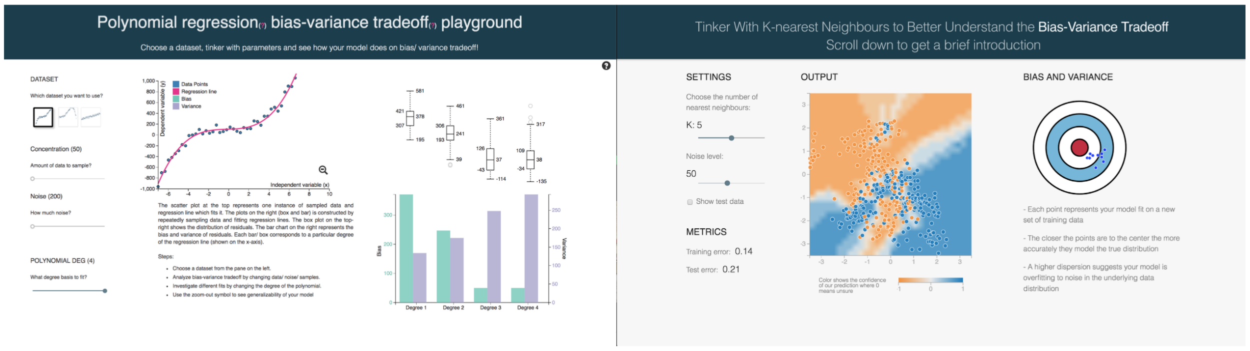
Visualizing the Bias-Variance Tradeoff

Halldor Thorhallsson
halldorb@cs.ubc.ca

Gursimran Singh
msimar@cs.ubc.ca


Polynomial regression(?) bias-variance tradeoff(?) playground

Choose a dataset, tinker with parameters and see how your model does on bias/ variance tradeoff!

DATASET

Which dataset you want to use?

Concentration (50)

Amount of data to sample?

Noise (200)

How much noise?

POLYNOMIAL DEG (4)

What degree basis to fit?

The scatter plot at the top represents one instance of sampled data and regression line which fits it. The plots on the right (box and bar) is constructed by repeatedly sampling data and fitting regression lines. The box plot on the top-right shows the distribution of residuals. The bar chart on the right represents the bias and variance of residuals. Each bar/ box corresponds to a particular degree of the regression line (shown on the x-axis).

Steps:

- Choose a dataset from the pane on the left.
- Analyze bias-variance tradeoff by changing data/ noise/ samples.
- Investigate different fits by changing the degree of the polynomial.
- Use the zoom-out symbol to see generalizability of your model

Tinker With K-nearest Neighbours to Better Understand the Bias-Variance Tradeoff
Scroll down to get a brief introduction

SETTINGS

Choose the number of nearest neighbours:

K: 5

Noise level:

50

Show test data

METRICS

Training error: 0.14

Test error: 0.21

BIAS AND VARIANCE

- Each point represents your model fit on a new set of training data
- The closer the points are to the center the more accurately they model the true distribution
- A higher dispersion suggests your model is overfitting to noise in the underlying data distribution

Fig. 1: Overview of the two playgrounds for explaining the bias-variance tradeoff in learning algorithms. Left: Polynomial Regression Playground; right: K-Nearest Neighbor Playground.

Abstract—This paper describes a visualization tool, aimed at novice machine learning students, for helping them obtain an intuitive understanding of one of the most fundamental concept in Machine Learning: The Bias-Variance Tradeoff. The tool uses two simple learning algorithms Polynomial Regression, and K-Nearest Neighbours, to cover the tradeoff for both classification and regression tasks. Users are able to adjust the hyperparameters of their models and the characteristics of the data itself. A change in any single parameter results in an automatic update to all visualisations that display in an intuitive way how the user's choice of parameters affect the tradeoff. Both Polynomial Regression and k-Nearest Neighbours playgrounds are available online at [34] and [38].

Index Terms—Machine Learning, education, interactive explainables

1 Introduction

Machine learning is at the forefront of many technological breakthroughs of the last decade like speech recognition [15,28], computer vision [7,20], language understanding [33,37] and driverless cars [32]. Since these breakthroughs happened in a relatively short span of time, existing pedagogical methods were inadequate to generate enough skill power, creating a demand-supply gap [17]. In the recent years, we have seen interesting attempts through MOOCs [27], blog articles [12] and intuitive visualizations [16,36] to cover this gap. Despite these resources, students often lack an intuitive understanding of the nuances of various machine learning concepts and algorithms, which behave quite differently with changing parameters.

For instance, nearly all machine learning algorithms allow a user to control the complexity of the model through parameters. The practitioners want their model to be complex enough to capture sophisticated relationships in the data while keeping it simple to prevent noise from affecting the outcome. This leads to a fundamental tradeoff known as the bias-variance tradeoff which is of paramount importance for optimal choice of the hyperparameters for the learning algorithms. Numerous machine learning resources have attempted to explain this fundamental concept which, albeit well presented, often lack the ability for a student to gain an intuitive understanding.

We argue that this lack of understanding is primarily due to the inability to play around with the parameters of algorithms. In order to give an intuitive understanding of the bias-variance tradeoff, we propose to develop interactive visualizations for two classical machine learning algorithms. It will allow students to tinker and experiment with the algorithms and their parameters, and in the process students will develop a strong understanding of the various machine learning concepts like the bias-variance tradeoff.

Our tool, an interactive visualisation system, is aimed at demystifying the bias-variance tradeoff for aspiring ML professionals. The tool is twofold, where the two parts focus on different machine learning algorithms and have slightly different interactive visualizations as to explain the bias-variance tradeoff in a comprehensive manner. The two, relatively simple, machine learning algorithms are polynomial regression models and K-nearest neighbours (KNN). The reason for picking these two particular algorithms is that they are simple and intuitive and encompass both regression and classification problems. Users taking their first steps in machine learning should therefore already be familiar with the algorithms or pick them up fairly quickly with the supplementary text accompanying our tool. The algorithms themselves are not the real focus of our tool, but rather understanding how adjusting hyperparameters affects the bias-variance tradeoff. Our system is not only meant for tinkering with parameters in a so-called playground, but is meant to provide a guided experience through text and interactive visualizations where by the end of the process the user should have an intuitive understanding of the bias-variance tradeoff.

The remainder of this paper is structured as follows. In Section 2 we will go over selected previous work done on visualizing the bias-variance tradeoff and visualizing fundamental machine learning concepts in general. We will abstract the data and tasks for the problem we are trying to solve with our visualization in Section 3. In Section 4 we will go over our visualization tool and how it solves the tasks mentioned in Section 3. In Section 5 we will go over the implementation details of our tool. Section 6 conducts a walkthrough of our visualization tool explaining a few use-cases. In Section 7 we will reflect on our tool and go over future work. Finally, Section 8 contains our conclusion.

