Techniques for Dealing with Large Datasets

Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

Machine Learning and Data

"It's not who has the best algorithm that wins. It's who has the most data"

— Andrew Ng

Example

Classify between confusable words *e.g.* {to, two, too} and {then, than}.

- For breakfast I ate … eggs.

Banko, M., & Brill, E. (2001, July). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 26-33). Association for Computational Linguistics.

Learning with Large Datasets

Suppose $m = 100{,}000{,}000$

(This is pretty realistic for many datasets. For example, in the domain of U.S. population, we can get around 300 million people in the U.S.)

$$\theta_j := \theta_j - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

Summing over 100 million entries in order to compute one step of gradient descent is too expensive !

However, ones may ask "why not just use 1,000 examples".
Perhaps, we may randomly pick 1,000 from 100 millions and that's enough !

Question

Suppose you are facing a supervised learning problem and have a very large dataset (m = 100,000,000). How can you tell if using all of the data is likely to perform much better than using a small subset of the data (*e.g.* m = 1,000)?

(i) There is no need to verify this; using a larger dataset always give much better performance.

(ii) Plot $J_{\textbf{train}}(\theta)$ as a function of the number of iterations of the optimization algorithm (such as gradient descent).

(iii) Plot a learning curve (*i.e.* $J_{\textbf{train}}(\theta)$ and $J_{\textbf{CV}}(\theta)$, plotted as a function of m) for some range of values of m (say up to m = 1,000) and verify that algorithm has bias when m is small.

(iv) Plot a learning curve for a range of values of m and verify that the algorithm has high variance when m is small.

Learning with Large Datasets

High Variance

Adding extra training examples would improve the performance.

High Bias

Increasing m to a 100 million will not do much better.

Stochastic Gradient Descent

Linear Regression with Gradient Descent (Recap)

Hypothesis Function:

$$h_\theta(x) = \Sigma_{j=0}^{n} \theta_j x_j$$

Cost Function:

$$J_{\textbf{train}}(\theta) = \frac{1}{2m} \Sigma_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient Update Rule:

Repeat until converged {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

(for every $j = 0, \ldots, n$)

}

Linear Regression with Gradient Descent (Recap)

Hypothesis Function:

$$h_\theta(x) = \Sigma_{j=0}^{n} \theta_j x_j$$

Cost Function:

$$J_{\textbf{train}}(\theta) = \frac{1}{2m} \Sigma_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient Update Rule:

Repeat until converged {

$$\theta_j := \theta_j - \boxed{\alpha \frac{1}{m} \Sigma_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}}$$

(for every $j = 0, \ldots, n$)

}

Batch GD *vs.* Stochastic GD

Batch Gradient Descent

$$J_{\textbf{train}}(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Repeat until converged {

$$\theta_j := \theta_j - \alpha \boxed{\frac{1}{m}\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}}$$

(for every $j = 0,\ldots,n$)

}

Stochastic Gradient Descent

$$\textbf{cost}(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$$J_{\textbf{train}}(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}\textbf{cost}(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset.
2. Repeat {

    for i = 1, …, m {

    $$\theta_j := \theta_j - \alpha \boxed{\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}}$$

    (for every $j = 0,\ldots,n$)

    }

}

$$\frac{\partial}{\partial\theta_j}\textbf{cost}(\theta, (x^{(i)}, y^{(i)}))$$

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

Batch GD Update / Training Step

In SGD, each training step is faster than Batch GD because we don't need to sum up over all the training examples !

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

But, for each training step, it is trying to fit just one training example !

Batch GD Update / Training Step

SGD Update / Training Step

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

But, for each training step, it is trying to fit just one training example !

Batch GD Update / Training Step

SGD Update / Training Step

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

But, for each training step, it is trying to fit just one training example !

Batch GD Update / Training Step

SGD Update / Training Step

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

Batch GD Update / Training Step

SGD Update / Training Step

In SGD, it will generally move the parameters toward the global minimum (but not always) !

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

In SGD, it seems like a random-walking, circuitous path toward the global minimum !

→ Batch GD Update / Training Step

→ SGD Update / Training Step

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

Batch GD Update / Training Step

SGD Update / Training Step

In SGD, it doesn't actually converge in the same sense as Batch GD does *i.e.* it doesn't head to the global minimum and stay there !

