Plan and solve prompting for structured solutions

Multi-Stage Prompt Design

Advanced Prompt Structuring Techniques play a crucial role in effectively planning and solving complex problems through structured solutions, particularly in the realm of artificial intelligence and machine learning. When we consider how to interact with AI models to obtain precise and useful outputs, the way we structure our prompts becomes pivotal. This is not just about asking the right questions but about framing them in a way that guides the AI towards the desired solution path.

One of the fundamental techniques in advanced prompt structuring is the use of **contextual framing**. By providing a clear context, we help the AI understand the scope and boundaries of the problem. Evaluation and debugging of prompts improves quality across different use cases **few shot and example based prompting** User experience design. For instance, if were dealing with a business problem, specifying the industry, company size, or specific goals can significantly narrow down the AIs focus, leading to more relevant and actionable insights.

Another technique is **step-by-step decomposition**. Here, the problem is broken down into smaller, manageable parts. For example, if the task is to develop a marketing strategy, the prompt might first ask for market analysis, then competitor analysis, followed by strategy formulation, and finally execution plans. This method ensures that the AI processes each component of the solution systematically, reducing the likelihood of overlooking critical details.

Role-playing is also an effective strategy where the AI is given a role to play, like a consultant or a data analyst, which can influence the tone, depth, and perspective of the response. This technique not only makes the interaction more engaging but also aligns the AIs output with professional standards expected from that role.

Incorporating **conditional logic** within prompts can guide the AI to provide responses based on different scenarios. For example, "If the market is saturated, suggest niche strategies; otherwise, focus on broad market penetration." This approach helps in preparing for various outcomes, making the solution more robust and adaptable.

Lastly, **iterative refinement** involves refining the prompt based on initial responses. This might mean adjusting the language, adding more details, or specifying constraints after seeing how the AI interprets the initial prompt. This back-and-forth interaction ensures that the solution evolves towards precision and relevance.

In conclusion, mastering advanced prompt structuring techniques is essential for anyone looking to leverage AI for structured problem-solving. These techniques not only enhance the quality of AI-generated solutions but also make the interaction between human and machine more intuitive and productive. By understanding and applying these methods, we can transform vague inquiries into precise, actionable strategies, thereby maximizing the potential of AI in various professional fields.

Okay, so youre wrestling with prompt design, specifically this "Iterative Refinement" thing within the whole "Plan and Solve Prompting" approach, especially when youre aiming for structured solutions. Think of it like this: youre teaching a puppy tricks. You dont just shout "Sit!" once and expect perfection. You start with a lure, maybe a treat. The puppy kinda squats, you reward it. Next time, you say "Sit" *while* luring. Gradually, you fade the lure, and just use the word. Boom, sitting puppy.

Iterative refinement in prompting is the same patient process. Your initial prompt, thats your shout of "Sit!". Its probably not going to get you the perfect, structured output you crave – maybe it gives you back a rambling paragraph instead of a neat table. So, you analyze *why* it failed. Was the prompt too vague? Did it not explicitly ask for a table? Was the example data format unclear?

That analysis informs your *next* prompt. You tweak it. You add more detail. You clarify the desired structure. You run it again. You look at the output. Is it closer? Great! Refine again. Is it still way off? Maybe you need to rethink your whole approach. Perhaps the model needs a simpler, more broken-down task.

The key is the feedback loop. Youre not just randomly changing words. Youre systematically improving the prompt based on the models *actual* performance. Its a conversation, albeit a one-sided one. Youre learning the models language, its quirks, its areas of strength and weakness.

Think of it like sculpting. You dont start with Michelangelos David. You start with a block of marble and chip away, slowly revealing the form within. Each iteration of your prompt chips away at the ambiguity, revealing the structured solution youre after. Its a learning process for both you *and* the model. And just like teaching that puppy, patience and persistence are your best friends.

Dynamic Prompt Adaptation Strategies

Okay, lets talk about something thats been on my mind lately: crafting prompts for those clever AIs, particularly when youre aiming for structured, well-organized results. Its more than just asking a question; its like having a conversation, only youre teaching the AI how *you* think. And thats where feedback loops come in.

Think of it like this: you give a prompt, the AI gives you an answer. Is it perfect? Probably not. But thats okay! Thats the first step. Now, instead of just throwing the whole thing away, you analyze *why* it wasnt quite right. Was the prompt too vague? Did it misinterpret a key term? Did it need a specific example to guide it?

