Theory and use of LIME

Mathias Gilje

April 2024

1 Introduction

With the rise of AI, the problem of interpretation has become paramount. To adress this, the field of interpretable machine learning is rapidly developing new tools that shed light into the mechanics of an otherwise black box.

This paper investigates one such tool. I start by defining LIME and provide a general setup for the paper, making some simplifying model assumptions. With this foundation I am able to show some useful theoretical results on LIME, most of which are from Garreau and Mardaoui [2021]. In particular I prove existence, uniqueness and distribution of the models asymptotic coefficients.

I then implement LIME in Python from scratch to interpret two black-box models: an image classifier and a large language model, comparing my results to the official package. By looking deeper at the transformer architecture, I am able to choose interpretable components such that my implementation outperforms official LIME, albeit at the expense of compromising agnosticity.

Lastly, I extend my assumptions to a linear model with interaction and complexity penalty. The surrogate models perform overall reasonably well, with most post-hoc explanations seeming intuitive from a visual or linguistic standpoint.

2 LIME

2.1 Definition and method

LIME (Ribeiro et al. [2016]) stands for Locally Interpretable Model-agnostic Explanation, and it is a method to interpret a large model with a smaller surrogate.

Formally, we are given a **black-box model** $f : V \to \mathbb{R}$ that we wish to interpret locally in the specific instance $x \in V$. In order to do so we introduce a **surrogate model** $g : \mathbb{R}^d \to \mathbb{R}$ from a class G of interpretable models. Specifically, g does not take x as input but instead d **interpretable components** $z \in \{0,1\}^d$ of x. This means that we can identify the instance vector $x \in V$ with $\mathbf{1} \in \mathbb{R}^d$, and we can manipulate x by turning on or off some of its interpreted components, yielding a new input $x_z \in V$. How this is done in practice will depend upon our choice of interpretable components, making LIME adaptable for various applications.

Our goal is to approximate g to f in the proximity of x. On the one hand to minimize a loss function between $g(z)$ and $f(x_z)$, denoted $\mathcal{L}(f, g)$, and on the other to keep a low complexity of g, denoted $\Omega(g)$, ensuring interpretability.
Finally, we would like to weigh those instances x_z closer to x higher than those x_z far from x. This is done by introducing a **proximity function** $\pi_x : \mathbb{R}^d \to \mathbb{R}$, that should also be taken into account in the loss function. Combining these considerations, the **LIME** $\xi_x : \mathbb{R}^d \to \mathbb{R}$ is given by

$$\xi_x = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{1}$$

2.2 Related work

Ribeiro et al. [2016] coined the term LIME with the paper "Why should I trust you?", adressing the need to interpret models that were used in areas such as medicine, finance and law, as well as detecting bias.

Building on this work, Garreau and Mardaoui [2021] showed weak convergence of coefficients of LIME and a direct expressions of ξ_x for toy-models such as indicator functions and scaled linear models in the case of image classification. I replicate their results but apply their results to an actual black-box model, a Convolutional Neural Network (Krizhevsky et al. [2012]).

Another related work is by Teso and Kersting [2019] where they use LIME on image classification to find and correct overdependencies of a CNN on some parts of an image. I instead focus on the theoretical aspect of proving that the models parameters are sensible as n increases.

General setup

LIME gives many degrees of freedom. Depending on the application, we get to choose the class G, the interpretable components, as well as π_x, the loss $\mathcal{L}$ and the complexity Ω. These choices result in regression problems with varying degrees of difficulty.

Let the rows of the design $Z \in \{0,1\}^{n\times(d+1)}$ be the interpretable components $z_i \in \{0,1\}^d$ for $i = 1, ..., n$, with 0'th column serving as intercept. If we have d interpretable components, there are 2^d combinations, so the design can be either:

1. **Fixed**. For small d, we let $n = 2^d$, and Z is generated by all combinations of rows $z_i \in \{0,1\}^d$.
2. **Stochastic**. For large d, Z has entries $z_{ij} \sim \mathcal{B}(\frac{1}{2})$ with the exception of the first row being $\mathbf{1}^T$.

The proximity function π_x is inverse exponential kernel with cosine distance:

$$\pi_x(z) = \exp\left(\frac{-d_{\cos}(\mathbf{1}, z)^2}{2\nu^2}\right) \qquad d_{\cos}(u, v) = 1 - \frac{u^T v}{||u|| \cdot ||v||}$$

with hyperparameter ν indicating how much the x_z close to x should be weighted compared to those far from x.

Let the loss function $\mathcal{L}(f, g, \pi_x)$ be the weighted least squares error (WLS):

$$\mathcal{L}(f, g, \pi_x) = \sum_{i=1}^{n} \pi_x(z_i) \cdot (f(x_{z_i}) - g(z_i))^2$$

Let Z be stochastic throughout this paper[a]. Then all its derived expressions will also be stochastic. If we introduce **weights** $\pi_i := \pi_x(z_i)$ and **outcomes** $y_i := f(x_{z_i})$, the LIME is given by

$$\xi_x = \arg\min_{g\in G} \sum_{i=1}^{n} \pi_i (y_i - g(z_i))^2 + \Omega(g) \tag{2}$$

[a] Except when specifically stated otherwise.

2.3 Theory

In order to solve the optimization problem, we make some simplifying assumptions.

Assumptions

A1. G is the linear models: $G = \{g : \mathbb{R}^{d+1} \to \mathbb{R} \mid \exists \beta \in \mathbb{R}^{d+1} : g(z) = z^T\beta\}$

A2. There is no complexity penalty: $\Omega(g) = 0$

Define the **diagonal weight matrix** $W \in \mathbb{R}^{n\times n}$ by $W = \text{Diag}[\pi_i]$ and the **outcome vector** $y \in \mathbb{R}^n$ by $y_i = f(x_{z_i})$.

Then by A1, the class G is parametrized by β and by A2 we can remove the complexity penalty, so if we rewrite (2) in terms of W and y, then ξ_x is parametrized by the well-known WLS estimator given by

$$\hat{\beta}_n^{\text{WLS}} = \arg\min_{\beta\in\mathbb{R}^d} \ (y - Z^T\beta)^T W(y - Z^T\beta) \tag{3}$$

With the well-known solution:

Lemma 1. If Z^TWZ is positive definite, the solution to (3) is given by

$$\hat{\beta}_n^{\text{WLS}} = (Z^TWZ)^{-1}ZWy \tag{4}$$

Lets define $\hat{\Sigma}_n := \frac{1}{n}Z^TWZ$ and $\hat{\Gamma}_n := \frac{1}{n}Z^TWy$ and rewrite the above as $\hat{\beta}_n = \hat{\Sigma}_n^{-1}\cdot\hat{\Gamma}_n$.

This motivates the definitions $\Sigma = \mathbb{E}(\hat{\Sigma}_n)$ and $\Gamma = \mathbb{E}(\hat{\Gamma}_n)$.

Lemma 25+26. Garreau and Mardaoui [2021]

$$\hat{\Sigma}_n \xrightarrow{P} \Sigma \text{ with respect to } ||\cdot||_{\text{op}}, \text{ and } \hat{\Gamma}_n \xrightarrow{P} \Gamma.$$

Define $\beta^f := \Sigma^{-1}\cdot\Gamma$. The question is whether $\hat{\beta}_n \xrightarrow{P} \beta^f$?

Theorem 2. Garreau and Mardaoui [2021]

$$\text{Assume } f : \mathbb{R}^m \to \mathbb{R} \text{ is bounded by } M > 0. \text{ Then } \hat{\beta}_n \xrightarrow{P} \beta^f.$$

We call β^f the **asymptotic coefficients**.

Proof. In appendix **A**. The proof is done in 5 parts:

A.1 Compute Σ and Σ^{-1} (showing it is invertible) and Γ.

A.2 Upper bound $||\hat{\beta}_n - \beta^f||$ in terms of $||\Sigma^{-1}||_{\text{op}}$, $||\Gamma||$, $||\hat{\Gamma}_n - \Gamma||$, and $||\hat{\Sigma}_n - \Sigma||_{\text{op}}$.

A.3 Find an upper bound for $||\Sigma^{-1}||_{\text{op}}$ and $||\Gamma||$.

A.4 Prove that $||\hat{\Sigma}_n - \Sigma||_{\text{op}} \to 0$ and $||\hat{\Gamma} - \Gamma|| \to 0$ in probability.

A.5 Put it all together to show $\hat{\beta}_n \xrightarrow{P} \beta^f$.

□

Since the entries of the stochastic Z are Bernoulli variables, there is a formula for calculating Σ in **Lemma 8**, as well as for Σ^{-1} in **Lemma 10** and Γ. Hence we can calculate β^f directly by multiplying $\Sigma^{-1} \cdot \Gamma$.

Corollary 3. Garreau and Mardaoui [2021] The coefficients β_j^f are given by

$$\beta_0^f = c_d^{-1}\Big(\sigma_0 \mathbb{E}(\pi f(x)) + \sigma_1 \sum_{i=1}^{d} \mathbb{E}(\pi z_i f(x))\Big)$$

$$\beta_j^f = c_d^{-1}\Big(\sigma_1 \mathbb{E}(\pi f(x)) + \sigma_2 \mathbb{E}(\pi z_j f(x)) + \sigma_3 \sum_{i=1, i\neq j}^{d} \mathbb{E}(\pi z_k f(x))\Big)$$

for $1 \leq j \leq d$, where $c_d, \sigma_0, ..., \sigma_3$ are defined in **Lemma 10**.