2 Related Work

Many articles [5,12,22,36] explain basic concepts in machine learning like bias-variance tradeoff, linear regression and k-Nearest neighbours. These, albeit, well written and supplemented with static visualizations, lacks interactivity. There is no facility for students to play with the parameters and get a sense of how the algorithms behave differently. For instance, the relationship between KNN's hyperparameter k and the bias-variance tradeoff has been explained in great detail by Scott Fortmann [12] in his online article. Similarly, [41] provides an intuitive explanation on the interpretation of RMSE loss for linear regression. A very nice intuitive and animated visualization story has been made for decision trees by Stephanie Yee and Tony Chu [8]. However, due to lack of interactivity and user's ability to see how algorithms behave with different parameters, the articles do not give a good intuition and provide limited understanding.

Other articles [18, 24] supplement text with some level of interactivity. This allows the user to explore concepts beyond what is presented in static visualizations. A recent journal, distill [10], presents excellent articles on machine learning and allow interactivity using a variety of controls. For instance, the article [14] explains the importance of momentum in gradient descent and illustrates it in the case of polynomial regression. However, many of these articles [6, 25, 26], focus on advanced concepts like neural networks and does not touch upon the fundamentals. Another set of articles, use similar ideas of limited exploration and guided explanations to explain basic concepts like principal component analysis [31], image kernels [29], and ordinary least squares [30]. We direct the reader to [35], which presents a list of such articles. However, none of these articles cover the fundamental concept of bias-variance tradeoff.

Also, all of this work only allows limited interaction with the aim of telling a particular story. These are not presented in a playground setting and does not allow a full exploration of concepts by changing multiple parameters. AISpace [19], provides interactive Java applets for teaching and exploring basic concepts in Artificial Intelligence. These tools take a playground kind of approach and allow experimentation beyond a constrained and guided setting. The validity and effectiveness of AISpace in teaching artificial intelligence has been established in [1]. Another interesting work in this domain is TensorFlow Playground [11] which provides an effective tool to explore and tinker the various parameters of a neural network and inspect its output. Both these tools are effective for exploration but does not touch upon bias-variance tradeoff. We take these as inspiration to design a playground for teaching bias-variance tardeoff to students.

3 Data and Task Abstractions

Munzner [23] presented a high-level framework to reason about visualizations. It asks three questions pertaining to visualizations - *what* data is to be presented, *why* you want to visualize, and *how* the visual encodings and interaction idioms will solve the problem. We discuss the *what* and *why* questions in section 3 and *how* in Section 4.

Moreover it suggests thinking these two questions in terms of domain-specific and abstract terminology. The domain-specific terminology discusses in terms of specific data-items we want to visualise and specific objectives we want to explain, while the abstract terminology discusses these in terms of generic visualisation concerns and idioms, without getting into specific knowledge of the domain.

We present the *what* and *why* questions in terms of domain-specific and abstract terminology. The former talks about the specific data and objectives in domain terminology, while the later talks in generic visualization terminology. The generic terminology helps us to compare our specific visualization problem with existing solutions and think about the space of vis design idioms systematically.

3.1 Domain description

The bias-variance tradeoff [13] is a fundamental tradeoff between reducing the two sources of errors due to which learning algorithms fail to estimate the target function (or generalize on unseen data). The expected generalization error of a learning algorithm can be written as the sum of these two sources of errors, namely the *bias* and *variance*.

Consider an independent variables denoted by X and a dependent variable denoted by Y which are related to each other by a relation like $Y = f(x) + \varepsilon$. Using a learning algorithm, we estimate a model $\hat{f}(x)$ of $f(x)$, whose expected error can be written as

$$\begin{aligned} Err(x) &= E((Y - \hat{f}(x))^2) \\ Err(x) &= (E(\hat{f}(x) - f(x)))^2 + E((\hat{f}(x) - E(\hat{f}(x)))^2) + \sigma_\varepsilon^2 \qquad (1) \\ Err(x) &= Bias^2 + Variance + IrreducibleError \end{aligned}$$

The first term is called *bias*, which pertains to erroneous simplifying-assumptions in the learning algorithm. The second term is the variance, which corresponds to sensitivity to inputs (and hence noise) in the training data. The third term is irreducible error, which corresponds to noise in the true function itself. An algorithm with high bias might not have enough flexibility to model the target function $f(x)$ (*underfitting*). On the other hand, a model with high variance, tends to model the random noise in data along with the target function (*overfitting*).

3.2 Domain task

The main goal of our visualization tool is to help students gain an intuitive understanding of the bias-variance tradeoff in learning algorithms. Given a set of data, we wish students should be able to experiment and try machine learning models with different complexity. On one hand, they should get visual feedback on how well their model approximated the training data, and the test data (*overfitting vs underfitting*). On the other hand, they should be able to get visual feedback on how their model is expected to perform on average (*bias vs variance*). As a result, they should be able to understand these two concepts and reason about their relationship. Finally, students should be able to realize the tradeoff - trying to decrease one, increases other and they wont be able to minimize both to full extent. As part of our secondary goal, students should understand how the tradeoff is affected by the data-sampling parameters like amount of noise, complexity and amount of data. We break the above goals and discuss these in detail below.

Task 1. Identify overfitting and underfitting - We want students to visually analyze training data along with their model's predictions and have some intuition on how well their model fits the training data and how well it captures the target function. By trying out models of different complexity, the students are expected to realize that complex models tend to model random noise along with the underlying true function. As a result, they don't generalize well on test data (overfitting). On the other hand, simple models, are not expressive enough to capture the complex relationships in the true function. Hence, they do poor on both training data and test data (underfitting).

Task 2. Understand bias and variance - We want students to understand that bias and variance are expectations, and hence they are computed by sampling multiple instances of data and training multiple models. For a specific input, the predictions of these models are collated. These collated predictions are used to compute bias and variance using the equation 1. Also, students should acquire a visual interpretation of equation 1: bias is the difference between expected prediction and true prediction, and variance is the spread of different predictions.

Task 3. Relate bias and variance to overfitting and underfitting - Through the visualization, the student should be able to intuit that the model which overfits has high variance. Students should get some visual feedback that this happens because overfitting models are sensitive to inputs. Hence if they resample data, a very small change in noise, will give a very different model. This leads to variance being very high. On the other hand, they should intuit that a model which underfits has high bias. Through visual feedback, they should understand that these models give quite similar predictions most of the time. However, they will always be quite off from the true predictions.

