INTRODUCTION TO DEEP LEARNING

Course number: 00240332

Lecture 2: Math and Machine Learning Basics

Xiaolin Hu (胡晓林)

Dept. of Computer Science and Technology

Tsinghua University

Outline

I. Math basics
II. Machine learning basics
III. Summary

Math objects

But I sometimes may not follow these conventions

- Scalar
  - A single number, often denoted by a lower case letter without boldface, e.g., a, b, x
- Vector
  - An array of numbers, often denoted by a lowercase letter with boldface, e.g., $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{x}$

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

- Matrix
  - A 2D array of numbers, often denoted by an uppercase letter with boldface, e.g., $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{X}$

$$\boldsymbol{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}$$

- Tensor
  - An n-D array of numbers, often denoted like this: e.g., $\mathbf{A}, \mathbf{B}, \mathbf{X}$

$$\mathbf{A} = \left(\begin{pmatrix} A_{1,1,1} & A_{1,2,1} \\ A_{2,1,1} & A_{2,2,1} \end{pmatrix}, \begin{pmatrix} A_{1,1,2} & A_{1,2,2} \\ A_{2,1,2} & A_{2,2,2} \end{pmatrix} \right)$$

Simple operations

- Matrix transpose: $\boldsymbol{A}^\top$

$$\boldsymbol{A} = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \qquad \boldsymbol{A}^\top = \begin{pmatrix} A_{1,1} & A_{2,1} \\ A_{1,2} & A_{2,2} \end{pmatrix}$$

- A vector can be viewed as a special matrix

$$\boldsymbol{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \qquad \boldsymbol{a}^\top = (a_1, a_2, a_3)$$

- Matrix product: if $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{n \times p}$, then $\boldsymbol{C} = \boldsymbol{AB}$ with shape $m \times p$ and $C_{i,j} = \sum_k A_{i,k} B_{k,j}$
- Elementwise product (Hadamard product): $\boldsymbol{C} = \boldsymbol{A} \odot \boldsymbol{B}$ where the 3 matrices are of the same shape and $C_{i,j} = A_{i,j} B_{i,j}$

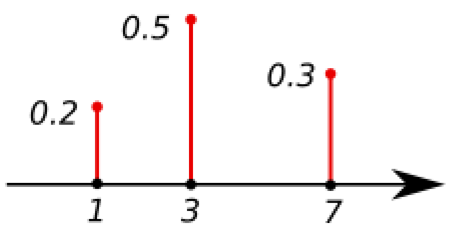
Random variable

- A random variable is a variable that can take on different values randomly
  - Denote the random variable by x and its two possible values by x_1 and x_2
  - For vectors, we write the random variable as $\mathbf{x}$ and one of its values as $\boldsymbol{x}$
  - Discrete versus continuous

Probability distribution

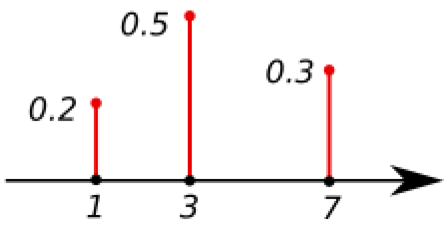
A probability distribution is a distribution of how likely a random variable or a set of random variables is to take on each of its possible states

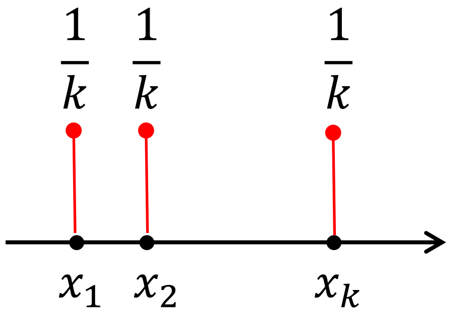
- A probability distribution over discrete variables may be described using a probability mass function (PMF)
  - The prob that $\mathrm{x} = x$ is denoted as $P(x)$ or $P(\mathrm{x} = x)$
  - $\mathrm{x} \sim P(x)$ specify which distribution x follows
- Joint probability
  - $P(\mathrm{x} = x, \mathrm{y} = y)$ or $P(x, y)$ denotes the prob that $\mathrm{x} = x$ and $\mathrm{y} = y$ simultaneously