Notice that we have the following result (trivial as it might be)

Corollary 4. β^f is also the WLS estimator of the fixed version of Z

Does this mean we can calculate β^f faster? unfortunately not since the entries of Γ require calculation of all 2^d combinations. For the statistical side, lets add a final assumption.

Assumptions

A3. There exists **optimal coefficients** $\beta^* \in \mathbb{R}^{d+1}$ for the outcome vector y

$$y = Z\beta^* + u \tag{5}$$

with **error vector** $u : \mathcal{X} \to \mathbb{R}^n$ satisfying

$$\mathbb{E}(u \mid Z) = \mathbf{0} \quad \text{and} \quad \mathbb{V}(u \mid Z) = \sigma^2 I \tag{6}$$

for some constant **std. error** $\sigma \in \mathbb{R}$.

This assumption allows us more knowledge of the coefficients.

Theorem 5. $\hat{\beta}_n \xrightarrow{P} \beta^*$ and consequently $\beta^f = \beta^*$. So $\hat{\beta}_n$ is **weakly consistent**.

Theorem 6. $\hat{\beta}_n \overset{\text{as}}{\sim} \mathcal{N}\big(\beta^*, \frac{\sigma^2}{n}\Sigma^{-1}\Omega\Sigma^{-1}\big)$ for a simple covariance matrix $\Omega \in \mathbb{R}^{(d+1)\times(d+1)}$

The proofs following the general method described in Cameron [2013], see appendix. This extends the results shown in Garreau and Mardaoui [2021] since it shows that β^f are in fact "the right" coefficients, and it's an extension of **Corollary 3**, because the entries of the covariance matrix can be calculated directly given σ.

3 Implementation and Examples

I implement LIME from scratch in Python and compare this to the original LIME code from Ribeiro et al. [2016] on an image and text example.
Both examples are classfication problems, with the black-box model returning a discrete probability distribution over o labels. Say we wish to compute LIME on an input instance vector $x \in V$. First we fix the top label with x as input. Then $f : V \to \mathbb{R}$ returns the probability of this label as we trudge the interpretable components of x.

3.1 Example 1: MNIST

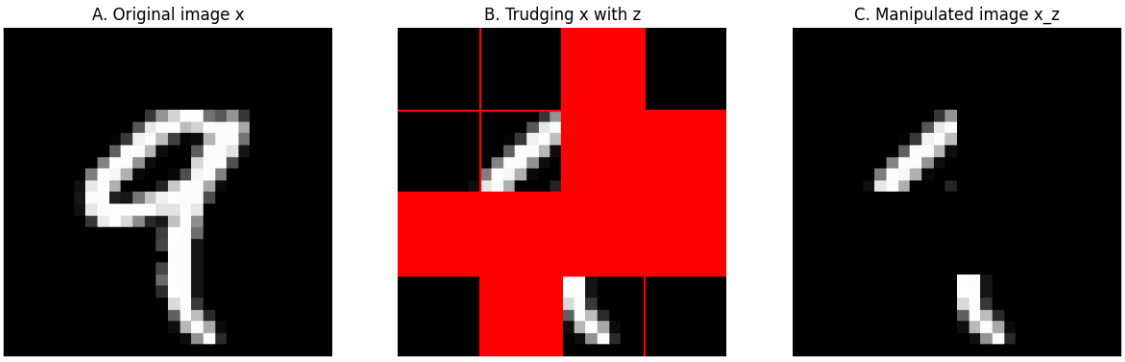
First I trained a Convolutional Neural Network[1] (CNN) on the MNIST database with PyTorch. I used the same architecture as DeepLift[2], with an Adam optimizer,

[1] **?**

[2] Shrikumar et al. [2017] see their Appendix D for details.

Cross Entropy Loss function, learning rate $\text{lr} = 10^{-3}$ and batch size 256 over two epochs, with test score of $\sim 97\%$.

LIME has been used on MNIST models by Garreau and Mardaoui [2021], where they used the state of the art Quickshift algorithm[3] to partition the image into *superpixels* that serve as interpretable components. This means that given $x \in \mathbb{R}^{28\times 28}$ then $z \in \{0,1\}^d$ represents which of the d superpixels should be turned on or off, and letting x_z be the manipulated image. I implemented image trudging and visualization in Python:



While the Quickshift algorithm is sensible for the very simple models Garreau and Mardaoui [2021] consider (indicator functions on particular areas on of the image and a scaled linear pixelwise model), one could argue that it might be incompatible with the CNN architecture of convolutions sliding over an area of the image. With this in mind we instead implemented a grid segmentation of the image to create the superpixels. The downside of this is that the grid might segment the image at unnatural places compared to Quickshift.

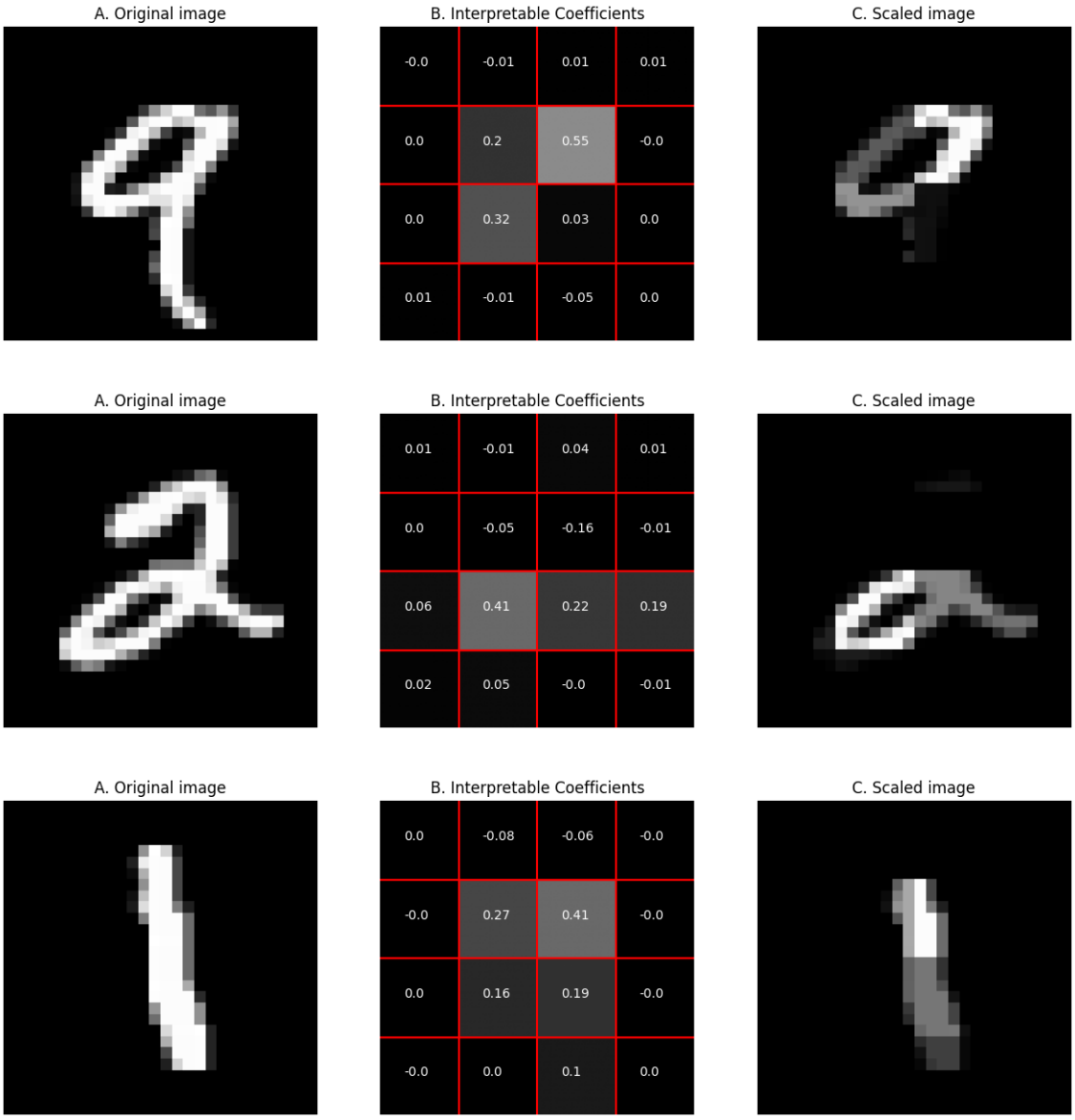
In the standard setting of LIME, turning a superpixel off means replacing each pixel with a *hide color* that is set to the mean value of the superpixel. We instead use black hiding color to remove the superpixel entirely. This gives more distinction between what superpixels contribute most to the prediction, since it yields data that closer resembles the training dataset.[4] Letting the feature matrix Z be sampled with $n = 10^4$ different interpretable components of the image A gives the interpretable coefficients $\beta_{10^4}^{\text{WLS}}$ B, which can be used to scale A as visual interpretation C [5]

[3] Vedaldi and Soatto [2008]