Task 4. Bias-variance tradeoff - We want students to internalize the fundamental tradeoff. They should visually analyze that if they try to reduce bias by making the model complex, it increases the variance. On the other hand, if they try to reduce the variance by making the model simple, the bias increases. In either case, they get a bad fit. We wants students to understand this tradeoff using different visualization idioms supporting investigation at multiple levels of detail.

Task 5. How the tradeoff changes - Finally we want students to tinker with the parameters of sampling algorithms like the complexity of datasets, amount of noise, amount of data and see how the fundamental tradeoff changes.

3.3 Domain data

For our visualization system, we predefine a set of true functions $f(x)$, and a set of possible choice of parameters P. We further categorize the parameters P into P_s, belonging to sampling algorithm and P_p belonging to prediction algorithm.

Sampling Algorithm - The true function $f(x)$ is hidden from the system, and instead we define a sampling algorithm. For some values of input X, we obtain values Y from the sampling algorithm as per the following equation:

$$Y = f(X) + \varepsilon$$

For the regression data, we use a Gaussian distribution with zero mean and standard deviation P_p such that $\varepsilon = \mathcal{N}(0, P_p)$. The value of sampling parameter P_p controls the amount of noise. In our prototype, we let the user control the value of P_p using a slider.

For the classification data it is important to have a non-linear decision boundary so that adjusting the number of nearest neighbours k will affect the bias and the variance of the model significantly. We accomplish this by choosing two so called meta-centroids, μ_1 and μ_2, so that one meta-centroid is in the upper left corner and one in the lower right corner of a 2D grid. We then sample 10 points from each of these two bivariate Gaussian distributions: $\mathcal{N}(\mu_1, \Sigma_{mean})$ and $\mathcal{N}(\mu_2, \Sigma_{mean})$ where Σ_{mean} is a fixed covariance matrix. We then sample 400 datapoints using the 20 points sampled previously as means. The two sets of 10 points represent two distinct classes that K-Nearest Neighbour will try to predict so 200 datapoints will be sampled from each set of 10 points. To elaborate: for each class we pick one of the 10 means generated at random (p_k) and sample a datapoint from the distribution: $\mathcal{N}(p_k, \Sigma_{data})$ where again, $\Sigma_d ata$ is a fixed covariance matrix for all datapoints.

Prediction algorithm - We fit a machine learning model $\hat{f}(X)$ to the data sampled using the sampling algorithm. For the given values of X, we obtain Y_{hat} using the model $\hat{f}(X)$. For classification, we use k-nearest neighbors (KNN) to learn $\hat{f}(X)$ and for regression we use polynomial regression with a basis of degree p. In our prototype, we let the user control the value of p using a slider.

Training and testing - To show testing error, we repeat the above process for a new values of X^T and obtain corresponding values Y^T, and Y^T_{hat}. For regression, we show generalization error by using values of X, which go beyond the values used while training.

Resampling and predictions - Using the process defined above, we obtain a data triplet $T_i = (X, Y, Y_{hat})$. We repeat this process multiple times to obtain a set of data triplets $T = \{T_1, T_2..T_n\}$. These sets of triplets are used to compute derived statistics like bias and variance.

Derived statistics - We compute derived statistics like bias and variance using equation 1 and residuals using the formula - $R_i(x) = \hat{f}_i(x) - f_i(x)$.

3.4 Abstract task

At the highest level, the goal of our visualization tool is to present information for *consumption* in pedagogical context. At the mid level, the goal of students is to *search and explore*, try out different combinations of parameters and analyze the behavior of learning algorithms. Regarding targets, the users are mainly interested in *identifying trends and outliers*, like how bias changes as we increase the degree of a polynomial. At the lowest level the user of our visualization system, wants to *explore* with three types of *queries* - identify, compare and summarize.

Identify - We expect the users to identify item characteristics like overfitting, underfitting, bad generalization, good generalization, simple models, complex models, low/ high bias, low/ high variance etc.

Compare - The user should be able to compare derived data for different models like their ability to generalize, complexity, bias, and variance.

Summarize - The user should be able to summarize and get an overview of derived data like residuals, bias and variance as the model's complexity increases. This will help them clearly visualize the bias-variance tradeoff.

3.5 Abstract data

The data consists of quantitative attributes which can be viewed as a set of a unidimensional and a univariate field sampled as a grid. However, we found useful to view it as tables having multiple keys and multiple quantitative attributes. This help us define operations like filtering and aggregation more eloquently. A brief summary of all attributes has been provided in table 1. Its important to note that some of attributes can be vectors.

Table 1: Dataset attributes

Attribute Name	Attribute Type	Description
Model complexity	Ordinal	Complexity of model
Noise	Ordinal	Amount of noise in samples
Sample concentration	Ordinal	Amount of data for training
Sampling ID	Ordinal	ID of a particular sampling
Dataset type	Categorical	Predefined target functions
Train/Test	Categorical	Data for training or testing
X	Quantitative	The independent variables
Y	Quantitative	The dependent variable
Y_{hat}	Quantitative	The prediction of the model

Derived Data: We compute multiple sets of derived data by aggregating data in the table 1. If we aggregate data across individual data points, we can compute metrics like training error and test error. If we do a different aggregate, on say sample ID instead of individual data points, we can compute residuals. If we aggregate on both sample ID and individual data points, we get bias and variance. This happens for each set of parameters P_s. Hence, for each derived data, we get different tables where P_s is the key and values are bias, variance, training error, testing error or residuals. Notice that number of items in these tables can be different.

3.6 What-why-how

- We present a separate what-why-how analysis for each of our visualization idioms in tables 2, 3, 4, 5. This is to reduce clutter in a single table. This also makes sense because we implemented two separate tools (for regression and classification) and we did not use all of our visualization idioms in both our tools.

4 Solution

To understand the bias-variance tradeoff we present two separate tools, one for regression and one for classification. We present a basic learning algorithm in each domain, namely polynomial regression and k-nearest neighbors. An overview of both our prototypes is presented in Fig 1. As discussed in section 3.3 and 3.5, we have multiple tables of raw and derived data. In order to prevent visual clutter, we implement a control panel to reduce the amount of data shown by filtering. This allows the student to interactively change display and queries for different values of parameters (P). In order to effectively support our goals (section 3.2), we implement two levels of visualizations. The first level is the single model view which supports a detailed visual analysis of a subset of our data. It allows the student to probe an instance of training data and the model which fits it. The second level, namely the multiple models view, gives an overview of how different models (trained by resampling data) perform in comparison to the target function. To support investigation of bias and variance at multiple levels of detail, we present three separate visualization idioms - bull's eye plot, box plot and bar plot. We discuss our visualization idioms and design choices in this section.