Probability mass function

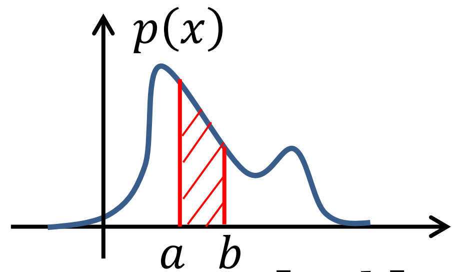
- To be a PMF of a random variable x, a function P must satisfy:
  - The domain of P must be the set of all possible states of x
  - $\forall x \in \mathrm{x}, 0 \leq P(x) \leq 1$
  - $\sum_{x \in \mathrm{x}} P(x) = 1$
- Uniform distribution
  - Consider a single discrete random variable x with k different states
  - $P(\mathrm{x} = x_i) = \frac{1}{k}, \forall\, i$





Probability density function

- A probability distribution over continuous variables may be described using a probability density function (PDF)
- To be a PDF, a function p must satisfy the following properties
  - The domain of p must be the set of all possible states of x
  - $\forall x \in \mathrm{x}, p(x) \geq 0$
  - $\int p(x)dx = 1$



Can $p(x) > 1$?

- The prob that x lies in the interval $[a, b]$ is given by $\int_a^b p(x)dx$
- Note $p(x)$ does not give the prob of a specific state directly

Typical prob distributions

- Bernoulli distribution: over a single binary random variable

$$P(\mathrm{x}=1)=\phi, P(\mathrm{x}=0)=1-\phi$$
$$P(\mathrm{x}=x)=\phi^x(1-\phi)^{1-x}$$
$$\mathbb{E}_{\mathrm{x}}[\mathrm{x}]=\phi, \mathrm{Var}(\mathrm{x})=\phi(1-\phi)$$

- Multinoulli or categorical distribution: over a single discrete variable with k different states where k is finite

$$P(\mathrm{x}=i|\boldsymbol{p})=p_i$$

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

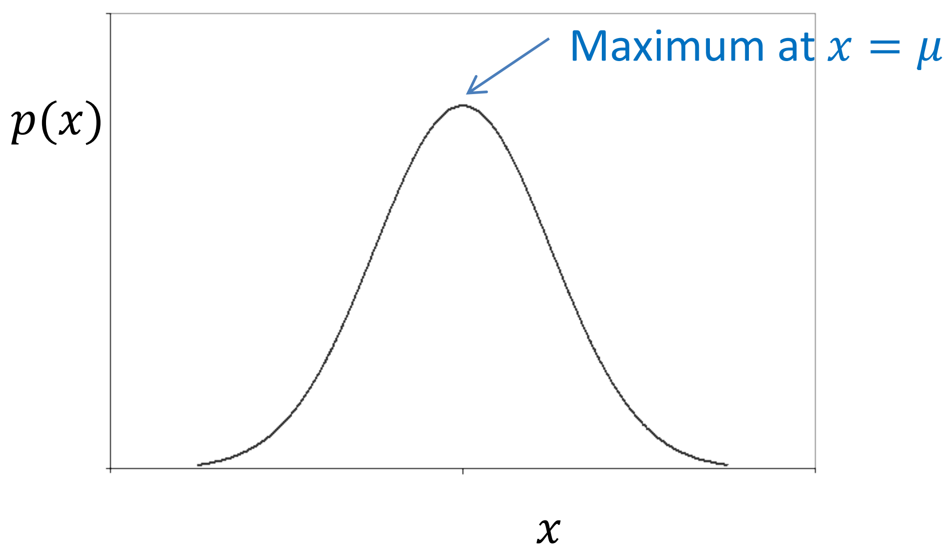
Typical prob distributions

- Gaussian distribution or normal distribution: over a continuous variable

$$\mathcal{N}(x;\mu,\sigma^2)=\sqrt{\frac{1}{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

  - Mean: $\mathbb{E}[\mathrm{x}]=\mu$
  - Variance: $\mathrm{Var}[\mathrm{x}]=\sigma^2$
  - Standard deviation: σ

The *central limit theorem* shows that the sum of many independent variables is approximately normally distributed



Marginal probability

Suppose we know the prob distribution over a set of variables. The prob distribution over just a subset of them is known as the marginal prob distribution

- Let $P(\mathrm{x}, \mathrm{y})$ denote the prob distribution of discrete random variables x and y, then

$$P(\mathrm{x} = x) = \sum_{y} P(\mathrm{x} = x, \mathrm{y} = y)$$

- Let $p(\mathrm{x}, \mathrm{y})$ denote the PDF of continuous random variables x and y, then

$$p(x) = \int P(x, y) dy$$

Conditional probability

The conditional probability is the probability of some event, given that some other event has happened