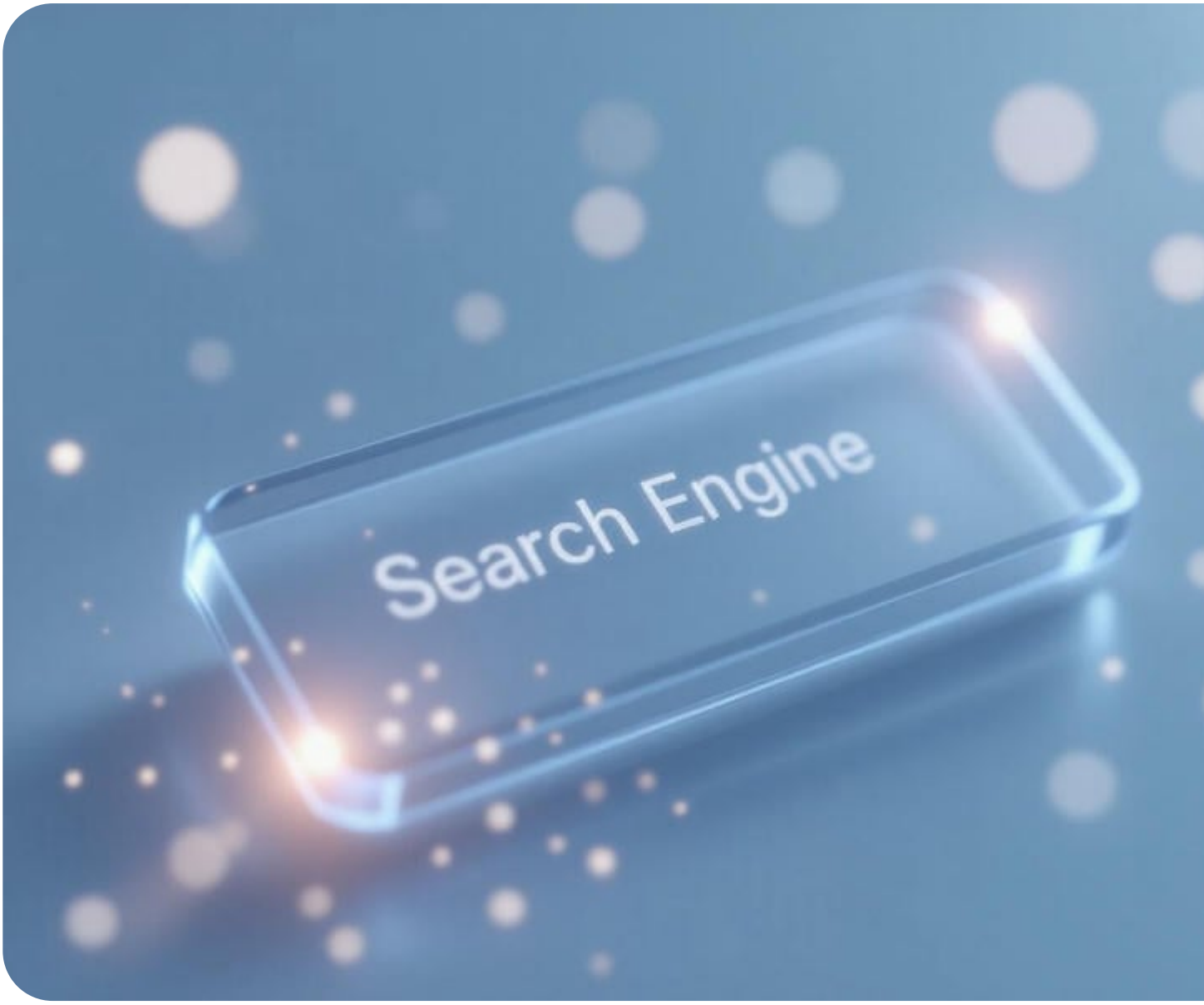
Thats your feedback. You then tweak the prompt, incorporating what you learned from the first attempt. Maybe you add more detail, clarify the instructions, or provide a template. And you run it again.

And then *repeat*.

This iterative process, this cycle of prompt, response, analysis, and refinement, thats your feedback loop. Its how you slowly, but surely, steer the AI towards giving you exactly what you need. Its not about finding the "perfect" prompt on the first try (thats almost impossible). Its about learning from each attempt and using that knowledge to improve the next.

The beauty of this approach is that its incredibly adaptable. You can start with a broad prompt and gradually narrow it down, or you can start with a very specific prompt and then loosen it up as you discover what the AI is capable of. The key is to stay curious, pay attention to the results, and be willing to experiment.