4.1 Filtering and faceting

In order to support tasks 1, 2 and 5 we want user to compare different models characterized by parameters like model complexity, amount of

Table 2: What, why, how analysis of single model view

System	Single model view
What: Data	Table of data as presented in table 1
Why: Task	Visually identify the complexity of data and that of model
Why: Task	Visually identify cases of overfitting and underfitting
Why: Task	Visually identify how well the model generalizes on unseen data
How: Encode	Express data values (X,Y) with horizontal and vertical spatial position and point marks
How: Encode	Express model with horizontal and vertical spatial position
How: Encode	Use color hue to distinguish data points and model (regression)
How: Encode	Use color hue to distinguish data points (classification)
How: Encode	Use color saturation to represent decision boundary (classification)
How: Manipulate	Filter based on model complexity, amount of noise, dataset type, etc
How: Manipulate	Filter and view test data along with training data

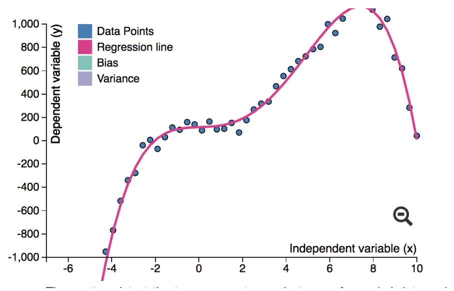
Table 3: What, why, how analysis of bull's eye diagram

System	Multiple models view, Bull's eye plot
What: Data	Quantitative attributes representing sum of residuals for each data sample
Why: Task	Identify the bias and variance for a particular model
How: Encode	Point marks express each model
How: Encode	Position channel is used to encode the error from the target function (bulls eye)
How: Encode	Spread of point marks encode the variance
How: Reduce	Item aggregation

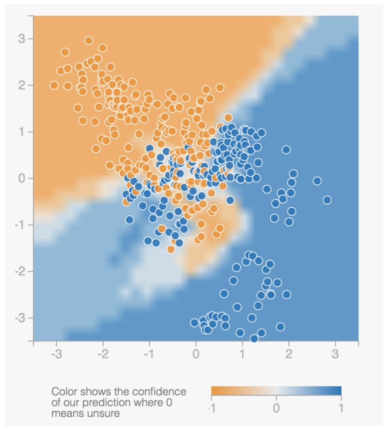
noise, amount of sampling and dataset type. Since, this corresponds to a large amount of data, trying to show everything at once, or by superimposing, leads to immense visual clutter. We solve this problem by providing a control panel on the left side, which allows a user to select a particular set of categorical values using sliders and buttons. Similarly, we facet single model view and multiple-models view to allow investigation at multiple levels of detail. Also, to allow simultaneously viewing, we present a multiform view by juxtaposing the two views side by side as shown in Fig 1. This helps him relate overfitting and underfitting (shown in single model view) with bias and variance (shown in multiple model view) and hence support our task 3.

4.2 Single model view

Single model view is made by plotting the data and model predictions on the same curve and using a color channel to distinguish them. This view allows the user to visually analyze training data along with their model's predictions in detail. This helps him have an intuition on how well their model fit the training data and how well it captures the target function. Visually viewing the data allows the user to identify overtrained and undertrained models. This supports task 1, 3 and 4. For our trivial datasets visually identifying these cases are fairly easy and intuitive for the user rather than analyzing train and test errors. For our two prototypes of regression and classification, we implement separate but similar idioms (Fig 2[a] and Fig 2[b])). In regression, we show a regression line and in classification we show decision boundary along with a gradient. Check out details in section 6. A summary of what, why, and how analysis of the single model view is described in table 2.



(a) Single model view - Polynomial Regression



(b) Single model view - k-Nearest Neighbours

Fig. 2: We plot the data and model predictions on the same curve and use a color channel to distinguish them. In (a), we show a regression line and in (b) we show decision boundary along with a gradient.

4.3 Multiple model view

In multiple model view, we illustrate the characteristics of models trained by sampling multiple instances of data. We visualize this information to help students get an intuition of bias and variance. To achieve this, we propose three different visualization idioms which allow investigation from multiple perspectives. These idioms vary in terms of intuitiveness, accuracy, level of detail, level of aggregation etc. We hypothesize that each idiom supports understanding bias and variance for various levels of background the students posses. Hence we provide one or more instances of these idioms in our visualization tools. We now describe each of these idioms in detail.

4.3.1 Bull's eye diagram

In the bulls eye diagram (shown in Fig 3), each dart or point represents one model trained by resampling data. The center of bull's eye represent the area of zero error and as we move away from the center, the error of the model increases. The middle of the area where all the darts lands roughly corresponds to bias and the spread of the darts roughly corresponds to variance. Table 3 presents a summary of what, why, and how analysis for this idiom.

4.3.2 Box plot

In this diagram, we plot residuals which are obtained by models trained on different samples of data (resampling). We obtain these residuals using the procedure explained in section 3.3. The quantitative residuals is mapped to the vertical axis and the categorical key attribute (model complexity) to the horizontal one. Each box in the box plot diagram (shown in Fig. 4) represents the distribution characteristics of residuals. As the box shifts downwards, it represents lower values of residuals which corresponds to low bias. On the other hand, the height of the bar represents the spread of residuals which roughly corresponds to variance. This diagram allows us to view the bias and variance as the

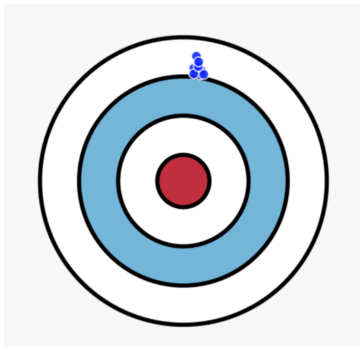


Fig. 3: Bull's eye diagram showing high bias and low variance The center of bull's eye represent the area of zero error and as we move away from the center, the error of the model increases. The middle of the area where all the darts lands roughly corresponds to bias and the spread of the darts roughly corresponds to variance.

degree of polynomial changes shown in Figure 4. As we move towards the right (model complexity increases), the boxes shift downwards (low bias) while their height increases (high variance). Similarly if we move towards the left, the boxes show high bias and low variance. Notice that the scale of bias and variance can be very different. So, in order to ensure discriminability, we have scaled the variance of residuals to match the scale of bias. Table 4 presents a summary of what, why, and how analysis of this idiom.