- The conditional prob that $\mathrm{y} = y$ given $\mathrm{x} = x$ is denoted by $P(\mathrm{y} = y | \mathrm{x} = x)$, which can be calculated as
$$P(\mathrm{y} = y | \mathrm{x} = x) = P(\mathrm{y} = y, \mathrm{x} = x) / P(\mathrm{x} = x)$$
- The chain rule
$$P\left(\mathrm{x}^{(1)}, \dots, \mathrm{x}^{(n)}\right) = P\left(\mathrm{x}^{(n)}\right) \Pi_{i=1}^{n-1} P\left(\mathrm{x}^{(i)} \middle| \mathrm{x}^{(i+1)}, \dots, \mathrm{x}^{(n)}\right)$$
- Exercise: Is the following correct?
$$P(\mathrm{a}, \mathrm{b}, \mathrm{c}) = P(\mathrm{a} | \mathrm{b}, \mathrm{c}) P(\mathrm{b} | \mathrm{c}) P(\mathrm{c})$$

Expectation

The expectation, or expected value, of some function $f(x)$ w.r.t. a prob distribution $P(\mathrm{x})$ is the average value that f takes on when x is drawn from P

- For discrete variables

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \sum_{x} P(x)f(x)$$

- For continuous variables

$$\mathbb{E}_{\mathrm{x}\sim P}[f(x)] = \int p(x)f(x)dx$$

- If the identity of the distribution is clear, we may write $\mathbb{E}_{\mathrm{x}}[f(x)]$
- Expectation is linear: if α and β do not depend on x, then

$$\mathbb{E}_{\mathrm{x}}[\alpha f(x) + \beta g(x)] = \alpha \mathbb{E}_{\mathrm{x}}[f(x)] + \beta \mathbb{E}_{\mathrm{x}}[g(x)]$$