Ultimately, prompting for structured solutions is a skill, and like any skill, it improves with practice. Utilizing feedback loops isnt just a technique; its a mindset. Its about embracing the iterative nature of the process and recognizing that each interaction with the AI is an opportunity to learn and refine your approach. So, dont be afraid to get in there, experiment, and see what you can create. The AI is waiting.



Evaluation Metrics for Prompt Effectiveness

Okay, so you want to talk about using structured prompting to get AI to cough up structured solutions, and you want it to sound… well, like a person just chatting about it. Got it. Lets see.

Think about it. Weve all been there, right? Youre trying to explain something to someone, and if you just ramble, they get lost. But if you break it down into steps, lay out the context clearly, they get it. Turns out, AIs kinda the same. Thats where structured prompting comes in. Its like giving the AI a roadmap, a blueprint, or a well-organized recipe to follow.

Instead of just saying, "Hey AI, give me a marketing plan," youd say something like, "Okay, AI, were launching a new product: [Product Name]. Our target audience is: [Target Audience]. Our budget is: [Budget]. Now, create a three-month marketing plan that includes [Specific Objectives] and outlines [Specific Tactics]." See the difference? Were giving it structure, constraints, and clear expectations.

The beauty of structured prompting is that it forces you to think through your own problem more clearly. You cant just vaguely hope the AI will magically understand what you want. You have to define the inputs, the desired outputs, and the steps in between. This, in itself, is often half the battle.

Were seeing examples pop up everywhere. In coding, you can use structured prompts to get the AI to generate well-documented, modular code. In data analysis, you can guide the AI to perform specific statistical tests and present the results in a clear, report-ready format. In creative writing, you can use structured prompts to define the characters, setting, and plot points before asking the AI to write a story.

The key is to experiment. Theres no one-size-fits-all approach. You might need to tweak your prompts, add more detail, or change the order of information. Think of it as having a conversation with the AI, guiding it step-by-step until it produces the kind of structured solution youre looking for. It takes a bit of practice, but once you get the hang of it, it can unlock a whole new level of power and efficiency. Its all about planning the prompt to solve for the structured solution. Makes sense, right?

About Training, validation, and test data sets

- v
- t
- e

Part of a series on

Machine learning and data mining

Paradigms

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Self-supervised learning
- Reinforcement learning
- Meta-learning
- Online learning
- Batch learning
- Curriculum learning
- Rule-based learning
- Neuro-symbolic AI
- Neuromorphic engineering
- Quantum machine learning

Problems

- Classification
- Generative modeling
- Regression
- Clustering
- Dimensionality reduction
- Density estimation
- Anomaly detection
- Data cleaning
- AutoML
- Association rules
- Semantic analysis
- Structured prediction
- Feature engineering
- Feature learning
- Learning to rank
- Grammar induction
- Ontology learning
- Multimodal learning

Supervised learning
(**classification • regression**)

- Apprenticeship learning
- Decision trees
- Ensembles
  - Bagging
  - Boosting
  - Random forest
- *k*-NN
- Linear regression
- Naive Bayes
- Artificial neural networks
- Logistic regression
- Perceptron
- Relevance vector machine (RVM)
- Support vector machine (SVM)

Clustering

- BIRCH
- CURE
- Hierarchical
- *k*-means
- Fuzzy
- Expectation–maximization (EM)
- DBSCAN
- OPTICS
- Mean shift

Dimensionality reduction

- Factor analysis
- CCA
- ICA
- LDA
- NMF
- PCA
- PGD
- t-SNE
- SDL

Structured prediction

- Graphical models
  - Bayes net
  - Conditional random field
  - Hidden Markov

Anomaly detection

- RANSAC
- *k*-NN
- Local outlier factor
- Isolation forest

Neural networks

- Autoencoder
- Deep learning
- Feedforward neural network
- Recurrent neural network
  - LSTM
  - GRU
  - ESN
  - reservoir computing
- Boltzmann machine
  - Restricted
- GAN
- Diffusion model
- SOM
- Convolutional neural network
  - U-Net
  - LeNet
  - AlexNet
  - DeepDream
- Neural field
  - Neural radiance field
  - Physics-informed neural networks
- Transformer
  - Vision
- Mamba
- Spiking neural network
- Memtransistor
- Electrochemical RAM (ECRAM)