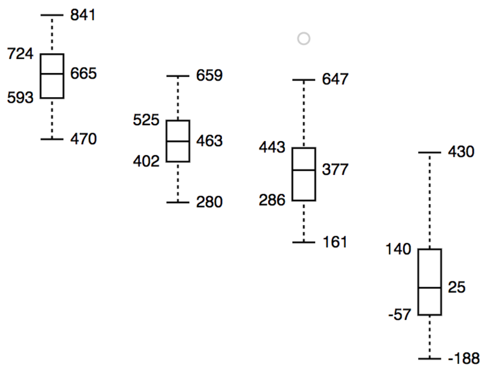


Fig. 4: Box plot showing bias variance tardeoff. As we move towards the right (model complexity increases), the boxes shift downwards (low bias) while their height increases (high variance). Similarly if we move towards the left, the boxes show high bias and low variance.

4.3.3 Dual scale bar plot

In this visualization idiom, we represent the bias and variance using a dual-scale bar chart as shown in Fig. 5. The green bars encode the actual value of bias and purple bar represents variance using vertical spatial position. Bias and variance for different complexity of models (polynomial degree) is separated and ordered by categorical variable. This diagram effectively demonstrates the bias-variance tradeoff without any extra cognitive load of the student. As shown in the Fig. 5, if we move towards right (model complexity increases), the bias decreases but variance increases. On the other hand, if we move towards the left (model complexity decreases), the variance decreases but the bias increases. Notice that the scales for bias and variance can be very different. The decision to display them on separate scales helps to clearly illustrate the tradeoff. As a result, a student will always observe the accurate picture of the tradeoff (Task 4).

Table 4: What, why, how analysis of box plot

System	Multiple models view, Box plot
What: Data	Table of ordinal attributes representing model complexity (key) and one quantitative attribute representing residuals (value)
Why: Task	Characterize distribution of residuals and understand bias and variance
Why: Task	Compare bias and variance of models with different complexity
How: Encode	One box-plot glyph per ordinal attribute expressing derived attribute values like median, quantiles, outliers, etc. 1D list alignment of glyphs separated with horizontal spatial position
How: Reduce	Item aggregation

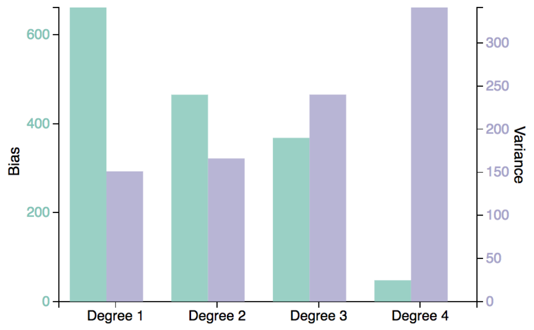


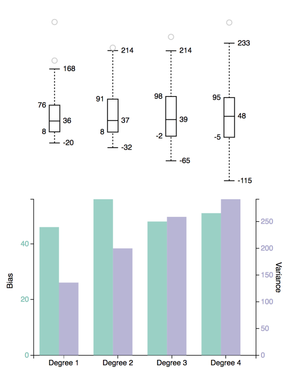
Fig. 5: Dual scale bar plot diagram showing bias and variance tradeoff. As we move towards the right (model complexity increases), bias decreases but variance increases. Similarly if we move towards the left, variance decreases but the bias increases.

4.3.4 Discussion

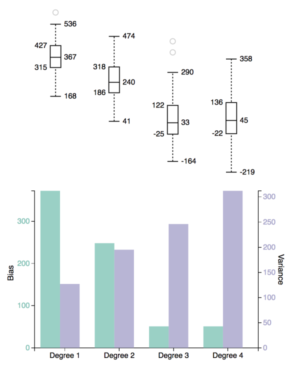
We presented three different visualization idioms to highlight four ideas: multiple models, bias, variance and the tradeoff. As discussed earlier, each idiom presents a different perspective and has different strengths and weaknesses. The bull's eye plot intuitively illustrate the idea of multiple models, bias and variance (Task 2 and 3). However, the tradeoff is not clearly visible (Task 4) and it imparts cognitive load on the user to remember it for different model-complexity. The box and bar plot supports task 4 better since we present the bias and variance of all models on the same view. In bar plot, we aggregate all models (one bull's eye diagram) and present it as a set of two bars (green and purple). This idiom clearly shows the tradeoff and minimal analysis is required. However, due to aggregation students may not be able to perceive the idea of multiple models. Hence doing bad on task 2. The box plot lies in the middle of the two, where each box encodes the idea of a distribution while the tradeoff is visible with some cognitive load. Hence it stands somewhere in the middle on task 2 and 4.

5 Implementation

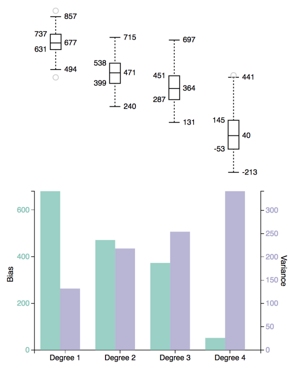
Our visualization tool is implemented as a web application using **HTML**, **CSS** and **JS**. We used the **D3.js** library for the visualizations. We used the **math.js** [9] library for mathematical calculations. We developed each part separately, Halldor coded the KNN part of the tool and Gursimran the Polynomial Regression part. We discussed conceptual parts and the overall design together. The general structure of our UI is inspired by TensorFlow Playground [12], that is the color scheme of our backgrounds, sliders and buttons and the three column separation of adjustable settings, output and visualizations.



(a) Linear looking dataset



(b) Cubic dataset



(c) Bi-quadratic dataset

Fig. 6: Bias-variance tradeoff in datasets of varying complexity. Notice in case of linear dataset (a), the bias does not change significantly. In case of cubic (b), bias does not change after degree three. In case of bi-quadratic (c), bias decreases uniformly. The bias saturates because any increase in degree of polynomial does not give us any better fit. This can be observed by trying out these models in the single-model view.