Stochastic GD (Recap)

1. Randomly shuffle dataset.
2. Repeat {

    for $i = 1, \ldots, m$ {

    $$\theta_j := \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

    (for every $j = 0, \ldots, n$)

    }

}

In practice, how many times do we want to repeat the outer loop ?

Depending on the training set size, if m is large, then looping one time may be enough *e.g.* m = 100,000,000

Question

Which of the following statements about SGD are true?
Circle all that apply.

(i) When the training set size is very large, SGD can be much faster than Batch GD.

(ii) The cost function $J_{\textbf{train}}(\theta) = (1/2m) \cdot \Sigma_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$ should go down with every training step of batch GD (assuming a well-tuned learning rate α) but not necessarily with SGD.

(iii) SGD is applicable only to linear regression but not to other models (*e.g.* logistic regression or neural networks).

(iv) Before beginning the outer loop of SGD, it is a good idea to 'shuffle' your training data into a random order.

Mini-batch Gradient Descent

Mini-batch GD

- Batch GD: use all m examples in each training step
- Stochastic GD: use 1 example in each training step
- Mini-batch GD: use b examples in each training step, where $1 \leq b < m$

Example: Let $b = 10$ and $m = 1{,}000$

Repeat {

for $i = 1, 11, 21, 31, \ldots, 991$ {

$$\theta_j := \theta_j - \alpha \frac{1}{10} \Sigma_{k=i}^{i+9} \left(h_\theta(x^{(k)}) - y^{(k)} \right) x_j^{(k)}$$

(for every $j = 0, \ldots, n$)

}

}

Compared to Batch GD, Mini-batch GD runs faster !

Question

Suppose you use mini-batch gradient descent on a training set of size m, and you use a mini-batch size of b. The algorithm becomes the same as batch gradient descent if:

(i) $b = 1$

(ii) $b = m / 2$

(iii) $b = m$

(iv) None of the above

Stochastic Gradient Descent Convergence

Checking for Convergence

Batch Gradient Descent

- Plot $J_{\textbf{train}}(\theta)$ as a function of the number of iterations of gradient descent and make sure that $J_{\textbf{train}}(\theta)$ is decreasing on every iteration

$$J_{\textbf{train}}(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Stochastic Gradient Descent

- During learning, compute $\textbf{cost}(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$

$$\textbf{cost}(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

- Every 1,000* iterations, plot $\textbf{cost}(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1,000 examples processed by the algorithm.

(This gives us an estimate of how well the algorithm is doing !)

* This is just an example.
We may change this number.

Examples of SGD's Plots

Plot **cost**$(\theta, (x^{(i)}, y^{(i)}))$, averaged over the last 1,000 examples

Improvement in SGD

Learning rate α is typically constant ➡

To make it converge, try slowly decrease α over time *e.g.*

$$\alpha = \frac{\textbf{constant1}}{\textbf{iterationNumber} + \textbf{constant2}}$$

However, people tend to not doing this because we need to play with extra constants, which makes the algorithm more finicky.

Improvement in SGD

Learning rate α is typically constant

To make it converge, try slowly decrease α over time *e.g.*

$$\alpha = \frac{\textbf{constant1}}{\textbf{iterationNumber} + \textbf{constant2}}$$

However, people tend to not doing this because we need to play with extra constants, which makes the algorithm more finicky.

Question

Which of the following statements about stochastic gradient descent are true? Circle all that apply.

(i) Picking a learning rate α that is very small has no disadvantage and can only speed up learning.

(ii) If we reduce the learning rate α (and run stochastic gardens descent long enough), it's possible that we may find a set of better parameters than with larger α.

(iii) If we want stochastic gradient descent to converge to a (local) minimum rather than wander of 'oscillate' around it, we should slowly increase α over time.

(iv) If we plot $\textbf{cost}(\theta, (x^{(i)}, y^{(i)}))$ (averaged over the last 1,000 examples) and SGD does not seem to be reducing the cost, one possible problem may be that the learning rate α is poorly tuned.

Getting lots of data: Artificial Data Synthesis

Character Recognition

Let's use character recognition as our running example

Artificial Data Synthesis for Photo OCR

Real Data

All of these examples are raw images.

How can we come up with a much larger training set?

Artificial Data Synthesis for Photo OCR

Real Data

Modern computers have font libraries *e.g.* from word processing softwares, websites, *etc.*

Artificial Data Synthesis for Photo OCR

Real Data

Synthetic Data

One way to synthesize artificial data is to:
(1) copy a font, (2) paste it on a random background

Synthesizing Data by Introducing Distortions

Apply artificial distortion

We need to think thoroughly and figure out what distortions are reasonable to apply on a raw image!