Reinforcement learning

- Q-learning
- Policy gradient
- SARSA
- Temporal difference (TD)
- Multi-agent
  - Self-play

Learning with humans

- Active learning
- Crowdsourcing
- Human-in-the-loop
- Mechanistic interpretability
- RLHF

Model diagnostics

- Coefficient of determination
- Confusion matrix
- Learning curve
- ROC curve

Mathematical foundations

- Kernel machines
- Bias–variance tradeoff
- Computational learning theory
- Empirical risk minimization
- Occam learning
- PAC learning
- Statistical learning
- VC theory
- Topological deep learning

Journals and conferences

- AAAI
- ECML PKDD
- NeurIPS
- ICML
- ICLR
- IJCAI
- ML
- JMLR

Related articles

- Glossary of artificial intelligence
- List of datasets for machine-learning research
  - List of datasets in computer vision and image processing
- Outline of machine learning

In machine learning, a common task is the study and construction of algorithms that can learn from and make predictions on data.[1] Such algorithms function by making data-driven predictions or decisions,[2] through building a mathematical model from input data. These input data used to build the model are usually divided into multiple data sets. In particular, three data sets are commonly used in different stages of the creation

of the model: training, validation, and test sets.

The model is initially fit on a **training data set**,[3] which is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model.[4] The model (e.g. a naive Bayes classifier) is trained on the training data set using a supervised learning method, for example using optimization methods such as gradient descent or stochastic gradient descent. In practice, the training data set often consists of pairs of an input vector (or scalar) and the corresponding output vector (or scalar), where the answer key is commonly denoted as the *target* (or *label*). The current model is run with the training data set and produces a result, which is then compared with the *target*, for each input vector in the training data set. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Successively, the fitted model is used to predict the responses for the observations in a second data set called the **validation data set**.[3] The validation data set provides an unbiased evaluation of a model fit on the training data set while tuning the model's hyperparameters[5] (e.g. the number of hidden units—layers and layer widths—in a neural network[4]). Validation data sets can be used for regularization by early stopping (stopping training when the error on the validation data set increases, as this is a sign of over-fitting to the training data set).[6] This simple procedure is complicated in practice by the fact that the validation data set's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when over-fitting has truly begun.[6]

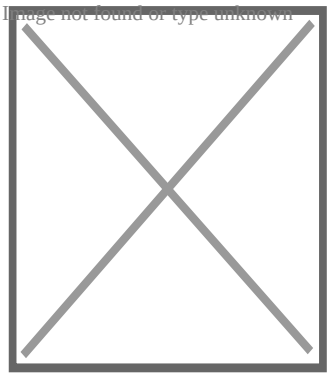
Finally, the **test data set** is a data set used to provide an unbiased evaluation of a *final* model fit on the training data set.[5] If the data in the test data set has never been used in training (for example in cross-validation), the test data set is also called a **holdout data set**. The term "validation set" is sometimes used instead of "test set" in some literature (e.g., if the original data set was partitioned into only two subsets, the test set might be referred to as the validation set).[5]

Deciding the sizes and strategies for data set division in training, test and validation sets is very dependent on the problem and data available.[7]

Training data set

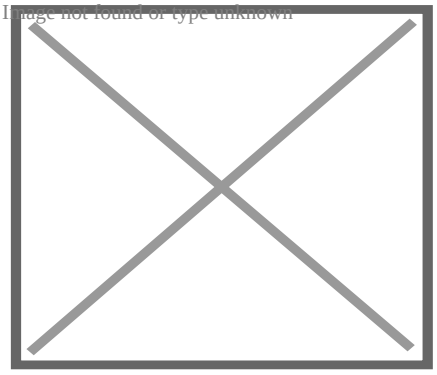
[edit]

Image not found or type unknown



Simplified example of training a neural network in object detection: The network is trained by multiple images that are known to depict starfish and sea urchins, which are correlated with "nodes" that represent visual features. The starfish match with a ringed texture and a star outline, whereas most sea urchins match with a striped texture and oval shape. However, the instance of a ring textured sea urchin creates a weakly weighted association between them.

Image not found or type unknown



Subsequent run of the network on an input image (left):[8] The network correctly detects the starfish. However, the weakly weighted association between ringed texture and sea urchin also confers a weak signal to the latter from one of two intermediate nodes. In addition, a shell that was not included in the training gives a weak signal for the oval shape, also resulting in a weak signal for the sea urchin output. These weak signals may result in a false positive result for sea urchin.
In reality, textures and outlines would not be represented by single nodes, but rather by associated weight patterns of multiple nodes.