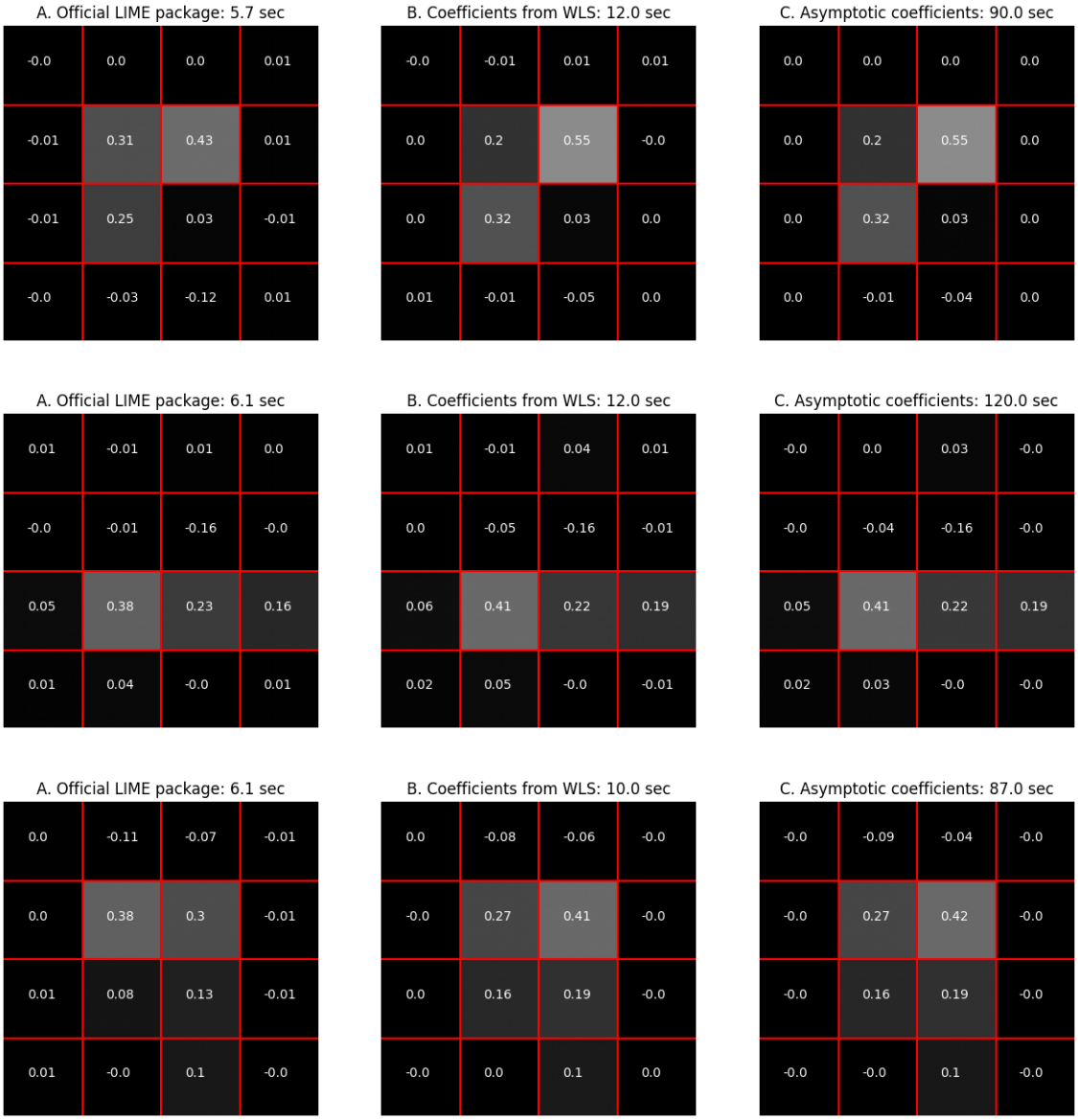
[4] There are no big square areas of grey in the trainingset but lots of big square black areas.

[5] Visualisation code initially from https://github.com/dgarreau/image_lime_theory

3.2 Results



Next, I implemented the formula for the asymptotic coefficients β^f from **Corollary 3** and reformatted the data, model and grid segmentation algorithm to be compatible with the official LIME package Ribeiro et al. [2016] for comparison:



As one can see, B and C are completely the same, and this holds across input. However, there are slight deviances to A. After having thouroughly compared my code to the official LIME library, the difference are due to differences in implementation. [6] There are also differences in computation time. A is twice as fast as B, due to

[6] Noted differences are the fact that A takes image pixels from scale 0-255 whilst I use pixels 0.0 to 1.0.

the fact that the A batches input data, with batch length set to minimize time complexity. C is much slower because β^f needs all combinations [7] However, the overall linear surrogate model is on the one hand simple and on the other able to give local interpretation of the model.

3.3 Example 2: GPT-2

In accordance with terminology in Bengio et al. [2003] consider a language model f that predicts the next word in a sequence. Formally, let $w_i^j = (w_i, w_{i+1}, ..., w_j)$ be a sequence of words from index i to j, where $w_k \in V$ is a word from the vocabulary V. The objective of a language model $\hat{P}$ is then to return a probability distribution over the next word w_{t+1} given a sequence w_1^t, that is $d = t$ and

$$\hat{P} : V^t \rightarrow \mathbb{R}^{|V|} \qquad f(w_1^t) = \hat{P}(w_{t+1} = \text{top label} \mid w_1^t) \quad \text{for} \quad w_{t+1} \in V$$

For example we have the following sentence

w_1^t	w_{t+1}	$\hat{P}(w_{t+1} \mid w_1^t)$
USA should not have invaded	Iraq	0.187
	the	0.086
	Syria	0.052
	Afghanistan	0.051

Lets use the official LIME Text Explainer on this. I imported the pre-trained GPT-2 model from the Huggingface Transformer library Wolf et al. [2019], that serves as a hub for use of large language models. In the official package, for each of the n rows, we let the interpretable components $z_{ij} \in \{0, 1\}^d$ denote whether word w_j, with $j = 1, ..., d$ and $i = 1, ..., n$ should be included in the input sentence to the language model. The official explainer only allows sampled stochastic design. We set $n = 500$. For notation simplification lets replace the notation $\mathbf{1}_{w_j}$ with simply w_j.
For example using official LIME on the above we get

$$\hat{P}(w_{t+1} = \text{Iraq} \mid w_1^t) \approx -0.05 + \underbrace{0.05w_1}_{\text{USA}} + \underbrace{0.02w_2}_{\text{should}} + \underbrace{0.00w_2}_{\text{not}} + \underbrace{0.00w_2}_{\text{have}} + \underbrace{0.07w_2}_{\text{invaded}}$$

This approach has several drawbacks. First of all, if we simply remove the words from the sentence, the resulting sentence might not be grammatically coherent, making it

[7] In this case 2^{16}, which is about 6 times the number of samples in A and B

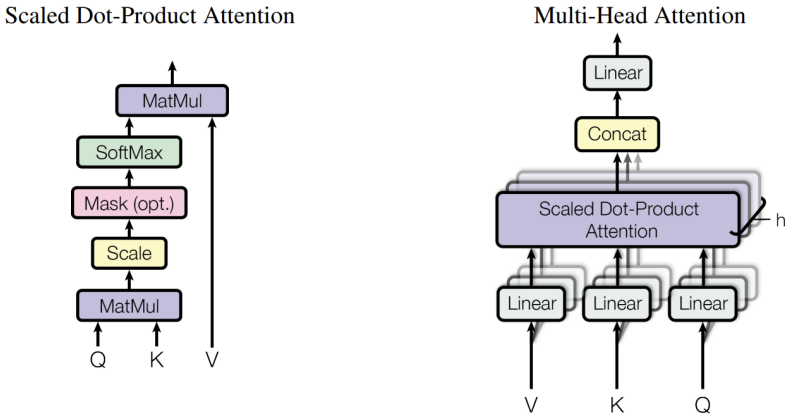
quite unlikely that the language model has stumbled upon this combination of words in its training dataset. It is plausible that this corrupts the output word distribution.

Furthermore, even though the M in LIME stands for "Model-agnostic", there are formatting aspects that the implementation does not take into account. The model architecture pre-processes the words before propagation. If we make some small compromises with agnosticity by assuming knowledge of the preprocessing (but not model parameters), we can provide a more sophisticated implementation of LIME. Lets delve into the architecture.

3.4 Transformers

GPT-2 has around 500M parameters. How does it work? First, the words are tokenized. This means that w_i is associated with an index from 1 to $|V|$ and a unit vector in $\mathbb{R}^{|V|}$ with entry 1 at that index.

Next, they are embedded with a linear transformation from $\mathbb{R}^{|V|} \to \mathbb{R}^d$ with $d << |V|$. Then they are positionally encoded. At this point each word is represented by a d-dimensional vector, and this is where the attention mechanism comes in.



$$\text{Head} = \text{SoftMax}\Big(\frac{Q^T \cdot K}{\sqrt{d_k}} \cdot A\Big) \cdot V$$

Each word vector is then transformed into three new vectors: a Query, Key and Value. The total sentence information is stored in three matrices, Q, K and V. Speciffically column i in Q is the Query of word i etc. for K and V.