Synthesizing Data by Introducing Distortions: Speech Recognition

- Original audio
- Audio on bad cellphone connection
- Noisy background: Crowd
- Noisy background: Machinery

With this approach, we can amplify from 1 example to 4 examples

Synthesizing Data by Introducing Distortions

- Distortion introduced should be representation of the type of noise/distortions in the test set.

Audio:

Background noise,
bad cellphone connection

- Usually does not help to add purely random/meaningless noise to your data.

$x_i := x_i +$ **random noise**

where x_i intensity (brightness) of pixel i

Question

Suppose you are training a linear regression model with m examples by minimizing:

$$J(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Suppose you duplicate every example by making two identical copies of it. That is, where you previously had one example $\left(x^{(i)}, y^{(i)}\right)$, you now have two copies of it, so you now have $2m$ examples. Is this likely to help?

(i) Yes, because increasing the training set size will reduce variance.

(ii) Yes, so long as you are using a large number of features (a low 'bias' learning algorithm).

(iii) No, you may end up with different parameters θ, but they are unlikely to do any better than the one learned from the original training set.

(iv) No, and in fact you will end up with the same parameters θ as before you duplicated the data.

Discussion on Getting More Data

1. Make sure we have a low bias classifier before expending the effort *i.e.* plot learning curves. If it is currently not a low bias classifier, then we may keep increasing the number of features / number of hidden units in neural network until you have a low bias classifier.
2. Now, we can put effort on creating a large number of artificial dataset.
3. Ask ourselves a question "how much work would it be to get 10x as much data as we currently have?".
   - Artificial data synthesis
   - Collect/label it yourself
   - 'Crowd source' *e.g.* Amazon Mechanical Turk

#hours?

m = 1,000
10 secs / example
m = 10,000

Question

You've just joined a product group that has been developing a machine learning application for the last 12 months using 1,000 training examples. Suppose that by manually collecting and labeling examples, it takes you an average of 10 seconds to obtain one extra training example. Suppose you work 8 hours a day. How many days will it take you to get 10,000 examples ? Pick the closet answer.

(i) About 1 day

(ii) About 3.5 days

(iii) About 28 days

(iv) About 200 days

Ceiling Analysis: What part of the pipeline to work on next

Estimating the errors due to each component (Ceiling Analysis)

Let's use Photo OCR as our running example

What part of the pipeline should you spend the most time trying to improve?

To make decisions, we use an evaluation metric.

Component	Character Accuracy
Overall system	72%

Estimating the errors due to each component (Ceiling Analysis)

Let's use Photo OCR as our running example

What part of the pipeline should you spend the most time trying to improve?

To make decisions, we use an evaluation metric.

Component	Character Accuracy
Overall system	72%
Text detection	89%

Estimating the errors due to each component (Ceiling Analysis)

Let's use Photo OCR as our running example

What part of the pipeline should you spend the most time trying to improve?

To make decisions, we use an evaluation metric.

Component	Character Accuracy
Overall system	72%
Text detection	89%
Character segmentation	90%

Estimating the errors due to each component (Ceiling Analysis)

Let's use Photo OCR as our running example

What part of the pipeline should you spend the most time trying to improve?

To make decisions, we use an evaluation metric.

	Component	Character Accuracy
17%	Overall system	72%
	Text detection	89%
1%	Character segmentation	90%
10%	Character recognition	100%

Estimating the errors due to each component (Ceiling Analysis)

Let's use Photo OCR as our running example

What part of the pipeline should you spend the most time trying to improve?

To make decisions, we use an evaluation metric.

How much could you possibly gain if one of these components became absolutely perfect?

