P(\mathbf{t} = (1,0,0)^\top)$
$= P(\mathrm{t}_1 = 1)^1 P(\mathrm{t}_2 = 1)^0 P(\mathrm{t}_3 = 1)^0 = 0.2$

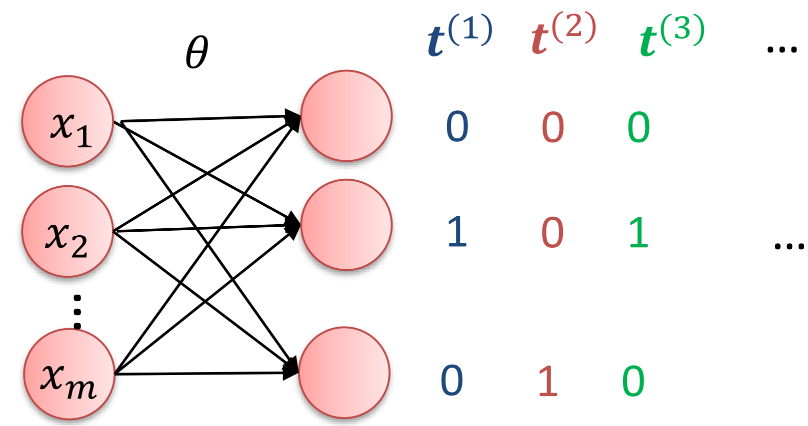
Similarly, $P(\mathbf{t} = (0,1,0)^\top) = 0.5;$
$P(\mathbf{t} = (0,0,1)^\top) = 0.3$

Formulation

- For K-class problems ($K > 2$), we try to learn a Multinoulli distribution $P(\mathrm{t} = k|\boldsymbol{x})$ where $k = 1, \dots, K$
- With one-hot representation $\mathbf{t} = (0, \dots, 1, \dots, 0)^\top$, then

$$P(\mathbf{t}|\boldsymbol{x}) = \Pi_{k=1}^{K} P(\mathrm{t}_k = 1|\boldsymbol{x})^{t_k}$$

A random variable A value



- Let $P(\mathrm{t}_k = 1|\boldsymbol{x})$ take the following form

$$P(\mathrm{t}_k = 1|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^{(k)\top}\boldsymbol{x})}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}^{(j)\top}\boldsymbol{x})} \triangleq h_k(\boldsymbol{x})$$

Clearly, $h_k(\boldsymbol{x}) \in (0,1)$ and $\sum_{k=1}^{K} h_k(\boldsymbol{x}) = 1$

Formulation

$$P(\mathrm{t}_k = 1|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^{(k)\top}\boldsymbol{x})}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}^{(j)\top}\boldsymbol{x})} \triangleq h_k(\boldsymbol{x})$$

- Given a test input $\boldsymbol{x}$, estimate $P(\mathrm{t}_k = 1|\boldsymbol{x})$ for each value of $k = 1, \ldots, K$.
- Goal: search for a value of $\boldsymbol{\theta}$ so that the probability $P(\mathrm{t}_k = 1|\boldsymbol{x})$ is
  - large when $\boldsymbol{x}$ belongs to the k-th class and
  - small when $\boldsymbol{x}$ belongs to other classes

where
$$\boldsymbol{\theta} = \begin{bmatrix} | & | & & | \\ \boldsymbol{\theta}^{(1)} & \boldsymbol{\theta}^{(2)} & \cdots & \boldsymbol{\theta}^{(K)} \\ | & | & & | \end{bmatrix}^{\top}.$$

- Since $h_k(\boldsymbol{x})$ is a (continuous) probability, we need to transform it into discrete values for classification ←How?

Softmax function

$$h_k(\boldsymbol{x}) = P(\mathrm{t}_k = 1|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^{(k)\top}\boldsymbol{x})}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}^{(j)\top}\boldsymbol{x})}$$

- The following function is called *softmax* function

$$\psi(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} = \frac{\exp(z_i)}{\exp(z_i) + \sum_{j\neq i}\exp(z_j)} \in (0,1)$$

- If $z_i > z_j$ for all $j \neq i$
  - Then $\psi(z_i) > \psi(z_j)$ for all $j \neq i$ but it is smaller than 1
- If $z_i \gg z_j$ for all $j \neq i$,
  - then $\psi(z_i) \to 1$ and $\psi(z_j) \to 0$ for $j \neq i$.