Intuitively, the Key K represents information about the word itself, the Query Q represents attention to other words, and the value V_i represents how much value the

combination contributes. Concretely, the word w_i can **attend** to another word w_j by dot product of collumn Q_i and K_j to give combined information about the pair (w_i, w_j). To find information on all pairs we have the matrix product $Q^T K$, which is scaled by $\sqrt{d_k}$ where d_k is the dimension of K to avoid a near-zero gradient during backpropagation.

The important part is the attention mask A. w_i's query Q_i should only attend to the keys of the words up til that word K_1, K_2,...,K_i. Formally this is done by multiplying the result with an attention mask A that is lower-triangular. Then a softmax is applied and the result is multiplied by V. This gives the output for a single head, and many heads are computed in parallel for a single attention block.

In order to find a local interpretation of GPT-2 for small sentences w_1^d, we use a fixed design Z and for each combination $i = 1, ..., 2^d$ let the interpretable component z_{ij} denote whether word w_j should be attended to in the attention matrix. In simple terms we let the attention matrix $A = Z$ (except the first column of Z serving as intercept).

We use weighted linear regression as before with same proximity π_i. We call the implementation of LIME on top of this Attention LIME, results shown in table sectio below.

3.5 Results

Sentence: Donald Trump will never be → President, $P = 16.58\%$

Features	intercept	Donald	Trump	will	never	be	R^2
Official LIME	-0.075	0.022	0.030	0.029	0.030	0.039	0.48
Attention LIME	-0.075	0.037	0.056	0.049	0.056	-0.011	**0.68**

Sentence: Live like there is no → tomorrow, $P = 65.49\%$

Features		Live	like	there	is	no	R^2
Official LIME	-0.022	0.010	0.009	0.008	0.007	0.006	0.023
Attention LIME	-0.274	0.132	0.122	0.126	0.114	0.044	**0.378**

Sentence: UK Parliament has implemented new → legislation, $P = 11.49\%$

Features		UK	Parliament	has	implemented	new	R^2
Official LIME	-0.054	0.016	0.045	-0.001	0.030	0.044	0.60
Attention LIME	-0.056	0.036	0.058	0.017	0.042	0.003	**0.80**

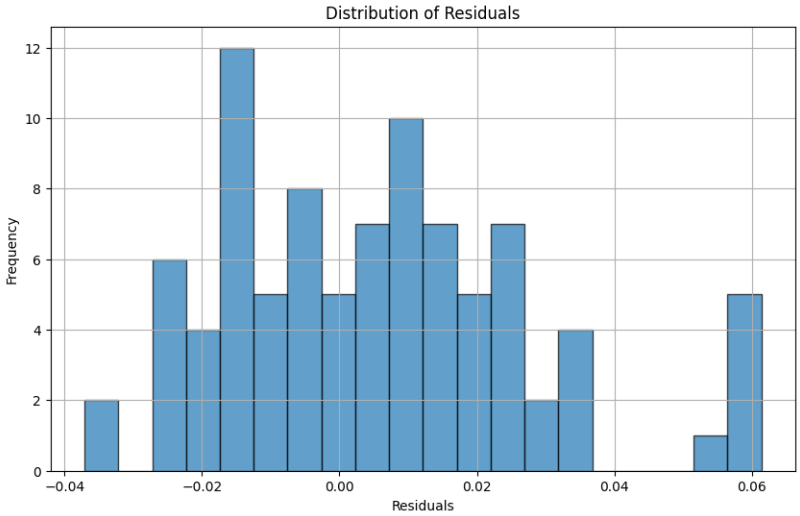
R^2 values are weighted by π_i. As you can see, the coefficients of Official LIME are generally smaller in absolute value than Attention LIME. Generally the order of words by contribution are the same, with relatively small differences between the implementations.

Note that we use a stochastic design with $n = 500$, which might spread out the mutual weighting of the datapoints. On the contrary, had we used a very small fixed design $n = 2^5$ it might have given large importance to single datapoints on the resulting coefficients in spite of the weighting.

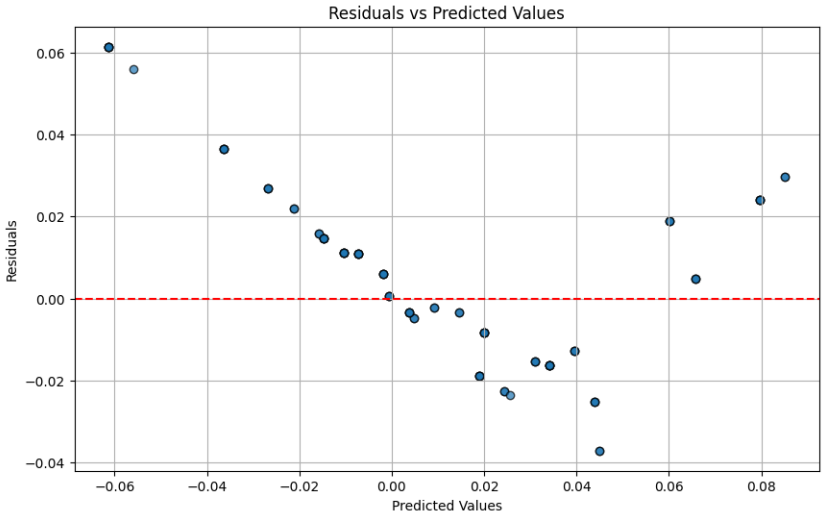
How does the model perform? In general, the coefficients provided can be justified intuitively post-hoc. In third example, it makes sence that the word "Parliament" and has the highest contribution the output "legislation". Similarly it makes sence that words in example 2 performs badly, since this is a general saying with the interaction of all words being necessary.

Overall, Attention LIME works better. This is plausibly due to the reasoning in the former section, but it could also in part be attributed to the fact that GPT-2 sometimes change the tokenization of the input sentence depending on what word comes before/after. The results show that choice of interpretable components does impact LIME performance, but we traded it in exchange for knowledge of the model pre-processing, making a compromise with being model agnostic.

Are the assumptions justified?
With respect to **A1** the R^2 values depend heavily on the instance input x. **A2** is used to simplify the mathematics, but it might be possible to get an even better model and simultaneously reducing complexity. As for the validity of **A3** we can look at the following is a histogram of the residuals of the example 3 official LIME:



Without doing a formal test, it does not seem crazy to assume $\mathbb{E}(u) = \mathbf{0}$. But the assumption of a constant variance does not seem justified, making an implementation of **Theorem 6** redundant:



With linear models, it becomes cumbersome to interperret coefficients for longer sentences. While the results do seem intuitive for post-hoc explanations for small sentences, we can perhaps improve the performance by increasing model complexity.

4 Theory revisited

Given the intuition about the model in the transformer section, lets extend our class of models to take interaction between words into account. With d interpretable components, this will yield $p = \frac{d(d+1)+2}{2}$ coefficients including intercept, making it cumbersome to visually interpret the model. To counter this we can add a complexity penalty to only get coefficients for the most contributing features. In effect, we make the new assumptions

New Assumptions

N1. G is the class of linear models with pairwise interaction:

$$G = \left\{ g : \mathbb{R}^{d+1} \to \mathbb{R} \mid \exists \beta \in \mathbb{R}^p \ : \ g(z) = \beta_{00} + \sum_{i=0}^{d} \sum_{i<j} \beta_{ij} z_i z_j \right\}$$

N2. The complexity penalty is $\Omega(\beta) = \lambda ||\beta||_1$ with hyperparameter $\lambda > 0$

We can represent the new framework in a similar fashion to (3) by letting the new (fixed or stochastic) design $X \in \mathbb{R}^{n \times p}$ have 0-th column serving as intercept, followed d interpretable components and $(d-1) + (d-2) + \cdots + 1$ interaction columns, with first row being $\mathbf{1}^T$ when X is stochastic. We let y and W be defined only in terms of the components $z \in \{0,1\}^d$, exactly as before.

The new LIME ξ can then parametrized by the LASSO (Least Absolute Shrinkage and Selection Operator) estimator.

$$\hat{\beta}_n^{\text{LASSO}} = \arg\min_{\beta \in \mathbb{R}^p} \ (y - X^T\beta)^T W (y - X^T\beta) + \lambda ||\beta||_1 \tag{7}$$

Unfortunately there is no general formula for the expression of the LASSO estimator, coefficients are calculated through algorithms like LARS Efron et al. [2004]. In general the coefficients are not necessarily unique, however that might be possible to show in this case with high probability, since the probability of X having full rank increases with n, implying $X^T W X$ is positive definite, so the objective function in LASSO is strictly convex, implying a unique global solution for $\hat{\beta}$. I will not go into the details here.

4.1 Results

Sentence: UK Parliament has implemented new → legislation, $P = 11.49\%$

Support	intercept	Parliament-implemented	Parliament-new	λ	R^2
Lasso LIME	0.023	0.00225	0	0.1	0.21
Lasso LIME	0.014	0.01081	0.01817	0.07	0.51

Sentence: Live like there is no → tomorrow, $P = 65.49\%$

Support	intercept	Live-there	there-no	λ	R^2
Lasso LIME	0.028	0.00225	0	0.2	0.008

Sentence: Donald Trump will never be → President, $P = 16.6\%$

Support	intercept	Trump-be	will-be	never-be	λ	R^2
Lasso LIME	0.016	0.0005	0	0	0.0085	0.006
Lasso LIME	0.015	0.002	0.001	0.001	0.008	0.05

We set the λ high at first and decrease it little by little to get $||\beta||_0 = 1$ and then $||\beta||_0 = 2$. It was not possible to decrease example 2 without the coefficients spreading out completely. Generally the R^2 values are smaller, but the post-hoc explanations seem very sensible. The most important features according to LIME is not surprising. Looking linguistically at the sentence the result is intuitive post-hoc but comes at a considerable price in R^2.