	Component	Character Accuracy
17%	Overall system	72%
	Text detection	89%
1%	Character segmentation	90%
10%	Character recognition	100%

Another Ceiling Analysis Example

Face recognition from images (artificial example)

Here is a possible pipeline:

Camera Image

From Advice for applying Machine Learning, Andrew Ng, Stanford University: http://cs229.stanford.edu/materials/ML-advice.pdf

Another Ceiling Analysis Example

Face recognition from images (artificial example)

Here is a possible pipeline:

Camera Image → Preprocess (*i.e.* remove background)

From Advice for applying Machine Learning, Andrew Ng, Stanford University: http://cs229.stanford.edu/materials/ML-advice.pdf

Another Ceiling Analysis Example

Face recognition from images (artificial example)

Here is a possible pipeline:

Run a sliding windows classifier to draw a box around a person's face

From Advice for applying Machine Learning, Andrew Ng, Stanford University: http://cs229.stanford.edu/materials/ML-advice.pdf

Another Ceiling Analysis Example

Face recognition from images (artificial example)

Here is a possible pipeline:

From Advice for applying Machine Learning, Andrew Ng, Stanford University: http://cs229.stanford.edu/materials/ML-advice.pdf

Another Ceiling Analysis Example

Face recognition from images (artificial example)

Here is a possible pipeline:

From Advice for applying Machine Learning, Andrew Ng, Stanford University: http://cs229.stanford.edu/materials/ML-advice.pdf

Another Ceiling Analysis Example

Component	**Character Accuracy**
Overall system	85%

Another Ceiling Analysis Example

Component	Character Accuracy
Overall system	85%
Preprocess (*i.e.* remove background)	85.1%

Another Ceiling Analysis Example

Component	Character Accuracy
Overall system	85%
Preprocess (*i.e.* remove background)	85.1%
Face detection	91%

Component	Character Accuracy
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%

Another Ceiling Analysis Example

Component	Character Accuracy
Overall system	85%
Preprocess (*i.e.* remove background)	85.1%
Face detection	91%

0.1%

5.9%

4%

Component	Character Accuracy
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%
Logistic regression	100%

1%

1%

3%

Another Ceiling Analysis Example

Component	Character Accuracy
Overall system	85%
Preprocess (*i.e.* remove background)	85.1%
Face detection	91%

0.1%

5.9%

4%

Component	Character Accuracy
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%
Logistic regression	100%

1%

1%

3%

Question

Suppose you perform ceiling analysis on a pipelined machine learning system, and when we plug in the ground-truth labels for one of the components, the performance of the overall system improves very little. This probably means: (circle all that apply)

(i) We should dedicate significant effort to collecting more data for that component.

(ii) It is probably not worth dedicating engineering resources to improving that component of the system.

(iii) If that component is a classifier training using gradient descent, it is probably not worth running gradient descent for 10x as long to see if it converges to better classier parameters.

(iv) Choosing more features for that component may help (*i.e.* reducing bias), and reducing the number of features for that component (*i.e.* reducing variance) is unlikely to do so.

Summary

- Not trust your own gut feeling about what components to work on.
- Do a ceiling analysis, often there is a much better and much more reliable way for deciding where to put a focused effort in order to really improve the performance of some component.

We have studied ML. So, we should know some of their critiques !

Adversarial Examples

- Change often indistinguishable to human eyes.
- Adversarial examples generalize across architectures, training sets.
- Adversarial perturbations η generalize across examples.
- Can construct in the physical world.
- Let's see some of the adversarial examples !
- Still an open problem. How fundamental ?

Adversarial Example 1

Can be reproduced from:

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus.

Intriguing properties of neural networks, 2013.

Adversarial examples generated for AlexNet

Left: correctly classified image
Center: perturbation
Right: classified as Ostrich

Adversarial Example 2

Fast Gradient Sign method

$\boldsymbol{x}$	$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$	$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"panda"	"nematode"	"gibbon"
57.7% confidence	8.2% confidence	99.3 % confidence

Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, ICLR 2015

Adversarial Example 3

An adversarial object from the paper
"Synthesizing Robust Adversarial Examples."

The turtle is classified as a "rifle" at every viewpoint as a rifle by the InceptionV3 image classifier.

Anish Athalye, Logan Engstrom, Andrew Ilyas, Kevin Kwok. Synthesizing Robust Adversarial Examples, ICML 2018

Interpretability

Why do we need interpretability?

- Trust
- Causality
- Transferability
- Informativeness
- Fairness and ethics

- Can we trust AI if we don't know how it works?: https://goo.gl/at8Lxs
- Reproducibility crisis: https://goo.gl/UuXSwn

Interpretability: Fallacy

All AI applications need to be transparent !

Is this a transparent algorithm?

If not, why do you use it?

From http://cs229.stanford.edu/materials/critiques-ml.pdf

IBM Project Debater

Project Debater

It is the 1st AI system that can debate humans on complex topics. The goal is to help people build persuasive arguments and make well-informed decisions.