Table 5: What, why, how analysis for bar plot

System	Multiple models view, Bar plot
What: Data	Table of ordinal attributes representing model complexity (key) and two quantitative attribute representing bias and variance
Why: Task	Identify, compare and summarize the bias and variance with model complexity
How: Encode	Point marks express quantitative attributes with vertical position
How: Encode	Color hue to distinguish the two quantitative attributes and scales
How: Encode	1D list alignment of ordinal key attribute separated with horizontal position
How: Reduce	Item aggregation

5.1 Polynomial Regression Playground

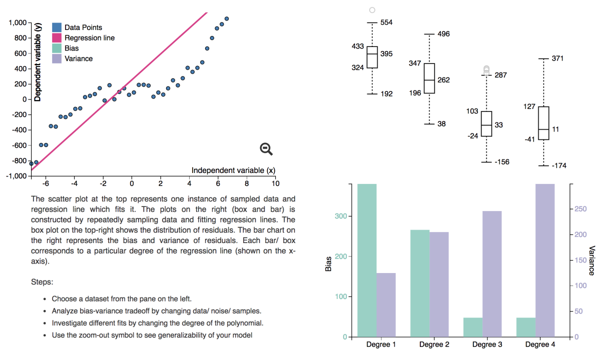
The project uses *bower* [4] to document JavaScript dependencies. Interactive walkthroughs are provided with the help of *intro.js* [40]. The code for UI is present in *index.html* and *styles.css*. The JS driver script for the Polynomial Regression playground is *index.html*. It calls javascript code for different components which is present inside the folder *js*. Visualization idioms are implemented in the files - *box.js*, *box_plot.js*, *bar_plot.js*, *line_plot.js*. The code for box plot and bar plot is adapted from [3] and [21]. The code for computing bias and variance is present in *bias-var.js*. The functions written to generate, sample data, train models and fit regression line are implemented in *ml-functions.js*. All the UI events handling code is implemented in *event_handling.js*.

5.2 KNN Playground

The main logic for the KNN part of the tool that connects all the different parts together is implemented in *index.html* file. The data generating part is implemented in *data.js*. We use an optimized data structure for finding the K-nearest neighbours of a given point that is called a KD-tree [2] using a Javascript library for its implementation [39]. The implementation of the heatmap is in *heatmap.js*. The heatmap is inspired from Tensorflow Playground's heatmap [11]. We use the same technique for generating our heatmap (**HTML** canvas element) and use Tensorflow Playground's legend and color gradient. The dartboard is inspired from Scott's Fortmann static image [12] and we regenerated it using SVG. All the dartboard related code, including the calculation of bias and variance, is in *dartboard.js*.

6 Results

We walk through usage scenarios in a comprehensive manner for both parts of our tool in the following subsections:



The scatter plot at the top represents one instance of sampled data and regression line which fits it. The plots on the right (box and bar) is constructed by repeatedly sampling data and fitting regression lines. The box plot on the top-right shows the distribution of residuals. The bar chart on the right represents the bias and variance of residuals. Each bar/ box corresponds to a particular degree of the regression line (shown on the x-axis).

Steps:

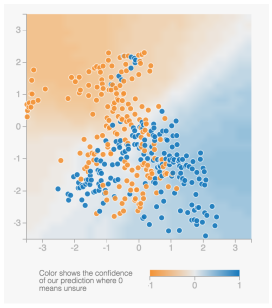
- Choose a dataset from the pane on the left.
- Analyze bias-variance tradeoff by changing data/ noise/ samples.
- Investigate different fits by changing the degree of the polynomial.
- Use the zoom-out symbol to see generalizability of your model

Fig. 7: The degree of model is not enough to capture the relationships in the data. This leads to underfitting as shown in the scatter plot. Notice the box plot corresponding to degree 1, its high up and small in height. This corresponds to high bias and low variance. This can also be seen in the bar chart.

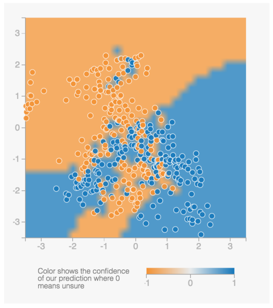
6.1 Polynomial Regression Playground

In section 3.2, we presented various tasks which our tool supports to understand the bias variance tradeoff. In this section we present various user scenarios that a student may follow. When a student encounters the system for the first time, he is presented with the polynomial regression playground shown in Fig 1[left]. Since the student may not be aware of the use cases, the system presents an *interactive walkthrough*. The walkthrough explains different components of the tools and introduces the student to various controls for tweaking and exploring the bias variance tradeoff. If a student is not familiar with the bias variance tradeoff, we have a small discussion at the bottom which introduces him to the topic and offers external resources for further exploration. A more advanced student may want to skip the walkthrough and discussion and go directly to the playground.

Overview: Recall the structure of the tool presented in section 4. The left side of the tool presents a control panel which allows the student to control the parameters like dataset type, amount of noise, amount of samples, and degree of polynomial. In the middle, we present a scatter plot (shown in Fig 2[a]) as the single model view for polynomial regression. On the right, we present two plots to show the



(a) K=100



(b) K=1

Fig. 8: The decision boundary for different values of k. When the value of k is high (a), we get a simple decision boundary (underfitting). When the value of k is low (b), we get a more complex decision boundary (overfitting)

bias variance tradeoff, namely box plot (shown in Fig 4) and the dual scale bar plot (shown in Fig 5). By default the playground presents a default choice of dataset and controls.

Scatter plot - The center visualization idiom presents the student with the dataset and the regression line. The student may want to explore by choosing a different degree of polynomial and inspecting the fit. Also, the student may wish to use the zoom-out button to inspect how well the model generalizes. This gives him the understanding of overfitting and underfitting (Task 1).

Box plot - At the top-right, the student inspects the box plot which makes him think about the underlying distribution where some models did good, others did bad. This reminds the student about multiple models being learnt and hence helps him understand bias and variance. Also, he gets this understanding that bias is the expected error which is approximately represented by the median of the box plot. The box plot also reminds him about the spread of the errors, which roughly corresponds to variance. Thus it helps in supporting task 2. Finally he visualizes the tradeoff as he observes the certain pattern (discussed in section 4.3.2) as he moves right in the curve (Task 4).

Bar plot - At the bottom right, the student observes the bias variance tradeoff in a dual scale bar plot. As explained in the section 4.3.3, it clearly and effectively demonstrates the bias-variance tradeoff as the model complexity changes.

Exploring individual fits - Once the student observes how the bias-variance changes with the model complexity, the student may wish to explore how the models look like. With this aim in mind, the student controls the slider for degree of polynomial and observes different fits. The student observes that an underfit model corresponds to high bias and low variance. The student makes this observation in box plot as well where the student sees the box high up with little spread (Fig 7). Hence the model is not sensitive to inputs and hence noise. Similarly the student observes the case of overfitting when the bias is bias is low and variance is high. The student observes that when he tries to fit a degree 3 polynomial with degree 4 regression line, the bias remains almost the same same but variance increases (Fig. 1[left]). Hence the fit is worse. This exercise helps The student relate overfitting and underfitting with bias and variance (Task 3).