5 Conclusion

In this paper, I have investigated the theory and use of LIME, showed asymptotic results for the WLS estimator with Bernoulli design matrix. I have applied LIME to an image example and implemented theoretical asymptotic solution, verifying the theoretical expression. Given the understanding of transformer architectures I found a (to my knowledge) new way of using Linear models as surrogate on transformers, which resulted in a higher R^2 value. I also investigated whether interactions would be an improvement. While the post-hoc explanations were sensible, they came at a considerable loss in R^2.

6 Limiatations and future work

It could be interesting to look at making the code more time-efficient, unlocking the possibility of calculating β^* for higher d. The assumptions **A1-A3** are quite restrictive in classes of models, and even though **N1-N2** is an improvement, there are many relationships not taken into account. Having weaker assumptions could be an interesting direction. It would be interesting to find the asymptotic distribution of $\hat{\beta}_n^{\text{WLS}}$ with the right assumptions on u. Second, given the strong assumptions of X, it should not be too difficult to prove uniqueness of $\hat{\beta}_n^{\text{LASSO}}$ with high probability. Adding an assumption similar to **A3** for LASSO should be enough to show that the support $S(\beta) = \{j \in \{1, ..., p\} \mid \beta_j \neq 0\}$ satisfies $S(\hat{\beta}_n) \subseteq S(\beta^*)$ with high probability as $n \to \infty$ as well as an upper bound on $||\hat{\beta}_n - \beta^*||_2$ and the weighted R^2. It could also be interesting to look further into the role of the hyperparameter ν in π_i.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137–1155, 2003.

Colin Cameron. Asymptotic theory for ols. 2013. URL https://cameron.econ.ucdavis.edu/e240a/asymptotic.pdf.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. arXiv preprint arXiv:math/0406456, 2004. URL https://arxiv.org/abs/math/0406456.

Damien Garreau and Dina Mardaoui. What does lime really see in images? ICML, 2021. URL https://arxiv.org/abs/2102.06307.

Damien Garreau and Ulrike von Luxburg. Looking deeper into tabular lime. 2020. URL https://arxiv.org/abs/2008.11092.

Ernst Hansen. Stochastic Processes. 2022.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc.,

2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Carl D. Meyer. Matrix Analysis and Applied Linear Algebra. 2000.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. Conference on Knowledge Discovery and Data Mining (KDD), 2016.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685, 2017.

Stefano Teso and Kristian Kersting. Identifying spurious correlations and correcting them with explanation-based learning. arXiv preprint arXiv:2211.08285, 2019. URL https://ar5iv.org/abs/2211.08285.

Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. European conference on computer vision, pages 705–718, 2008.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. 2019. URL https://arxiv.org/abs/1910.03771.

A Theory details

Lemma 1
If Z^TWZ is positive definite, the solution to (3) is given by

$$\hat{\beta}_n^{\text{WLS}} = (Z^TWZ)^{-1}ZWy \tag{8}$$

Proof. Assume Z^TWZ is positive definite. We want to find the global minimum of the function $f : \mathbb{R}^d \to \mathbb{R}$ defined by

$$f(\beta) = (y - Z^T\beta)^TW(y - Z^T\beta) = \sum_{i=1}^{k} \pi_i(y_i - Z_i^T\beta)^2$$

We set $\nabla f = 0$ and find the Hesse-matrix

$$\begin{aligned}\nabla f &= \nabla\big[(y^T - \beta^T Z)W(y - Z^T\beta)\big] \\ &= \nabla\big[y^T W y - y^T W Z^T \beta - \beta^T Z W y + \beta^T Z W Z^T \beta)\big] \\ &= -2ZWy + 2Z^T W Z\beta \quad \Leftrightarrow \quad Z^T W Z\beta = ZWy\end{aligned}$$

And wee see that the Hessian is $2Z^T WZ$. By assumption it is positive definite, so it is in particular invertible, yielding the unique solution

$$\beta = (Z^T W Z)^{-1} ZWy$$

By the General Second Derivative Test we have that since the Hessian is positive definite, β is a local minima. Since β is unique, it must be a global minimum. $\square$

A.1 Computing expectations

By definition of $\hat{\Sigma}_n$, Z and W

$$\hat{\Sigma}_n = \tfrac{1}{n} Z^T W Z = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^n \pi_i & \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,1} & \cdots & \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,d} \\ \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,1} & \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,1} & \cdots & \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,1} z_{i,d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,d} & \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,1} z_{i,d} & \cdots & \frac{1}{n}\sum_{i=1}^n \pi_i z_{i,d} \end{pmatrix}$$

Where d was the number of interpretable components. Recalling that since Z is Stochastic with first row $\mathbf{1}^T$, and that since $z_{i,j} \in \{0, 1\}$ we have $z_{i,j}^2 = z_{i,j}$.

Lets calculate π_i explicitly:

$$d_{\cos}(\mathbf{1}, z_i) = 1 - \frac{\mathbf{1}^T z_i}{||\mathbf{1}|| \cdot ||z_i||} = 1 - \frac{\sum_{j=1}^d z_{i,j}}{\sqrt{d}\sqrt{\sum_{j=1}^d z_{i,j}^2}} = 1 - \sqrt{\frac{\sum_{j=1}^d z_{i,j}}{d}}$$

Let s denote the number of turned off interpretable components, or the number of $z_i j = 0$ in row i. Then $\sum_{j=1}^d z_{i,j} = d - s$ so

$$d_{\cos}(\mathbf{1}, z_i) = 1 - \sqrt{1 - \tfrac{s}{d}}$$

and we have $\pi_i = \psi(\frac{s}{d})$ for the function

$$\psi(t) := \exp\left(\frac{-(1-\sqrt{1-t})^2}{2\nu^2}\right) \quad t \in [0,1]$$

Recall that $z_{i,j} \sim \mathcal{B}(\frac{1}{2})$ are i.i.d. and the row-sums completely determine π_i, so the π_i's are all binomially distributed over the values $\psi(s/d)$. This means that

$$\mathbb{E}(\pi_i z_{i,j}) = \mathbb{E}(\pi z_1) \text{ and } \mathbb{E}(\pi_i z_{i,j} z_{i,l}) = \mathbb{E}(\pi z_1 z_2)$$

for some $z_1, z_2 \sim \mathcal{B}(\frac{1}{2})$ and π having same distribution as π_i.

Definition 7. Garreau and Mardaoui [2021] Let $\alpha_0 := \mathbb{E}(\pi)$ and $\alpha_p := \mathbb{E}(\pi z_1 z_2 \cdots z_p)$ This definition yields:

$$\Sigma = \begin{pmatrix} \mathbb{E}(\pi) & \mathbb{E}(\pi z_1) & \cdots & \mathbb{E}(\pi z_1) \\ \mathbb{E}(\pi z_1) & \mathbb{E}(\pi z_1) & \cdots & \mathbb{E}(\pi z_1 z_2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{E}(\pi z_1) & \mathbb{E}(\pi z_1 z_2) & \cdots & \mathbb{E}(\pi z_1) \end{pmatrix} = \begin{cases} \alpha_0, & j = k = 0 \\ \alpha_1, & k = 0 \text{ or } j = 0 \text{ or } k = j > 0 \\ \alpha_2, & \text{otherwise} \end{cases}$$

Where j, k is a 0-indexing of Σ. So how do we compute α_p?