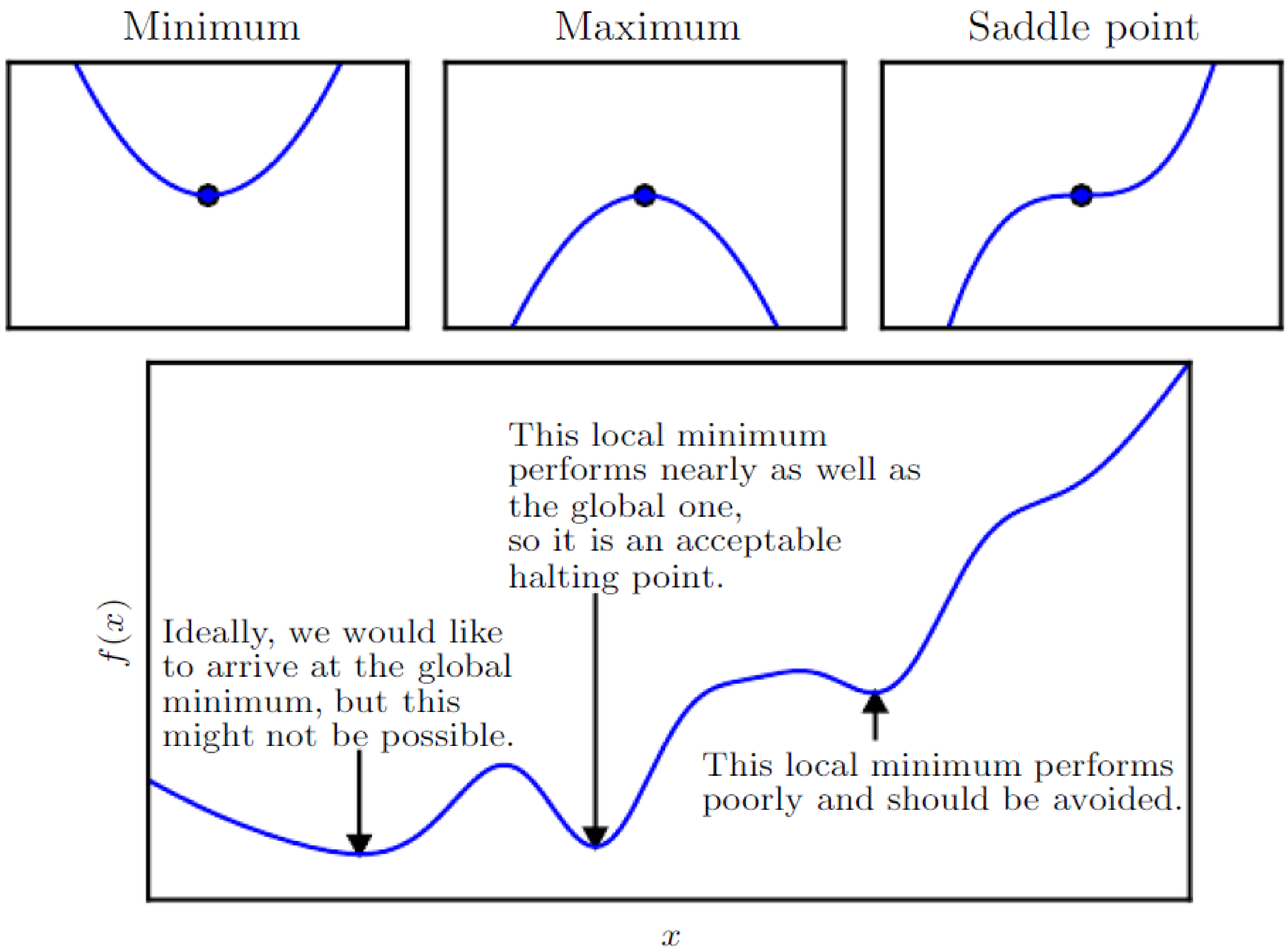
Gradient-based optimization

- The function we want to minimize or maximize is called objective function
- When we are minimizing it, we may also call it the cost function, loss function, or error function
- The derivative of a function $y = f(x)$, denoted by $f'(x)$ or $\frac{dy}{dx}$, gives the slope, or gradient, of f at the point x
- Gradient descent

$$x' = x - \eta f'(x)$$

where $\eta > 0$ is the learning rate

Critical points



Gradient decent for multivariate functions

- For a function of a single variable $y = f(x)$, the gradient decent method is
$$x' = x - \eta f'(x)$$
- For a function $y = f(\boldsymbol{x})$, the partial derivative is denoted by $\partial f / \partial x_i$
- The gradient decent method becomes
$$\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

where $\eta > 0$ and $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \begin{pmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \dots \\ \partial f / \partial x_n \end{pmatrix}$

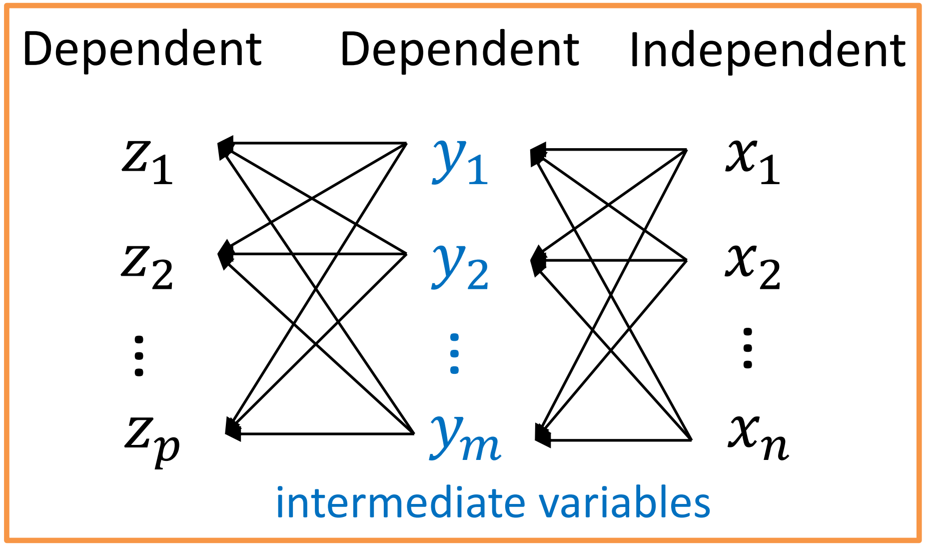
2D case

Rules in calculus

- Chain rule: the derivative of the composition function $f(g(x))$ is
$$[f(g(x))]' = f'(g(x))g'(x)$$
or in Leibniz's notation
$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$
- Product rule: the derivative of product of two functions
$$(f \cdot g)' = f' \cdot g + f \cdot g'$$
or in Leibniz's notation
$$\frac{d}{dx}(u \cdot v) = \frac{du}{dx} \cdot v + u \cdot \frac{dv}{dx}$$
- Quotient rule
$$\frac{d}{dx}\left(\frac{f(x)}{g(x)}\right) = \frac{f'g - fg'}{g^2}$$