Project Debater relies on three pioneering capabilities:

1. data-driven speech writing and delivery,
2. listening comprehension, and
3. the modeling of human dilemmas.

4 Basic Steps:

1. Understanding a topic,
2. Argument construction,
3. Content organization,
4. Constructing an argument and rebuttal.

Debated on February 11, 2019 (https://en.wikipedia.org/wiki/Project_Debater)

Now, you should consider yourself as an expert in machine learning (and AI) !

Thank you very much !

Acknowledgement

- Bishop, C. (2006), *Pattern Recognition and Machine Learning*, Springer, Chapters 3, 4, 6, 7.
- Hastie, T., Timshirari, R., and Friedman, J. (2016), *Elements of Statistical Learning: Data Mining, Inference, and Prediction,* Springer, Chapters 2, 3, 4, 12.
- Bonnin, R. (2016), *Building Machine Learning Projects with TensorFlow*, Packt Publishing.
- Ng, A. (2017), *Machine Learning,* Stanford University*.
- Dailey, M. (2018), *Machine Learning*, Asian Institute of Technology

These notes contains materials from Bishop (2016), Hastie *et al.* (2016), Ng (2017), and Dailey (2018)

* Adapted from Coursera and CS229 (Stanford University)

Biography

- **Name:** Teeradaj Racharak (X)
- **Affiliation:** Japan Advanced Institute of Science and Technology (Kanazawa city, Japan)
- **Academic Position:** Assistant Professor of Artificial Intelligence (AI), School of Information Science
- **Area:** Intelligent Robotics
- **Research Interests:** Formal development of *human intelligence* for AI, in particular:
  - (Interpretable) Machine learning
  - Computational logic
  - Explainable AI (XAI)
- **Contact:** r.teeradaj@gmai.com

(Me in Nara, 2018)

Artificial Intelligence (AI)

Designing and building machines that behaves intelligently

behaves intelligently $\approx$ human-like

learn and reason

Can a machine ~~behave~~ like a human?

Agent Reasoning

Argument Mining

Cathy V. Hathorn

★★★★★ **Best Book**

September 12, 2018

Format: Kindle Edition | Verified Purchase

Loved this novel. Think this book is Deanna best work, loved it. Great writer will be looking for more of her books

noonie

★★★☆☆ **Entertaining**

November 27, 2018

Format: Kindle Edition

I felt the book at times went on and on about detail that wasn't interesting..bottom line it was ok just wordy..

(reviews made on amazon.com)

Integration between Learning and Reasoning

Gao, Yang, and Francesca Toni, Argumentation Accelerated Reinforcement Learning for Cooperative Multi-Agent Systems, *ECAI,* 2014

Study Information

Japan Advanced Institute of Science and Technology

9 Areas:

(1) Human Life Design
(2) Knowledge Management
(3) Security and Networks
(4) Intelligent Robotics
(5) Entertainment Technology
(6) Energy and Environment
(7) Materials Chemistry
(8) Applied Physics
(9) Bioscience and Biotechnology

Komatsu Airport offers regular international flights to and from Shanghai (China), Seoul (Korea) and Taipei (Taiwan).

JAIST is the 1st independent national graduate university without undergraduate division.

Japan Advanced Institute of Science and Technology

Entrance Fee/Tuition Fee for 2017

Divison	Screening Fee	Entrance Fee	Tuition fee
Master's Program	30,000 yen	282,000 yen	267,900 yen（semester） 535,800 yen（year）
Doctoral Program			

Master (2 Yrs) = 30,000 + 282,000 + (535,800 * 2) = 1,383,600 JPY

≈ 400,000 THB

PhD (3 Yrs) = 30,000 + 282,000 + (535,800 * 3) = 1,919,400 JPY

≈ 550,000 THB

From https://www.jaist.ac.jp/english/studentlife/support/fee.html

Some Available Scholarships

- JAIST Scholarship: https://www.jaist.ac.jp/english/studentlife/support/scholarships.html
- JSPS scholarship: http://www.jsps.go.jp/english/index.html
- TAIST Tokyo Tech: http://www2.siit.tu.ac.th/ictprojects/taist/#/home
- JAIST-NECTEC-SIIT: http://www2.siit.tu.ac.th/ictprojects/siitjaist/
- AIT scholarship: https://www.ait.ac.th/admissions/scholarships/
- ทุน ก.พ.: https://www.ocsc.go.th/scholarship/degree