Lemma 8. Garreau and Mardaoui [2021] Let $0 \leq p \leq d$, then

$$\alpha_p = \frac{1}{2^d} \sum_{s=0}^{d} \binom{d-p}{s} \psi(s/d)$$

Proof. Recall that π is binomially distributed on the values $\psi(s/d)$. We can formalise this by $\pi = \psi(S/d)$ with $S = d - \sum_{i=1}^{d} z_i$ being the random number of turned off interpretable components among the z_i's for $i = 1, ..., d$. Formally $S \sim \mathcal{B}(d, 1/2)$

$$\mathbb{P}\big(\pi = \psi(s/d)\big) = \mathbb{P}\big(S = s\big) = \frac{1}{2^d}\binom{d}{s}$$

By law of total expectation on S

$$\alpha_p = \mathbb{E}(\pi z_1 \cdots z_p) = \sum_{s=0}^{d} \mathbb{E}(\pi z_1 \cdots z_p \mid S = s)\, \mathbb{P}(S = s) = \frac{1}{2^d} \sum_{s=0}^{d} \binom{d}{s} \mathbb{E}(\pi z_1 \cdots z_p \mid S = s)$$

Each term is only positive if all of $z_1, ..., z_p$ are 1, otherwise the term vanishes.

$$\begin{aligned}\alpha_p &= \frac{1}{2^d}\sum_{s=0}^{d}\binom{d}{s}\mathbb{E}(\pi z_1 \cdots z_p \mid z_1 = 1, ..., z_p = 1, S = s)\,\mathbb{P}(z_1 = 1, ..., z_p = 1 \mid S = s) \\ &= \frac{1}{2^d}\sum_{s=0}^{d}\binom{d}{s}\psi(s/d)\mathbb{P}(z_1 = 1, ..., z_p = 1 \mid S = s)\end{aligned}$$

If the first p interpretable components are turned on, then all the s interpretable components that should be turned off must lie among the other $d - p$ interpretable components. There are $\binom{d-p}{s}$ of such combinations. The number of combinations where s of the d interpretable components are turned off is simply $\binom{d}{s}$

$$\mathbb{P}(z_1 = 1, ..., z_p = 1 \mid S = s) = \frac{\binom{d-p}{s}}{\binom{d}{s}} = \frac{(d-p)!}{s!(d-p-s)!}\cdot\frac{s!(d-s)!}{d!}$$

such that

$$\begin{aligned}\alpha_p &= \frac{1}{2^d}\sum_{s=0}^{d}\frac{d!}{s!(d-s)!}\cdot\frac{(d-p)!}{(d-p-s)!}\cdot\frac{(d-s)!}{d!}\cdot\psi(s/d) \\ &= \frac{1}{2^d}\sum_{s=0}^{d}\binom{d-p}{s}\psi(s/d) \quad \square\end{aligned} \tag{9}$$

□

Note that the sum in fact only has $d - p$ terms, because if $s > d - p$ we have to turn off one of the first p interpretable components.

Lemma 9. Garreau and Mardaoui [2021] For $0 \leq p \leq d$ we have

$$\frac{e^{-\frac{1}{2\nu^2}}}{2^p} \leq \alpha_p \leq \frac{1}{2^p}$$

Proof. Since $\psi(t)$ is defined on $t \in [0, 1]$ and decreasing, we must have $e^{-\frac{1}{2\nu^2}} \leq \psi(t) \leq 1$ by setting $t = 0, 1$. This implies that

$$\frac{1}{2^d}\sum_{s=0}^{d}\binom{d-p}{s}e^{-\frac{1}{2\nu^2}} \leq \alpha_p \leq \frac{1}{2^d}\sum_{s=0}^{d}\binom{d-p}{s}\cdot 1$$

Recall that there are in fact only $d - p$ terms in the sums, and it is a simple combinatorial argument that $\sum_{s=0}^{d-p}\binom{d-p}{s} = 2^{d-p}$ yielding

$$\frac{e^{-\frac{1}{2\nu^2}}}{2^p} \leq \alpha_p \leq \frac{1}{2^p} \quad \square$$

$\square$

Now that we have an explicit formula for the cells of Σ, we are going to show that it is invertible and compute Σ^{-1}.

Lemma 10. Garreau and Mardaoui [2021] For $d \geq 1$, the inverse of Σ is given by

$$\Sigma^{-1} = \frac{1}{c_d} \begin{pmatrix} \sigma_0 & \sigma_1 & \sigma_1 & \cdots & \sigma_1 \\ \sigma_1 & \sigma_2 & \sigma_3 & \cdots & \sigma_3 \\ \sigma_1 & \sigma_3 & \sigma_2 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \sigma_3 \\ \sigma_1 & \sigma_3 & \cdots & \sigma_3 & \sigma_2 \end{pmatrix} \text{ where } \begin{cases} c_d &= (d-1)\alpha_0\alpha_2 - d\alpha_1^2 + \alpha_0\alpha_1 \\ \sigma_0 &= (d-1)\alpha_2 + \alpha_1 \\ \sigma_1 &= -\alpha_1 \\ \sigma_2 &= \frac{(d-2)\alpha_0\alpha_2 - (d-1)\alpha_1^2 + \alpha_0\alpha_1}{\alpha_1 - \alpha_2} \\ \sigma_3 &= \frac{\alpha_1^2 - \alpha_0\alpha_2}{\alpha_1 - \alpha_2} \end{cases}$$

With Σ^{-1} being a $(d+1) \times (d+1)$ dimensional matrix and assuming $c_d \neq 0$ and $\alpha_1 - \alpha_2 \neq 0$.

Proof. Lets assume that there exists $c_d, \sigma_0, ...\sigma_3$ such that Σ^{-1} has the shape above. Recall the structure of Σ:

$$\Sigma = \begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_1 & \cdots & \alpha_1 \\ \alpha_1 & \alpha_1 & \alpha_2 & \cdots & \alpha_2 \\ \alpha_1 & \alpha_2 & \alpha_1 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \alpha_2 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_2 & \alpha_1 \end{pmatrix}$$

Assuming Σ^{-1} has the above shape (without knowing the values σ in terms of α) we must have $\Sigma \cdot \Sigma^{-1} = I$. This can be used to create a linear system of equations in σ and c_d that can be solved in terms of α.
Noticing the simple structures of Σ and Σ^{-1} this yields 5 equations, namely 2 from multiplying the first row of Σ with the two first columns of Σ^{-1} and the second row of Σ combined with the first 3 columns of Σ^{-1}:

$$\begin{aligned} \alpha_0\sigma_0 + d\alpha_1\sigma_1 &= c_d \\ \alpha_0\sigma_1 + \alpha_1\sigma_2 + (d-1)\alpha_1\sigma_3 &= 0 \\ \alpha_1\sigma_0 + \alpha_1\sigma_1 + (d-1)\alpha_1\sigma_3 &= 0 \\ \alpha_1\sigma_1 + \alpha_1\sigma_2 + (d-1)\alpha_2\sigma_3 &= c_d \\ \alpha_1\sigma_1 + \alpha_1\sigma_3 + \alpha_2\sigma_2 + (d-2)\alpha_2\sigma_3 &= 0 \end{aligned}$$

This is a linear system of 5 equations with 5 unknowns, that can be solved one variable at a time or with Gaussian elimination.
Solving the linear system yields the solution given in the statement, and it can be checked by matrix multiplying the resulting Σ^{-1} with Σ. □

□

To show that Σ is invertible, it is enough to show $c_d > 0$ and $\alpha_1 - \alpha_2 > 0$. This is in fact rather difficult, given the expression of α.

Lemma 11 (Four letter identity)**.** Garreau and von Luxburg [2020] Let $A_j, B_j, C_j, D_j \in \mathbb{R}$ for $j = 0, ..., d$. Then

$$\Big(\sum_{j=0}^{d} A_j C_j\Big)\cdot\Big(\sum_{j=0}^{d} B_j D_j\Big) - \Big(\sum_{j=0}^{d} A_j B_j\Big)\cdot\Big(\sum_{j=0}^{d} C_j D_j\Big) = \frac{1}{2}\sum_{j=0}^{d}\sum_{k=0}^{d}(A_j D_k - A_k D_j)(C_j B_k - C_k B_j).$$

Proof.
The proof is simply to multiply out the right hand side and notice symmetry between k and j. We have

$$(A_j D_k - A_k D_j)(C_j B_k - C_k B_j) = A_j C_j B_k D_k - A_j B_j C_k D_k - A_k B_k C_j D_j + A_k C_k B_j D_j$$

The first and fourth term yield the first product, whilst the second and third term yield the second product, cancelling out factor $1/2$. □

□

Lemma 12 (Binomial identities)**.** Garreau and von Luxburg [2020]

$$\sum_{j=0}^{d}\binom{d}{j} = 2^d,\ \sum_{j=0}^{d}\binom{d}{j} j = d2^{d-1},\ \sum_{j=0}^{d}\binom{d}{j} j^2 = d(d+1)2^{d-2},\ \frac{1}{2}\sum_{j,k}^{d}\binom{d}{j}\binom{d}{k}(j-k)^2 = d\cdot 4^{d-1}$$

Proof. Recall the Binomial Theorem

$$(x+y)^d = \sum_{j=0}^{d}\binom{d}{j} x^j y^{d-j}$$

Setting $x = y = 1$ yields (1), differentiating wrt x first then setting $x = y = 1$ yields

(2) and differentiating wrt x twice then setting $x = y = 1$ yields (3). Next, note that

$$\begin{aligned}
\frac{1}{2}\sum_{j,k}^{d}\binom{d}{j}\binom{d}{k}(j-k)^2 &= \sum_{j,k}^{d}\binom{d}{j}\binom{d}{k}k^2 - \sum_{j,k}^{d}\binom{d}{j}\binom{d}{k}jk && \text{(by symmetry)} \\
&= \underbrace{\sum_{j}^{d}\binom{d}{j}}_{1}\underbrace{\sum_{k}^{d}\binom{d}{k}k^2}_{3} - \underbrace{\left(\sum_{j}^{d}\binom{d}{j}j\right)^2}_{2} && \text{(Rearraning sums)} \\
&= 2^d \cdot d(d+1)2^{d-2} - d^2 4^{d-1} = d4^{d-1} \quad \square
\end{aligned}$$

□

Lemma 13 (Σ is invertible)**.** Garreau and Mardaoui [2021] For any $d \geq 1$, $\nu > 0$ and $0 \leq p \leq d$ we have

$$\frac{e^{-\frac{1}{\nu^2}}}{2^{p+1}} \leq \alpha_p - \alpha_{p+1} \leq \frac{1}{2^{p+1}} \quad \text{and} \quad \frac{e^{-\frac{1}{\nu^2}}}{4} \leq c_d \leq \frac{1}{4}$$

Specifically $\alpha_1 - \alpha_2 > 0$ and $c_d > 0$.