Derivative of two-step composition

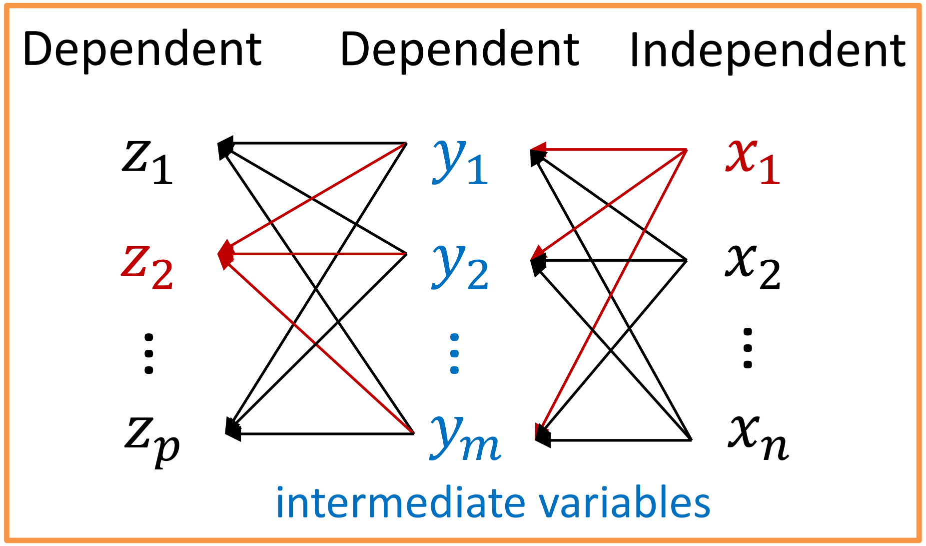
- Independent variables $x_1, x_2, \dots, x_n$
- Each y_i is a function of $x_1, x_2, \dots, x_n$
- Each z_i is a function of $y_1, y_2, \dots, y_m$



What's partial derivative of z_i w.r.t. x_j?

Derivative of two-step composition

- Independent variables $x_1, x_2, \dots, x_n$
- Each y_i is a function of $x_1, x_2, \dots, x_n$
- Each z_i is a function of $y_1, y_2, \dots, y_m$



What's partial derivative of z_i w.r.t. x_j?

$$\frac{\partial z_2}{\partial x_1} = \frac{\partial z_2}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \frac{\partial z_2}{\partial y_2}\frac{\partial y_2}{\partial x_1} + \cdots$$

$$\frac{\partial z_i}{\partial x_j} = \sum_{k=1}^{m} \frac{\partial z_i}{\partial y_k}\frac{\partial y_k}{\partial x_j}$$

Sum over the intermediate variables

for any $i \in \{1,2,\dots,p\}$ and $j \in \{1,2,\dots,n\}$

Outline

I. Math basics
II. Machine learning basics
III. 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

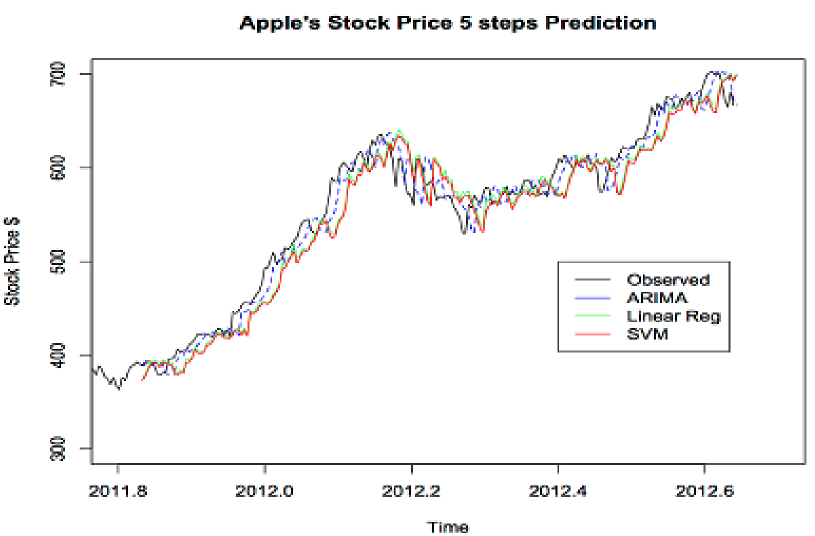
- Machine learning (ML) tasks are usually described in terms of how the ML system should process an example
- An example is a collection of features that have been quantitatively measured from some object or event
  - Features of a bucket: color, diameter, height, material, etc
  - Features of an animal: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






The tasks T

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



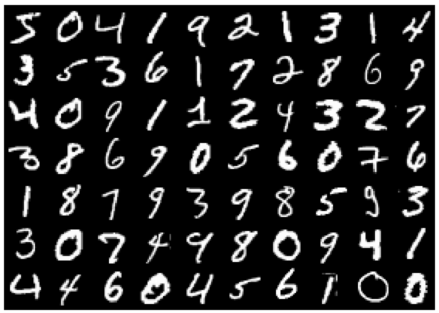
- Regression
  - Find a function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$, and m is often 1