Exploring different datasets - The student may want to explore the tradeoff with different datasets. We provide the student with three datasets which help him make different conclusions. The three datasets corresponds to degree 3, degree 4 and degree 1 (training data). The student analyzes the tradeoff in these three cases. Firstly, he observes that in case of linear dataset, the increase in p, does not lead to any decrease in bias and only leads to increase in variance, hence making the model worse (Fig 6[a]). The student observes this phenomenon when p is more than 3 in degree 3 dataset as well (Fig 6[b]). Finally, the student observes that in degree 4 dataset, where higher degree polynomials can lead to better fits, bias is reduced (Fig 6[c]). This gives

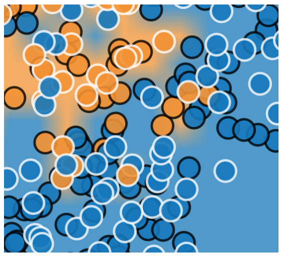


Fig. 9: A zoomed in view of the heatmap with test points. Test points have a black outline while training points have a white one. Because k is very small it overfits to variance in the blue cluster and misclassifies a lot of test datapoints in the orange class. This is a sign of high variance.

the student better understanding that bias corresponds to better fitting of data.

Exploring other parameters - Having gained a solid understanding of bias-variance tradeoff as the model complexity changes, the student may wish to explore the effect of sample size and noise. He may have a certain hypothesis in mind which he wants to test in our playground. When he tries different values using slides, he makes a couple of observations. First he observes that variance reduces if we increase more amount of data but bias does not change. Second he observes increase in noise leads to increase in both bias and variance. The student reasons about bias and variance by observing box plot and concludes that all observations make sense (Task 5).

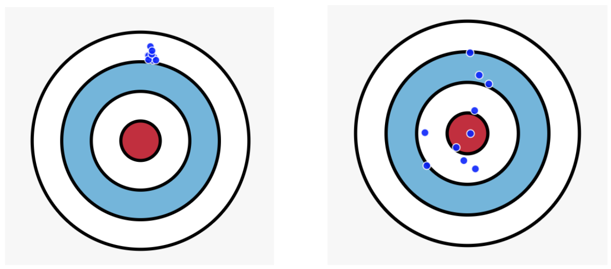
6.2 KNN playground

An overview of the KNN playground is presented in Fig 1[right]. Users will see instructions in the header to scroll down for a brief explanation of the bias-variance tradeoff and K-nearest neighbours. More advanced users may wish to skip this step and start playing with the playground right away. The text that appears when users scroll down familiarizes the student with key concepts such as training error, test error and the bias-variance tradeoff. We also cover the dartboard diagram idiom with static images (originally used by Scott Fortmann [12]) so that when entering the playground later, the user will be up to speed on what the position of the darts represents. Also, further down there is more text explaining the K-nearest neighbour algorithm. For now, users are instructed to go back to the playground and try to get a sense of how adjusting parameters affects the bias-variance tradeoff. However, they are also instructed to revisit the text and read on once theyve finished using the playground.

When entering the playground the user can start adjusting the sliders on the left (see leftmost column of Fig. 1[right]) to adjust the *noise* in the data and to set the k parameter for the model. Users can also check the checkbox to view test datapoints on the heatmap. As soon as users adjust the noise or the k parameter a few things happen. Firstly, the decision boundary of the heatmap (See middle column of Fig. 1[right]) changes and secondly the darts will be cleared and reanimated into new positions on the dartboard (See rightmost column of Fig 1[right].).

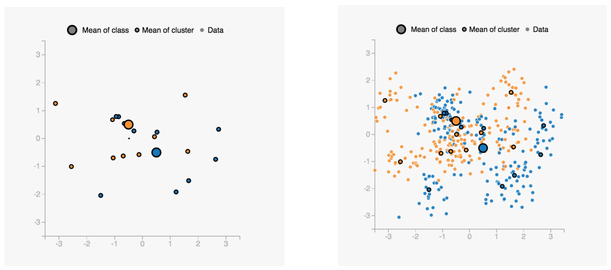
Heatmap: Users should see right away that significantly increasing k results in a simple decision boundary as can be seen in Figure 8[a].The algorithm will start misclassifying large portions of the data as can be seen for the orange cluster at the bottom part of Fig 8[a]. Decreasing k makes the boundary more complicated and will start to fit the training data perfectly (see Fig 8[b]). Then when users check the test data checkbox they will see on the heatmap that a low k misclassifies a lot of points since their color is not the same as the color of the heatmap around them (Task 2).

The *noise* parameter will generate a new set of datapoints on the heatmap and update the heatmap and its the decision boundary accordingly. Users will see that when they increase the noise it becomes more difficult for the model to correctly classify all datapoints and the bound-



(a) High bias and low variance (b) Low bias and high variance

Fig. 10: Dartboard showing bias-variance tradeoff. In (a), the darts are away from the center (high bias) but they are close together (low variance). In (b), darts are closer to center (low bias), but they are spread out (high variance).



(a) (b)

Fig. 11: An overview of the data generating process. In (a), the large points are the centroids of the smaller points who will be the centroids of the clusters for the actual data. In (b), we have the data along with the centroids and meta-centroids. Each small centroid is surrounded by data points.

ary will become less sharp and become wider and more desaturated depicting that the algorithm is less confident on what class to predict in that area. Decreasing the *noise* however will make the clusters more distinct and denser making the decision boundary more precise (that is jagged and very saturated even around boundaries) just like when decreasing k (See Fig 8[b] (Task 5). The users should therefore be starting to get a sense of how the bias and variance of the model change as they adjust parameters. The *noise* does not seem to affect misclassifying whole clusters but increasing k will. This will give users a sense of how increasing k increases bias, that is the model underfits the data while increasing k will increase the variance in the model since test points are not being classified as well (Task 3). Meanwhile, adjusting the noise should show user that it is strongly positively correlated with the variance. Users can also see how adjusting these parameters affects the exact training and test error as seen in the lower left corner of Fig 1[right] (Task 5).

Dartboard: When the user adjusts the *noise* or k the dartboard is cleared and new datapoints, or darts, are animated flying onto the dartboard. The users should have a sense of what the group structure of darts means after reading the text accompanying the playground. When users increase k they will see how the darts become more dense signifying that the variance of the model is becoming less and less. However, the darts are also landing farther and farther from the bullseye, signifying a higher bias (See Fig. 10[a]). When decreasing k users will ultimately see the darts again getting closer to the bullseye which means that the bias is becoming less. At some point they will be getting more and more dispersed however which means that variance is increasing (See Fig. 10[b]). When users increase the *noise* parameter the darts will also become more dispersed signifying a higher variance. However, when the *noise* is decreased the points become less dispersed which means that variance is decreasing. The dartboard is therefore effective at conveying the idea of bias and variance (Task 2) and showing that optimizing for bias and variance is a tradeoff (Task 4).