Proof. By Pascals formula $\binom{n}{m} + \binom{n}{m-1} = \binom{n+1}{m}$ for $1 \leq m \leq n$ so

$$\alpha_p - \alpha_{p+1} = \frac{1}{2^d}\sum_{s=0}^{d}\left(\binom{d-p}{s} - \binom{d-p-1}{s}\right)\psi(s/d) = \frac{1}{2^d}\sum_{s=0}^{d}\binom{d-p-1}{s-1}\psi(s/d)$$

By using the same argument as in **Lemma 9** we therefore have the first inequality. Next we study c_d. We can rewrite c_d in terms of $\alpha_0 - \alpha_1$ and $\alpha_1 - \alpha_2$

$$\begin{aligned}
c_d &= (d-1)\alpha_0\alpha_2 - d\alpha_1^2 + \alpha_0\alpha_1 \\
&= d\alpha_1(\alpha_0 - \alpha_1) - (d-1)\alpha_0(\alpha_1 - \alpha_2)
\end{aligned}$$

Then we use the above expression for $\alpha_p - \alpha_{p+1}$

$$c_d = \frac{1}{4^d}\left[d\cdot\underbrace{\sum_{s=0}^{d}\binom{d-1}{s}\psi(s/d)}_{\alpha_1}\cdot\underbrace{\sum_{s=0}^{d}\binom{d-1}{s-1}\psi(s/d)}_{\alpha_0-\alpha_1} - (d-1)\cdot\underbrace{\sum_{s=0}^{d}\binom{d}{s}\psi(s/d)}_{\alpha_0}\cdot\underbrace{\sum_{s=0}^{d}\binom{d-2}{s-1}\psi(s/d)}_{\alpha_1-\alpha_2}\right]$$

Since $(m+1)\binom{m}{n} = (n+1)\binom{m+1}{n+1}$

$$c_d = \frac{1}{4^d}\left[\sum_{s=0}^{d}\binom{d-1}{s}\psi(s/d)\cdot\sum_{s=0}^{d}\mathbf{s}\binom{\mathbf{d}}{\mathbf{s}}\psi(s/d) - \sum_{s=0}^{d}\binom{d}{s}\psi(s/d)\cdot\sum_{s=0}^{d}\mathbf{s}\binom{\mathbf{d-1}}{\mathbf{s}}\psi(s/d)\right]$$

and we can use **Lemma 11**

$$A_s := \binom{d-1}{s}\sqrt{\psi(s/d)},\;\; B_s := s\sqrt{\psi(s/d)}, C_s := \sqrt{\psi(s/d)}, D_s := \binom{d}{s}\sqrt{\psi(s/d)}.$$

so we get

$$c_d = \frac{1}{4^d}\cdot\frac{1}{2}\left[\sum_{s,t}^{d}\left(\binom{d-1}{s}\binom{d}{t} - \binom{d-1}{t}\binom{d}{s}\right)(t-s)\psi(s/d)\psi(t/d)\right]$$

Using the fact that

$$\binom{d-1}{s} = \binom{d}{s}\cdot\frac{d-s}{d}$$

we have

$$\binom{d-1}{s}\binom{d}{t} - \binom{d-1}{t}\binom{d}{s} = \binom{d}{s}\binom{d}{t}\cdot\frac{t-s}{d}$$

which gives

$$c_d = \frac{1}{4^d}\cdot\frac{1}{2d}\sum_{s,t}^{d}\binom{d}{s}\binom{d}{t}(t-s)^2\psi(s/d)\psi(t/d)$$

as before we have $e^{-\frac{1}{2\nu^2}} \le \psi(s/d) \le 1$, so

$$\frac{1}{2}\sum_{s,t}^{d}\binom{d}{s}\binom{d}{t}(t-s)^2 = d4^{d-1}$$

By **Lemma 12**.

$$c_d \ge \frac{1}{4^d}\cdot\frac{1}{d}\cdot d4^{d-1}\cdot\left(e^{-\frac{1}{2\nu^2}}\right)^2 = \frac{e^{-\frac{1}{\nu^2}}}{4}$$

And similarly $c_d \le \frac{1}{4}$ $\square$. $\square$

Recall the definition of $\hat{\Gamma}_n = \frac{1}{n}Z^TWy$. Matrix multiplication yields:

$$\hat{\Gamma}_n = \begin{pmatrix} \frac{1}{n}\sum_{i=1}^{n}\pi_i f(x_i) \\ \frac{1}{n}\sum_{i=1}^{n}\pi_i z_{i,1} f(x_i) \\ \vdots \\ \frac{1}{n}\sum_{i=1}^{n}\pi_i z_{i,d} f(x_i) \end{pmatrix} \quad \Rightarrow \quad \Gamma = \begin{pmatrix} \mathbb{E}(\pi f(x)) \\ \mathbb{E}(\pi z_1 f(x)) \\ \vdots \\ \mathbb{E}(\pi z_d f(x)) \end{pmatrix}$$

Where $z_j \in \{0,1\}$ refers to interpretable component j.

A.2 Conditional Upper Bound

In this section we wish to show that if $||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32$

$$||\hat{\beta}_n - \beta_f|| \leq 2||\Sigma^{-1}||_{\text{op}}||\hat{\Gamma}_n - \Gamma|| + 2||\Sigma^{-1}||_{\text{op}}^2 \cdot ||\Gamma|| \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}}$$

First, lets define the **operator** norm.

Definition 14. Meyer [2000] Given the standard Eucledian 2-norm $||\cdot||$ on $\mathbb{R}^n$, the *operator* norm on a matrix $A \in \mathbb{R}^{n\times n}$ is given by

$$||A||_{\text{op}} = \max_{||x||=1} ||Ax||$$

We show that $||\cdot||_{\text{op}}$ is in fact a matrix norm, corresponding to the first four conditions of the following

Proposition 15. Let $A, B \in \mathbb{R}^{n\times n}$, $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$. Then $||\cdot||_{\text{op}}$ satisfies the following conditions:

(i) $||A||_{\text{op}} \geq 0$ and $||A||_{\text{op}} = 0 \quad \Leftrightarrow \quad A = \mathbf{0}$

(ii) $||\alpha A||_{\text{op}} = |\alpha| \cdot ||A||_{\text{op}}$

(iii) $||A + B||_{\text{op}} \leq ||A||_{\text{op}} + ||B||_{\text{op}}$

(iv) $||A \cdot B||_{\text{op}} \leq ||A||_{\text{op}} \cdot ||B||_{\text{op}}$

(v) $||Ax|| \leq ||A||_{\text{op}} \cdot ||x||$

(vi) If A is invertible, $||A^{-1}||_{\text{op}} = \left(\min_{||y||=1} ||Ay||\right)^{-1}$

Proof. Conditions (i)-(iii) are inherited directly from the Eucledian norm $||\cdot||$.

$$||A \cdot \tfrac{x}{||x||}|| \leq ||A||_{\text{op}}$$

is true since $\frac{x}{||x||}$ has norm 1, showing (v). Using this we can show (iv). Let $||x|| = 1$:

$$||ABx|| \leq ||A||_{\text{op}} \cdot ||Bx|| \leq ||A||_{\text{op}} \cdot ||B||_{\text{op}} \cdot 1$$

and applying max over all $||x|| = 1$ yields (iv). Finally, note that:

$$||A||_{\text{op}} = \max_{||x||=1} |Ax|| = \max_{x\neq\mathbf{0}} \frac{||Ax||}{||x||}$$

If A is invertible, the map $x \mapsto Ax$ is bijective on $\mathbb{R}^{n\times n}$ so substitute $x = Ay$:

$$||A^{-1}||_{\text{op}} = \max_{x\neq \mathbf{0}} \frac{||A^{-1}x||}{||x||} = \max_{Ay\neq \mathbf{0}} \frac{||A^{-1}Ay||}{||Ay||} = \max_{y\neq \mathbf{0}} \frac{||y||}{||Ay||} = \max_{||y||=1} \frac{1}{||Ay||} = \Big(\min_{||y||=1} ||Ay||\Big)^{-1}$$

Where we used $Ay = 0 \Leftrightarrow y = 0$. □ □

Lemma 16. Garreau and von Luxburg [2020] Let $H \in \mathbb{R}^{n\times n}$ and I the identity. If $||H||_{\text{op}} \leq -1 + \frac{\sqrt{7}}{2} \approx 0.32$ then $I + H$ is invertible and

$$||(I + H)^{-1}||_{\text{op}} \leq 2$$

Proof. Assume for contradiction that $I + H$ is not invertible. Then there exists $x \in \mathbb{R}^n$ such that $(I + H)x = 0$ with $x \neq \mathbf{0}$. Multiplying through with $1/||x||$ there must be an x_0 with unit norm such that $(I + H)x_0 = 0$ implying $Hx_0 = -x_0$. But then $||Hx_0||_{\text{op}} \geq || - x_0|| = 1$ which contradicts the assumption.