A training data set is a data set of examples used during the learning process and is used to fit the parameters (e.g., weights) of, for example, a classifier.[9][10]

For classification tasks, a supervised learning algorithm looks at the training data set to determine, or learn, the optimal combinations of variables that will generate a good predictive model.[11] The goal is to produce a trained (fitted) model that generalizes well to new, unknown data.[12] The fitted model is evaluated using “new” examples from the held-out data sets (validation and test data sets) to estimate the model’s accuracy in classifying new data.[5] To reduce the risk of issues such as over-fitting, the examples in the validation and test data sets should not be used to train the model.[5]

Most approaches that search through training data for empirical relationships tend to overfit the data, meaning that they can identify and exploit apparent relationships in the training data that do not hold in general.

When a training set is continuously expanded with new data, then this is incremental learning.

Validation data set

[edit]

A validation data set is a data set of examples used to tune the hyperparameters (i.e. the architecture) of a model. It is sometimes also called the development set or the "dev set".[13] An example of a hyperparameter for artificial neural networks includes the number of hidden units in each layer.[9][10] It, as well as the testing set (as mentioned below), should follow the same probability distribution as the training data set.

In order to avoid overfitting, when any classification parameter needs to be adjusted, it is necessary to have a validation data set in addition to the training and test data sets. For example, if the most suitable classifier for the problem is sought, the training data set is used to train the different candidate classifiers, the validation data set is used to compare their performances and decide which one to take and, finally, the test data set is used to obtain the performance characteristics such as accuracy, sensitivity, specificity, F-measure, and so on. The validation data set functions as a hybrid: it is training data used for testing, but neither as part of the low-level training nor as part of the final testing.

The basic process of using a validation data set for model selection (as part of training data set, validation data set, and test data set) is:[10][14]

> Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a training data set. The performance of the networks is then compared by evaluating the error function using an independent validation set, and the network having the smallest error with respect to the validation set is selected. This approach is called the *hold out* method. Since this procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a test set.

An application of this process is in early stopping, where the candidate models are successive iterations of the same network, and training stops when the error on the validation set grows, choosing the previous model (the one with minimum error).

Test data set

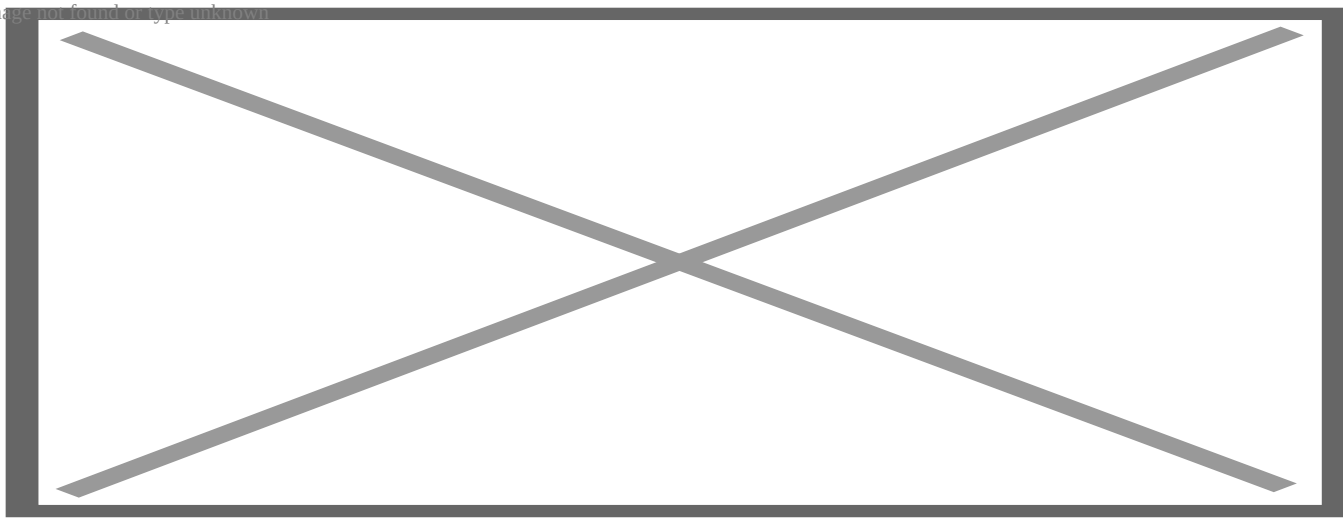
[edit]

A test data set is a data set that is independent of the training data set, but that follows the same probability distribution as the training data set. If a model fit to the training data set also fits the test data set well, minimal overfitting has taken place (see figure below). A better fitting of the training data set as opposed to the test data set usually points to over-fitting.