After adjusting the parameters in the playground the user should have an intuitive understanding on how k and *noise* affect the bias-variance tradeoff. However, hopefully, the user is interested in seeing how the data was generated after playing with the parameters since it always seems to be highly structured . The users remember to scroll back down to where they left off when they started playing with the playground. There, a text describes the data generation process that we describe in Section 3. The users then see the sampling of the datapoints animated on a grid explaining the data generation process visually. By watching the animation, the users gain an intuitive understanding on how the centroid of each cluster of datapoints is generated from a meta-centroid for each respective class (see Fig. 11[a] and Fig. 11[b]). Hopefully, seeing this animated graph, the user will then understand more clearly how the choice of k might affect the classification of the centroids themselves and their immediate area which is central to the classification task (Task 5).

7 Discussion and Future Work

We believe that our visualization tool provides a good starting point for giving users an intuitive understanding of the bias-variance tradeoff in the context of adjusting both hyperparameters and data characteristics. Like mentioned in Section 2, the current options for students wanting to learn more about the tradeoff are mostly math-heavy blog posts that include very intuition or blog posts with mostly static images and very little interactivity.

However, we recognize that there are several limitations with our approach. For one, our visualization tool is segregated into two parts which results in some confusion for users we have tested it on. Having a single view with a toggle for both algorithms (KNN and Polynomial Regression) would have been better than developing them as two completely independent parts. This however, can be mitigated by a simple switch in the visualization tool resulting in being able to toggle between algorithms.

Another limitation is that the user has very little control over the data generating process. There is a tradeoff here, between giving the user full flexibility and how easy it is to see the extremities of the bias-variance tradeoff in action. For example, if a user generated data with a linear decision boundary for the K-nearest neighbours algorithm, increasing k would not result in a higher bias. The user might have to play around with the playground for a long time until the bias-variance tradeoff became intuitive. For this tool, we carefully generated our data to display the extremities of the bias-variance tradeoff. This, we argue, is the best option for teaching novice users about the tradeoff. For more advanced users we could add extra features for more control of the data generating process.

Furthermore, the tool is still a bit too laggy in our opinion. Multiple remedies were made to make the tool faster, like calculating the K-nearest neighbours using KD-trees [2]. However, the tool still results in a ~ 1 second lag after adjusting parameters which limits the dynamic feeling of our visualizations.

8 Conclusion

In this paper we presented a visualization tool aimed at giving novice machine learning students an intuitive understanding of the bias-variance tradeoff. We used two algorithms, K-Nearest neighbours and Polynomial regression, to explain the tradeoff for both classification and regression tasks. To maximize the visual understanding of how hyperparameters affect the tradeoff we had to carefully design appropriate data generating functions. The output of those functions was then used as an input to the learning algorithm with selected hyperparameters adjusted by the user.

We designed multiple visualization idioms to show the visual output of the model, its performance and its bias and variance. Those visualizations provide the user with an intuitive sense of how the complexity of the model and data characteristic affect the tradeoff which is imperative for understanding more complex machine learning concepts.

It is our hope that this tool can be used in courses, both onsite at universities and online, as well as an online resource to convey this fundamental concept to aspiring machine learning professionals in a clear manner.

References

[1] S. Amershi, G. Carenini, C. Conati, A. K. Mackworth, and D. Poole. Pedagogy and usability in interactive algorithm visualizations: Designing and evaluating cispace. *Interacting with Computers*, 20(1):64–96, 2007.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[3] M. Bostock. Box plots, 2017.

[4] bower. Bower. `https://github.com/bower/bower`, 2017.

[5] J. Brownlee. Gentle introduction to the bias-variance trade-off in machine learning, 2016.

[6] S. Carter and M. Nielsen. Using artificial intelligence to augment human intelligence. *Distill*, 2(12):e9, 2017.

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016.

[8] S. Y. . T. Chu. A visual introduction to machine learning, 2015.

[9] J. de Jong. math.js. `https://github.com/josdejong/mathjs`, 2017.

[10] Distill. Distill, 2016.

[11] T. Flow. Tinker with a neural network, 2015.

[12] S. Fortmann. Understanding the bias-variance tradeoff, 2012.

[13] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[14] G. Goh. Why momentum really works. *Distill*, 2017. doi: 10.23915/distill.00006

[15] A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 273–278. IEEE, 2013.

[16] N. Harris. Visualizing k-means clustering, 2014.

[17] B. Insider. There's a shortage of ai engineers in the us, 2017.

[18] A. Karpathy. Convnetjs demo: toy 2d classification with 2-layer neural network, 2014.

[19] B. Knoll, J. Kisynski, G. Carenini, C. Conati, A. Mackworth, and D. Poole. Aispace: Interactive tools for learning artificial intelligence. In *Proc. AAAI 2008 AI Education Workshop*, p. 3, 2008.

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[21] liufly. Dual-scale-d3-bar-chart. `https://github.com/liufly/Dual-scale-D3-Bar-Chart`, 2017.

[22] N. Mccrea. An introduction to machine learning theory and its applications: A visual tutorial with examples, 2014.

[23] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[24] C. Olah. Neural networks, manifolds, and topology, 2014.

[25] C. Olah and S. Carter. Attention and augmented recurrent neural networks. *Distill*, 2016. doi: 10.23915/distill.00001

[26] C. Olah, A. Mordvintsev, and L. Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

[27] L. Pappano. The year of the mooc. *The New York Times*, 2(12):2012, 2012.

[28] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. The kaldi speech recognition toolkit. In *IEEE 2011 workshop on automatic speech recognition and understanding*, number EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[29] V. Powell. Image kernels, 2014.

[30] V. Powell. Ordinary least squares regression, 2014.

[31] V. Powell. Principal component analysis, 2014.

[32] G. Ros, A. Sappa, D. Ponsa, and A. M. Lopez. Visual slam for driverless cars: A brief survey. In *Intelligent Vehicles Symposium (IV) Workshops*, vol. 2, 2012.

[33] R. Sarikaya, G. E. Hinton, and A. Deoras. Application of deep belief networks for natural language understanding. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(4):778–784, 2014.

[34] G. Singh. Polynomial regression playground, 2014.

[35] sp4ke. awesome-explorables, 2017.

[36] Stephanie and Tony. A visual introduction to machine learning, 2017.

[37] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[38] H. Thorhallsson. k-nearest neighbours playground, 2014.

[39] Ubilabs. kd-tree-javascript. `https://github.com/ubilabs/kd-tree-javascript`, 2017.

[40] usablica. Intro.js. `https://github.com/usablica/intro.js`, 2017.

[41] E. Visually. Ordinary least squares regression, 2014.