Regression results might be converted to classification results

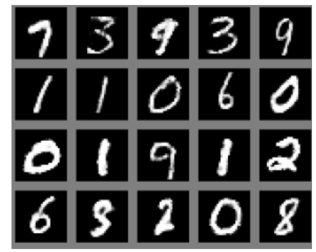
The tasks T

- 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

- A performance measure is required 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: accuracy or 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

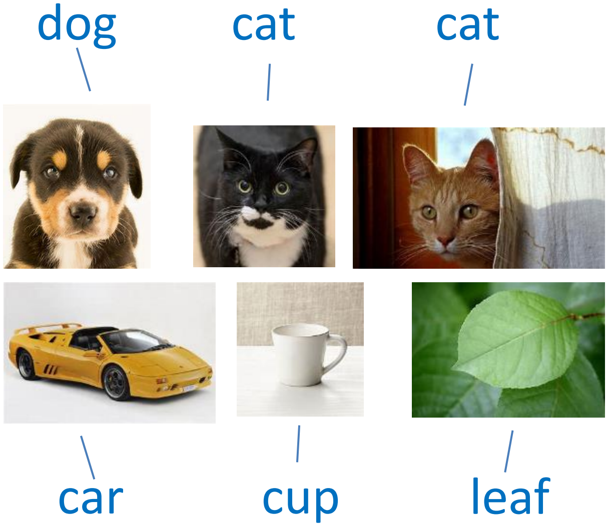
The experience, E

- 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 $p(\mathbf{x})$

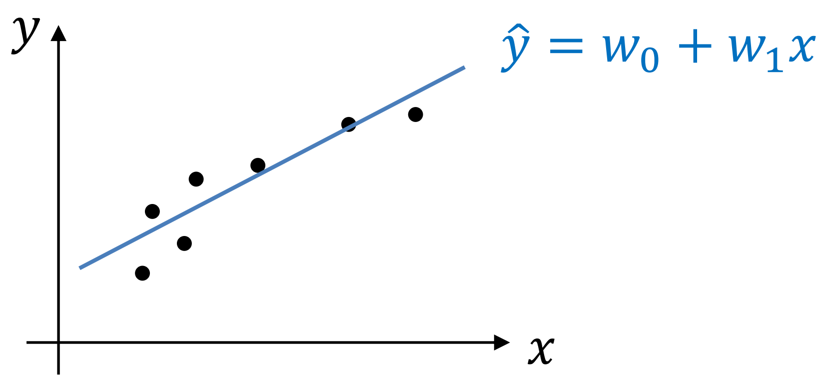
- Supervised learning algorithms

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



Example: linear regression

1D case



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 $\{(\boldsymbol{x}_i, y_i)\}^{\text{test}}$

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

Example: linear regression

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

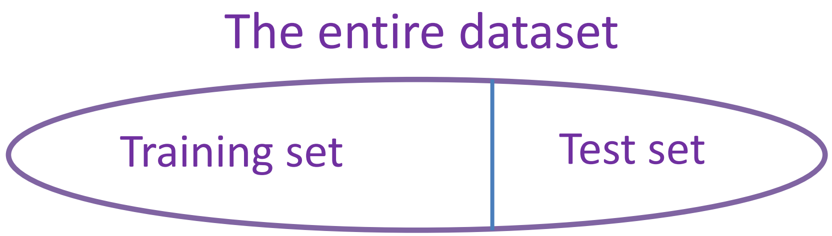
$$\text{MSE}_{\text{train}} = \frac{1}{q}\sum_i (\hat{y}_i - y)^{\text{train}}$$