Maximum conditional likelihood

- Since

$$P(\mathbf{t}|\boldsymbol{p}) = \Pi_{k=1}^{K} P(\mathrm{t}_k = 1)^{t_k}$$

- Given a dataset $\{(\boldsymbol{x}^{(1)}, \boldsymbol{t}^{(1)}), \ldots, (\boldsymbol{x}^{(N)}, \boldsymbol{t}^{(N)})\}$. The conditional likelihood function (independence assumption):

$$P\left(\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)} \middle| \boldsymbol{X}\right) = \Pi_{n=1}^{N} \Pi_{k=1}^{K} P(\mathrm{t}_k^{(n)} = 1 | \boldsymbol{x}^{(n)})^{t_k^{(n)}}$$

- Maximizing this likelihood function is equivalent to minimizing

Take log and negative, then minimize

$$E(\boldsymbol{\theta}) = -\frac{1}{N} \ln P(\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)})$$

$$= -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \ln \boxed{\frac{\exp(\boldsymbol{\theta}^{(k)\top} \boldsymbol{x}^{(n)})}{\sum_{j=1}^{K} \exp(\boldsymbol{\theta}^{(j)\top} \boldsymbol{x}^{(n)})}}$$

$h_k^{(n)}$

Cross-entropy function

$$= -\frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \ln h_k^{(n)} = -\frac{1}{N} \sum_{n=1}^{N} \ln P(\mathrm{t}_k^{(n)} = 1 | \boldsymbol{x}^{(n)})$$

Softmax is over-parameterized

- The hypothesis

$$h_k(\boldsymbol{x}) = P(\mathrm{t}_k = 1|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^{(k)\top}\boldsymbol{x})}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}^{(j)\top}\boldsymbol{x})} = \frac{\exp((\boldsymbol{\theta}^{(k)} - \boldsymbol{\phi})^\top \boldsymbol{x})}{\sum_{j=1}^{K}\exp((\boldsymbol{\theta}^{(j)} - \boldsymbol{\phi})^\top \boldsymbol{x})}$$

  Then the new parameters $\widehat{\boldsymbol{\theta}}^{(k)} \equiv \boldsymbol{\theta}^{(k)} - \boldsymbol{\phi}$ will result in the same prediction

- Minimizing the cross-entropy function has infinite number of solutions since

$$E(\boldsymbol{\theta}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^{(n)} \ln \frac{\exp(\boldsymbol{\theta}^{(k)\top}\boldsymbol{x}^{(n)})}{\sum_{j=1}^{K}\exp(\boldsymbol{\theta}^{(j)\top}\boldsymbol{x}^{(n)})} = E(\boldsymbol{\theta} - \boldsymbol{\Phi})$$

  where $\boldsymbol{\Phi} = (\boldsymbol{\phi}, ..., \boldsymbol{\phi})$

Relationship between softmax regression and logistic regression

Let $K = 2$ in softmax

Sigmoid function

- The hypotheses

$$h_1(\boldsymbol{x}) = P(\mathrm{t}_1 = 1|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^{(1)\top}\boldsymbol{x})}{\exp(\boldsymbol{\theta}^{(1)\top}\boldsymbol{x}) + \exp(\boldsymbol{\theta}^{(2)\top}\boldsymbol{x})} = \sigma(\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(2)})$$

$$h_2(\boldsymbol{x}) = P(\mathrm{t}_2 = 1|\boldsymbol{x}) = \frac{\exp(\boldsymbol{\theta}^{(2)\top}\boldsymbol{x})}{\exp(\boldsymbol{\theta}^{(1)\top}\boldsymbol{x}) + \exp(\boldsymbol{\theta}^{(2)\top}\boldsymbol{x})} = 1 - \sigma(\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(2)})$$