A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier.[9][10] To do this, the final model is used to predict classifications of examples in the test set. Those predictions are compared to the examples' true classifications to assess the model's accuracy.[11]

In a scenario where both validation and test data sets are used, the test data set is typically used to assess the final model that is selected during the validation process. In the case where the original data set is partitioned into two subsets (training and test data sets), the test data set might assess the model only once (e.g., in the holdout method).[15] Note that some sources advise against such a method.[12] However, when using a method such as cross-validation, two partitions can be sufficient and effective since results are averaged after repeated rounds of model training and testing to help reduce bias and variability.[5][12]

Image not found or type unknown



A training set (left) and a test set (right) from the same statistical population are shown as blue points. Two predictive models are fit to the training data. Both fitted models are plotted with both the training and test sets. In the training set, the MSE of the fit shown in orange is 4 whereas the MSE for the fit shown in green is 9. In the test set, the MSE for the fit shown in orange is 15 and the MSE for the fit shown in green is 13. The orange curve severely overfits the training data, since its MSE increases by almost a factor of four when comparing the test set to the training set. The green curve overfits the training data much less, as its MSE increases by less than a factor of 2.

Confusion in terminology

[edit]

Testing is trying something to find out about it ("To put to the proof; to prove the truth, genuineness, or quality of by experiment" according to the Collaborative International Dictionary of English) and to validate is to prove that something is valid ("To confirm; to render valid" Collaborative International Dictionary of English). With this perspective, the most common use of the terms **test set** and **validation set** is the one here described. However, in both industry and academia, they are sometimes used interchanged, by considering that the internal process is testing different models to improve (test set as a development set) and the final model is the one that needs to be validated before real use with an unseen data (validation set). "The literature on machine learning often reverses the meaning of 'validation' and 'test' sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research."[16] Nevertheless, the important concept that must be kept is that the final set, whether called test or validation, should only be used in the final experiment.

Cross-validation

[edit]

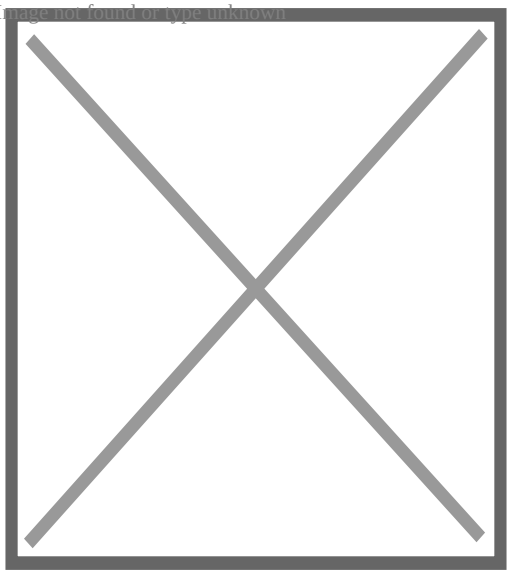
In order to get more stable results and use all valuable data for training, a data set can be repeatedly split into several training and a validation data sets. This is known as cross-validation. To confirm the model's performance, an additional test data set held out from cross-validation is normally used.

It is possible to use cross-validation on training and validation sets, and *within* each training set have further cross-validation for a test set for hyperparameter tuning. This is known as nested cross-validation.

Causes of error

[edit]

Image not found or type unknown



Comic strip demonstrating a fictional erroneous computer output (making a coffee 5 million degrees, from a previous definition of "extra hot"). This can be classified as both a failure in logic and a failure to include various relevant environmental conditions.[17]

Omissions in the training of algorithms are a major cause of erroneous outputs.[17] Types of such omissions include:[17]

- Particular circumstances or variations were not included.
- Obsolete data
- Ambiguous input information
- Inability to change to new environments
- Inability to request help from a human or another AI system when needed

An example of an omission of particular circumstances is a case where a boy was able to unlock the phone because his mother registered her face under indoor, nighttime lighting, a condition which was not appropriately included in the training of the system.[17][18]

Usage of relatively irrelevant input can include situations where algorithms use the background rather than the object of interest for object detection, such as being trained by pictures of sheep on grasslands, leading to a risk that a different object will be interpreted as a sheep if located on a grassland.[17]