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

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

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

Capacity, overfitting and underfitting



Large training error → low model capacity

Small training error → high model capacity

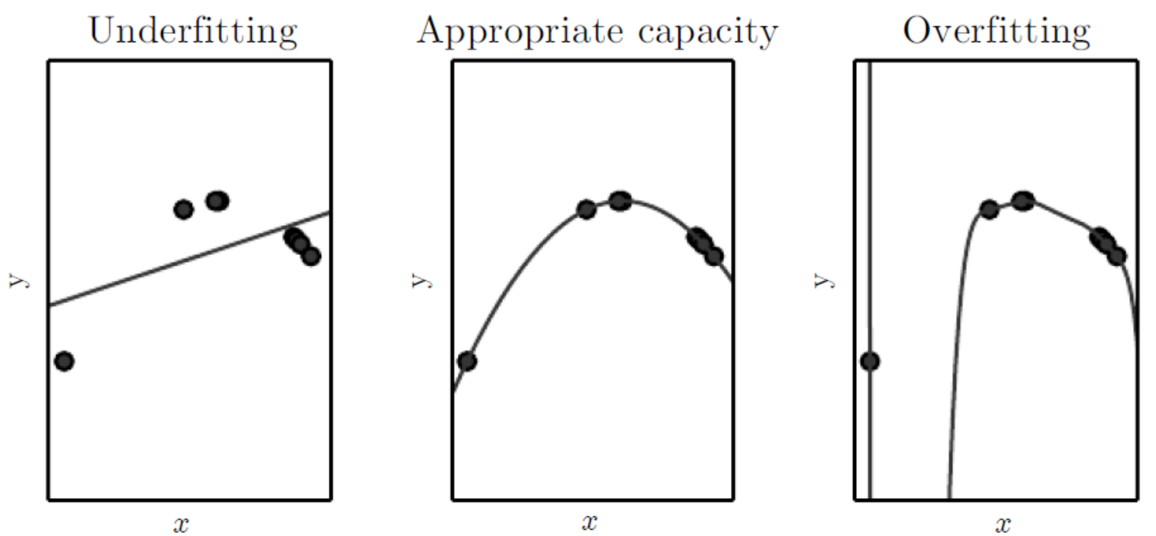
- What we want:
  - Small training error & small test error
  - If the training error is too large, the model is underfitting the training set
  - If the training error is very small but the test error is very large, the model is overfitting the training set
- A ML algorithm must perform well on new, previously unseen inputs
  - This ability is called generalization

Example: 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:

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



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

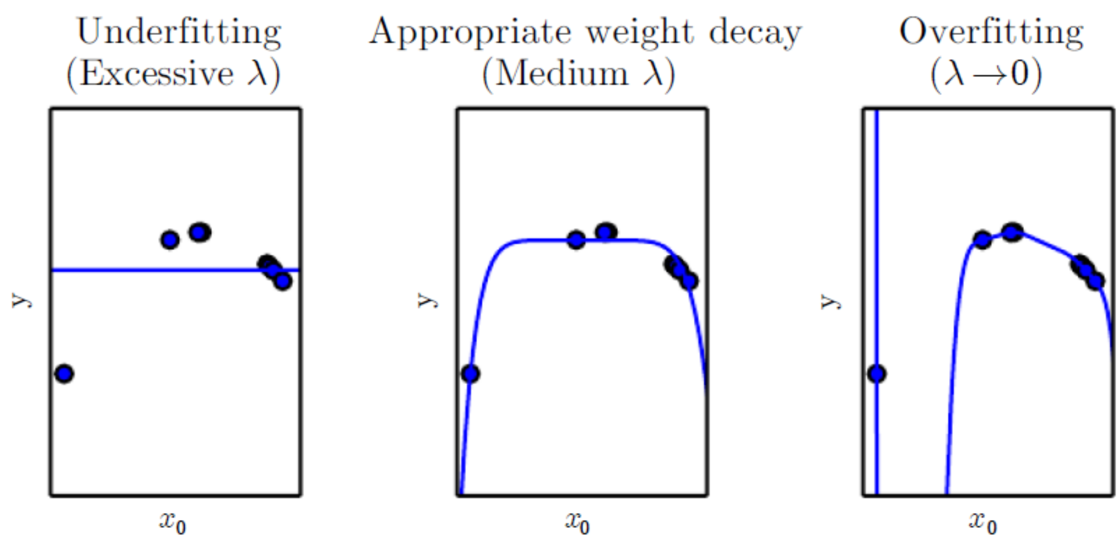
Regularization

- 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

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

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

- Many machine learning algorithms 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: $\boldsymbol{w}$

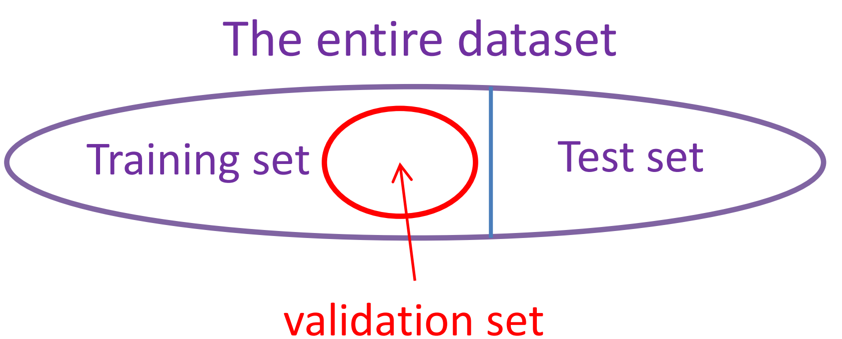
Question

- What are hyperparameters of a neural network?