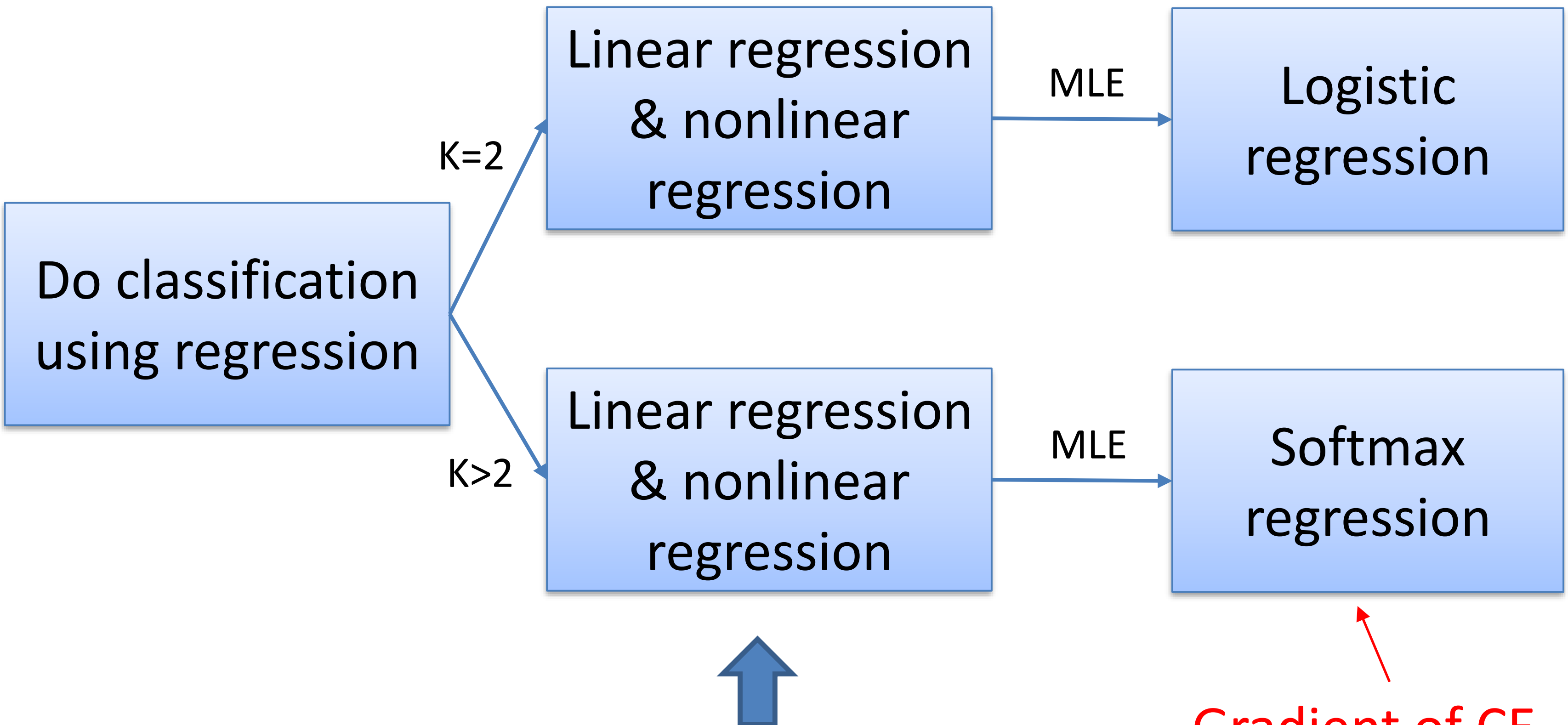
The same as in the two-unit version of the logistic regression if we define a new variable $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(2)}$.

- The error function for each sample

$$E^{(n)}(\boldsymbol{\theta}) = -t_1^{(n)} \ln h_1^{(n)} - t_2^{(n)} \ln h_2^{(n)} = -t_1^{(n)} \ln h_1^{(n)} - (1 - t_1^{(n)}) \ln(1 - h_1^{(n)})$$

The same as in the logistic regression

Summary of Part 3

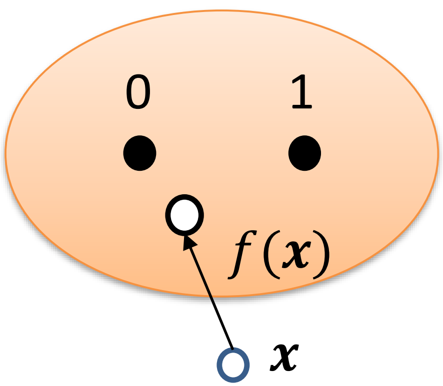
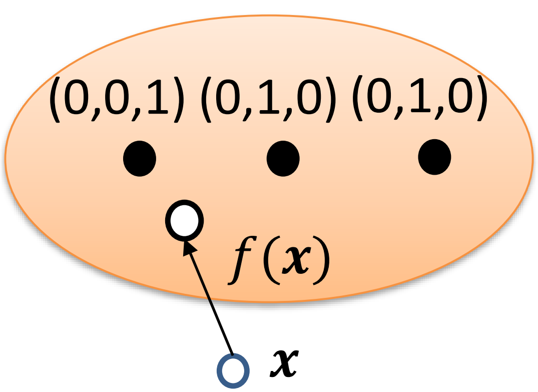


If the task is regression, then we stop here

Gradient of CE loss?