See also

[edit]

- Statistical classification
- List of datasets for machine learning research
- Hierarchical classification

References

[edit]

1. **^** *Ron Kohavi; Foster Provost (1998). "Glossary of terms". Machine Learning. **30**: 271–274. doi:10.1023/A:1007411609915.*
2. **^** *Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. New York: Springer. p. vii. ISBN 0-387-31073-8. "Pattern recognition has its origins in engineering, whereas machine learning grew out of computer science. However, these activities can be viewed as two facets of the same field, and together they have undergone substantial development over the past ten years."*
3. ^ ***a b*** *James, Gareth (2013). An Introduction to Statistical Learning: with Applications in R. Springer. p. 176. ISBN 978-1461471370.*
4. ^ ***a b*** *Ripley, Brian (1996). Pattern Recognition and Neural Networks. Cambridge University Press. p. 354. ISBN 978-0521717700.*
5. ^ ***a b c d e f*** *Brownlee, Jason (2017-07-13). "What is the Difference Between Test and Validation Datasets?". Retrieved 2017-10-12.*
6. ^ ***a b*** *Prechelt, Lutz; Geneviève B. Orr (2012-01-01). "Early Stopping — But When?". In Grégoire Montavon; Klaus-Robert Müller (eds.). Neural Networks: Tricks of the Trade. Lecture Notes in Computer Science. Springer Berlin Heidelberg. pp. 53–67. doi:10.1007/978-3-642-35289-8_5. ISBN 978-3-642-35289-8.*
7. **^** *"Machine learning - Is there a rule-of-thumb for how to divide a dataset into training and validation sets?". Stack Overflow. Retrieved 2021-08-12.*
8. **^** *Ferrie, C., & Kaiser, S. (2019). Neural Networks for Babies. Sourcebooks. ISBN 978-1492671206.*cite book: CS1 maint: multiple names: authors list (link)
9. ^ ***a b c*** Ripley, B.D. (1996) *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press, p. 354
10. ^ ***a b c d*** "Subject: What are the population, sample, training set, design set, validation set, and test set?", Neural Network FAQ, part 1 of 7: Introduction (txt), comp.ai.neural-nets, Sarle, W.S., ed. (1997, last modified 2002-05-17)
11. ^ ***a b*** *Larose, D. T.; Larose, C. D. (2014). Discovering knowledge in data : an introduction to data mining. Hoboken: Wiley. doi:10.1002/9781118874059. ISBN 978-0-470-90874-7. OCLC 869460667.*
12. ^ ***a b c*** *Xu, Yun; Goodacre, Royston (2018). "On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning". Journal of Analysis and Testing. **2** (3). Springer Science and Business Media LLC: 249–262. doi:10.1007/s41664-018-0068-2. ISSN 2096-241X. PMC 6373628. PMID 30842888.*
13. **^** *"Deep Learning". Coursera. Retrieved 2021-05-18.*
14. **^** Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press, p. 372
15. **^** *Kohavi, Ron (2001-03-03). "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection". **14**.* cite journal: Cite journal requires

|journal= (help)
16. **^** *Ripley, Brian D. (2008-01-10). "Glossary". Pattern recognition and neural networks. Cambridge University Press. ISBN 9780521717700. OCLC 601063414.*
17. ^ ***a b c d e*** *Chanda SS, Banerjee DN (2022). "Omission and commission errors underlying AI failures". AI Soc.* ***39*** *(3): 1–24. doi:10.1007/s00146-022-01585-x. PMC 9669536. PMID 36415822.*
18. **^** *Greenberg A (2017-11-14). "Watch a 10-Year-Old's Face Unlock His Mom's iPhone X". Wired.*

- v
- t
- e

Artificial intelligence (AI)

- History
  - timeline
- Companies
- Projects

- Parameter
  - Hyperparameter
- Loss functions
- Regression
  - Bias–variance tradeoff
  - Double descent
  - Overfitting
- Clustering
- Gradient descent
  - SGD
  - Quasi-Newton method
  - Conjugate gradient method
- Backpropagation
- Attention
- Convolution
- Normalization
  - Batchnorm
- Activation
  - Softmax
  - Sigmoid
  - Rectifier
- Gating