- What are learnable parameters of a neural network?

Validation sets

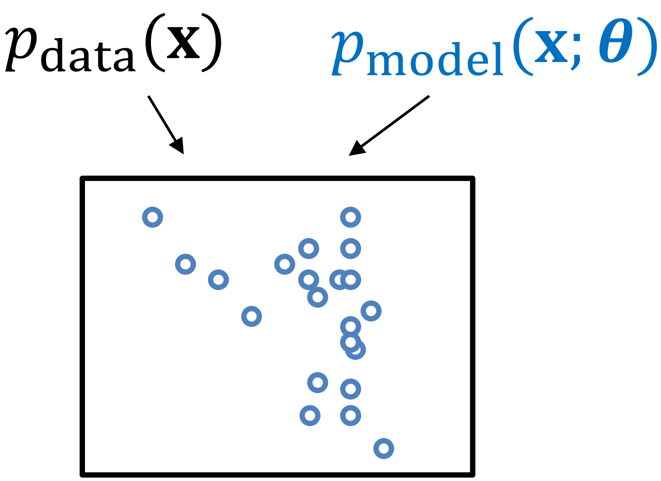
- How to choose the hyperparameters considering that we cannot see the test set?
  - Set them such that the training error is as small as possible?
- We need another set on which the model is not trained on
  - Make the error on this set as small as possible
  - This is called the validation set
- How 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 $p_{\text{data}}(\mathbf{x})$
- Task: 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 $\boldsymbol{\theta}$

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 $p_{\text{data}}(\mathbf{x})$
- Task: 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
- Decompose the training set into minibatches and optimize the cost function $L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta})$ defined over individual minibatches $\left(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}\right)$

> - $J(\boldsymbol{\theta}) = \sum_{i=1}^{N'} L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta})$
> - The batchsize ranges from 1 to a few hundreds

- At every iteration, update θ as follows

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

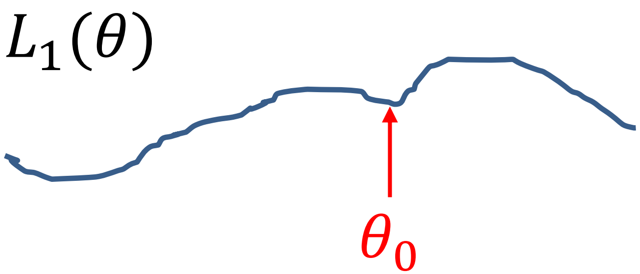
$$\boldsymbol{g}' = \nabla_{\boldsymbol{\theta}} L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta})$$

Advantages of SGD

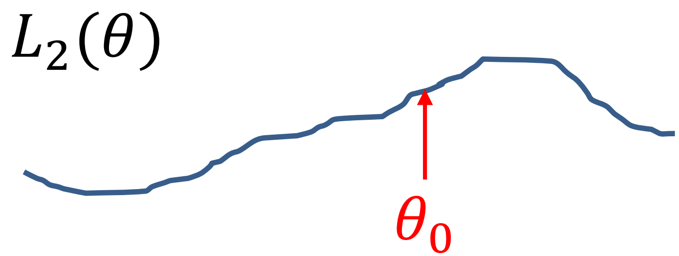
① Avoid large memory requirement when dealing with large training data

② Stochasticity is beneficial for escaping from "traps"

Note that $L(\boldsymbol{X}^{(i)}, \boldsymbol{y}^{(i)}; \boldsymbol{\theta})$ are different from different minibatches, so are their gradients



Local minimum



Not local minimum

Outline

I. Math basics
II. Machine learning basics
III. Summary

Summary of this lecture

I. Math basics

- PMF
- PDF
- Joint Prob
- Marginal prob
- Conditional prob
- Gradient decent

II. Machine learning basics

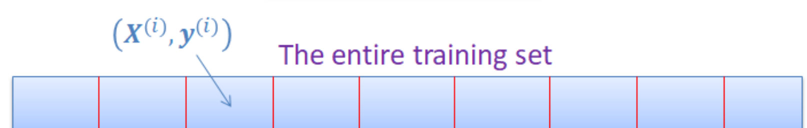
- Task T
- Performance P
- Experience E

Model capacity

MLE

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

SGD



Recommended reading

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