Summary of Part 3

Regression for multi-class classification

	Linear regression	Nonlinear regression
	$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$	$f(\boldsymbol{x}) = g(\boldsymbol{w}^\top \boldsymbol{x} + b)$ where g is nonlinear 1. MSE always applies 2. If g is the sigmoid function and the CE is used, it is logistic regression
	$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b}$	$\boldsymbol{f}(x) = \boldsymbol{g}(\boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b})$ where $\boldsymbol{g}$ is nonlinear 1. MSE always applies 2. If $\boldsymbol{g}$ is the softmax function and the CE is used, it is softmax regression

Outline

1. Math basics
2. Machine learning basics
3. Regression and classification
4. Summary

Summary of this lecture

Knowledge

1. Math basics

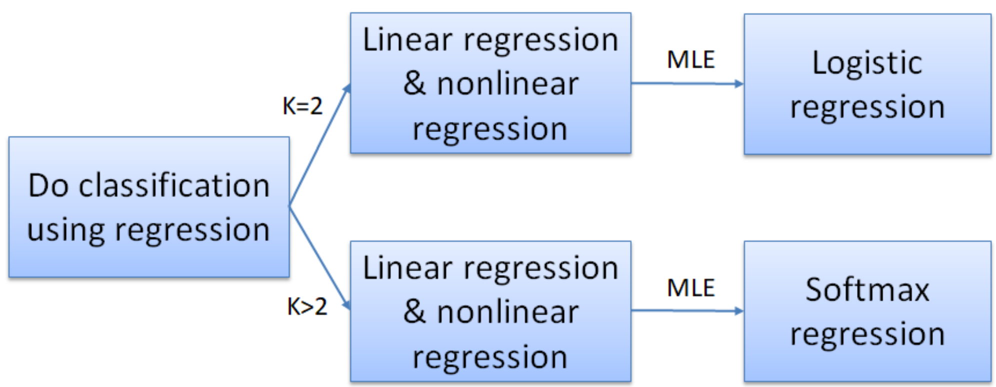
Linear algebra | Probability theory | Optimization

2. Machine learning basics

Task T | Performance P | Experience E

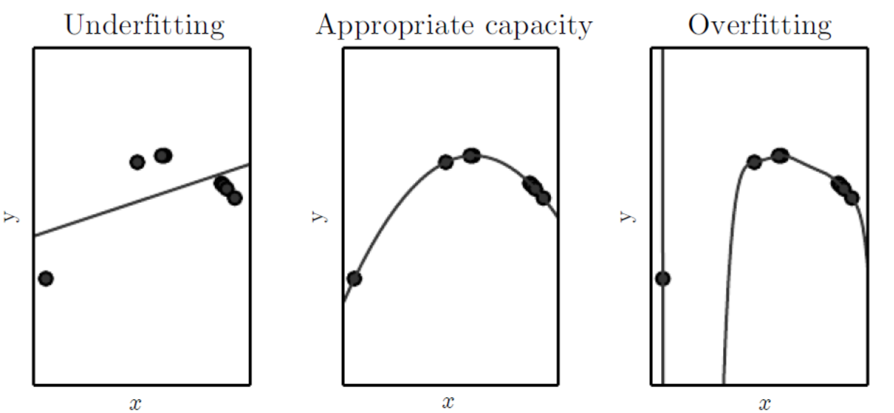
Supervised VS unsupervised | Model capacity | MLE | SGD

3. Regression and classification



Open question

- Nonlinear fitting
  - $f(x) = b + wx$
  - $f(x) = b + w_1 x + w_2 x^2$
  - $f(x) = b + \sum_{i=1}^{9} w_i x^i$



Model	Number of params
AlexNet	60 M
ResNeXt-101	44.3M
GPT2	1.5B
GPT3	175B

Why don't DL models seem to have overfitting problem?

References

- Chapters 2-5 in Deep Learning by Goodfellow, Bengio and Courville, 2016, MIT Press