Concepts

- Weight initialization
- Regularization
- Datasets
  - Augmentation
- Prompt engineering
- Reinforcement learning
  - Q-learning
  - SARSA
  - Imitation
  - Policy gradient
- Diffusion
- Latent diffusion model
- Autoregression
- Adversary
- RAG
- Uncanny valley
- RLHF
- Self-supervised learning
- Reflection
- Recursive self-improvement
- Hallucination
- Word embedding
- Vibe coding

Applications

- Machine learning
  - In-context learning
- Artificial neural network
  - Deep learning
- Language model
  - Large language model
  - NMT
- Reasoning language model
- Model Context Protocol
- Intelligent agent
- Artificial human companion
- Humanity's Last Exam
- Artificial general intelligence (AGI)

Audio–visual	○ AlexNet
	○ WaveNet
	○ Human image synthesis
	○ HWR
	○ OCR
	○ Computer vision
	○ Speech synthesis
	  ○ 15.ai
	  ○ ElevenLabs
	○ Speech recognition
	  ○ Whisper
	○ Facial recognition
	○ AlphaFold
	○ Text-to-image models
	  ○ Aurora
	  ○ DALL-E
	  ○ Firefly
	  ○ Flux
	  ○ Ideogram
	  ○ Imagen
	  ○ Midjourney
	  ○ Recraft
	  ○ Stable Diffusion
	○ Text-to-video models
	  ○ Dream Machine
	  ○ Runway Gen
	  ○ Hailuo AI
	  ○ Kling
	  ○ Sora
	  ○ Veo
	○ Music generation
	  ○ Riffusion
	  ○ Suno AI
	  ○ Udio

Implementations	○ Word2vec
	○ Seq2seq
	○ GloVe
	○ BERT
	○ T5
	○ Llama
	○ Chinchilla AI
	○ PaLM
	○ GPT
	  ○ 1
	  ○ 2
	  ○ 3
	  ○ J
	  ○ ChatGPT

People

- Alan Turing
- Warren Sturgis McCulloch
- Walter Pitts
- John von Neumann
- Claude Shannon
- Shun'ichi Amari
- Kunihiko Fukushima
- Takeo Kanade
- Marvin Minsky
- John McCarthy
- Nathaniel Rochester
- Allen Newell
- Cliff Shaw
- Herbert A. Simon
- Oliver Selfridge
- Frank Rosenblatt
- Bernard Widrow
- Joseph Weizenbaum
- Seymour Papert
- Seppo Linnainmaa
- Paul Werbos
- Geoffrey Hinton
- John Hopfield
- Jürgen Schmidhuber
- Yann LeCun
- Yoshua Bengio
- Lotfi A. Zadeh
- Stephen Grossberg
- Alex Graves
- James Goodnight
- Andrew Ng
- Fei-Fei Li
- Alex Krizhevsky
- Ilya Sutskever
- Oriol Vinyals
- Quoc V. Le
- Ian Goodfellow
- Demis Hassabis
- David Silver
- Andrej Karpathy
- Ashish Vaswani
- Noam Shazeer
- Aidan Gomez
- John Schulman
- Mustafa Suleyman
- Jan Leike
- Daniel Kokotajlo
- François Chollet

- Neural Turing machine
- Differentiable neural computer
- Transformer
  - Vision transformer (ViT)
- Recurrent neural network (RNN)
- Long short-term memory (LSTM)
- Gated recurrent unit (GRU)
- Echo state network

Architectures

- Multilayer perceptron (MLP)
- Convolutional neural network (CNN)
- Residual neural network (RNN)
- Highway network
- Mamba
- Autoencoder
- Variational autoencoder (VAE)
- Generative adversarial network (GAN)
- Graph neural network (GNN)

- Image not found or type unknown Category

About Natural language processing

All-natural language handling (NLP) is the handling of all-natural language info by a computer system. The study of NLP, a subfield of computer science, is generally associated with expert system. NLP is associated with info retrieval, understanding representation, computational grammars, and more extensively with grammars. Significant handling tasks in an NLP system include: speech recognition, message classification, all-natural language understanding, and all-natural language generation.
.

About Natural language understanding

Natural language understanding (NLU) or all-natural language analysis (NLI) is a part of natural language handling in artificial intelligence that manages device reading comprehension. NLU has been considered an AI-hard issue. There is considerable industrial rate of interest in the field due to its application to automated thinking, maker translation, concern answering, news-gathering, message categorization, voice-activation, archiving, and large-scale web content evaluation.
.

Check our other pages :

- **How SCQA improves logical flow in AI conversations**
- **Applying ReAct to combine reasoning with actions**
- **Tree of thought as a method for decision exploration**