Now that we know $I + H$ is invertible, **Proposition 15 (vi)** gives

$$||(I + H)^{-1}||_{\text{op}} = \Big(\min_{||y||=1} ||(I + H)y||\Big)^{-1} = \Big(\min_{||y||=1} ||(I + H)y||^2\Big)^{-1/2}$$

so to find an upper bound for $||(I + H)^{-1}||_{\text{op}}$ is equivalent to a lower bound on $||(I + H)y||^2$. Let $||y|| = 1$ and consider

$$\begin{aligned}
||(I + H)y||^2 &= y^T(I + H)^T(I + H)y \\
&= 1 + y^T(H^T + H + H^TH)y \\
&\geq 1 - |y^T(H^T + H + H^TH)y| \\
&\geq 1 - ||y|| \cdot ||(H^T + H + H^TH)y|| && \text{Cauchy-Schwarz} \\
&\geq 1 - ||H^T + H + H^TH||_{\text{op}} && \text{def. of } ||\cdot||_{\text{op}} \\
&\geq 1 - 2||H||_{\text{op}} - ||H||^2_{\text{op}} && \textbf{Proposition 15 (iii), (iv)}
\end{aligned}$$

This implies that

$$||(I + H)^{-1}||_{\text{op}} \leq (1 - 2||H||_{\text{op}} - ||H||^2_{\text{op}})^{-1/2}$$

the function $f(x) = (1 - 2x - x^2)^{-1/2}$ is positive and increasing on $[0, -1 + \frac{\sqrt{7}}{2}]$ with $f(-1 + \frac{\sqrt{7}}{2}) = 2$ and the bound holds. □ □

Lemma 17. Garreau and von Luxburg [2020] Let $H \in \mathbb{R}^{n\times n}$ and I the identity. If $||H||_{\text{op}} \leq -1 + \frac{\sqrt{7}}{2} \approx 0.32$ then

$$||(I+H)^{-1} - I||_{\text{op}} \leq 2||H||_{\text{op}}$$

Proof. We write

$$(I+H)^{-1} - I = (I+H)^{-1}(I - (I+H)) = -H(I+H)^{-1}$$

so by taking op norms and **Proposition 15 (iv)**

$$||(I+H)^{-1} - I||_{\text{op}} \leq ||H||_{\text{op}} \cdot ||(I+H)^{-1}||_{\text{op}}$$

and **Lemma 16** yields the result. □ □

Lemma 18. Garreau and von Luxburg [2020] Let $X, H \in \mathbb{R}^{n\times n}$ with X invertible and $Y, H' \in \mathbb{R}^n$. If $||X^{-1}H|| \leq -1 + \frac{7}{2}$ then

$$||(X+H)^{-1}(Y+H') - X^{-1}Y|| \leq 2||X^{-1}||_{\text{op}} \cdot ||H'|| + 2||X^{-1}||^2_{\text{op}}||H||_{\text{op}}||Y||$$

Proof. Consider the left hand side of the inequality. If we add and subtract the mixed term $(X+H)^{-1}Y$ and use the triangle inequality

$$||(X+H)^{-1}(Y+H')-X^{-1}Y|| \leq ||(X+H)^{-1}(Y+H')-(X+H)^{-1}Y||+||(X+H)^{-1}Y-X^{-1}Y||$$

This can be reduced

$$= ||X^{-1}(I+X^{-1}H)^{-1}H'|| + ||X^{-1}(I+X^{-1}H)^{-1} - I)Y||$$

Since H', Y are vectors, we can use **Proposition 15 (v)** to get

$$\leq ||X^{-1}(I+X^{-1}H)^{-1}||_{\text{op}} \cdot ||H'|| + ||X^{-1}(I+X^{-1}H)^{-1})||_{\text{op}} \cdot ||Y||$$

and **Proposition 15 (iv)**

$$\leq ||X^{-1}||_{\text{op}} \cdot ||(I+X^{-1}H)^{-1}||_{\text{op}} \cdot ||H'|| + ||X^{-1}||_{\text{op}} \cdot ||(I+X^{-1}H)^{-1}) - I||_{\text{op}} \cdot ||Y||$$

Since $||X^{-1}H||_{\text{op}} \leq -1 + \frac{\sqrt{7}}{2}$, we have by **Lemma 16** that

$$||(I+X^{-1}H)^{-1})||_{\text{op}} \leq 2$$

and by **Lemma 17** and **Proposition 15 (iv)**

$$||(I+X^{-1}H)^{-1}) - I||_{\text{op}} \leq 2||X^{-1}H||_{\text{op}} \leq 2||X^{-1}||_{\text{op}} \cdot ||H||_{\text{op}}$$

If we enter this into above inequality we get

$$\leq 2||X^{-1}||_{\text{op}} \cdot ||H'||_{\text{op}} + 2||X^{-1}||^2_{\text{op}} \cdot ||H'||_{\text{op}} \cdot ||Y|| \quad \square$$

□

Lets set $X = \Sigma, Y = \Gamma, H = \hat{\Sigma}_n - \Sigma, H' = \hat{\Gamma}_n - \Gamma$. By **Lemma 13** Σ is invertible, so if we assume $||\Sigma(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32$, **Lemma 18** gives us

$$||\hat{\beta}_n - \beta_f|| \leq 2||\Sigma^{-1}||_{\text{op}}||\hat{\Gamma} - \Gamma|| + 2||\Sigma^{-1}||_{\text{op}}^2 \cdot ||\Gamma|| \cdot ||\hat{\Sigma} - \Sigma||_{\text{op}}$$

A.3 Capping $||\Sigma^{-1}||_{\text{op}}$ and $||\Gamma||$

Lets first cap the value of $||\Sigma^{-1}||_{\text{op}}$. If we specified $\nu > 0$ we could simply calculate Σ^{-1} using **Lemma 10**. But we want a general bound. We do this by introducing the **Frobenius** norm.

Definition 19. Given the standard Eucledian 2-norm on $\mathbb{R}^n$, the *Frobenius* norm on a matrix $A \in \mathbb{R}^{n \times n}$ is given by

$$||A||_{\text{F}} = \sqrt{\sum_{i,j} a_{i,j}^2}$$

It is easily shown that $||A||_{\text{F}}$ is a matrix norm (satisfying **Proposition 15 (i)-(iv)**). However this is not even necessary, all we need is the following result.

Lemma 20. Let $A \in \mathbb{R}^{n \times n}$. Then $||A||_{\text{op}} \leq ||A||_{\text{F}}$.

Proof. Let $x \in \mathbb{R}^n$ with $||x|| = 1$. Let a_i be the rows of A and consider $||Ax||^2$:

$$||Ax||^2 = \sum_{i=1}^{n} \langle a_i, x \rangle^2 \leq \sum_{i=1}^{n} ||a_i||^2 \cdot ||x||^2 = \sum_{i,j} a_{ij}^2 = ||A||_{\text{F}}^2$$

Using Cauchy-Schwarz. Since x was arbitrary, we can apply $\max_{||x||=1}$ on both sides and take square roots to get the desired result. □ □

Specifically, $||\Sigma^{-1}||_{\text{op}} \leq ||\Sigma^{-1}||_{\text{F}}$ which only depends on the cells σ and factor c_d. We therefore only need to cap the values of σ.

Lemma 21. Garreau and Mardaoui [2021] We have the following bounds:

$$|c_d| \geq \frac{e^{-\frac{1}{\nu^2}}}{4}, \quad |\sigma_0| \leq \frac{3d}{4}, \quad |\sigma_1| \leq \frac{1}{2}, \quad |\sigma_2| \leq 2e^{\frac{1}{\nu^2}}, \quad |\sigma_3| \leq e^{\frac{1}{\nu^2}}$$

Proof. We prove this simply by combinations of **Lemma 9** and 13. The first inequality is given directly by 13. By definition of σ_0

$$|\sigma_0| = |(d-1)\alpha_2 + \alpha_1| \leq \frac{d-1}{4} + \frac{1}{2} \leq \frac{3d}{4}$$

Because of 9 and $d \geq 2$. Similarly $|\sigma_1| = |-\alpha_1| \leq \frac{1}{2}$. Next lets cap $|\alpha_1^2 - \alpha_0\alpha_2|$. We already know

$$\frac{e^{-\frac{1}{\nu^2}}}{4} \leq \alpha_1^2 \leq \frac{1}{4} \quad \text{and} \quad \frac{e^{-\frac{1}{\nu^2}}}{4} \leq \alpha_0\alpha_2 \leq \frac{1}{4}$$

So the distance between the two is at most the difference between their upper and lower bounds

$$|\alpha_1^2 - \alpha_0\alpha_2| \leq \frac{1 - e^{-\frac{1}{\nu^2}}}{4} \leq \frac{1}{4}$$

Since $0 < e^{-\frac{1}{\nu^2}} < 1$. We can rewrite σ_2

$$|\sigma_2| = \frac{|c_d + \alpha_1^2 - \alpha_0\alpha_2|}{|\alpha_1 - \alpha_2|} \leq \frac{\left(\frac{1}{4} + \frac{1}{4}\right)}{\left(\frac{e^{-\frac{1}{\nu^2}}}{4}\right)} = 2e^{\frac{1}{\nu^2}}$$

and lastly

$$|\sigma_3| = \frac{|\alpha_1^2 - \alpha_0\alpha_2|}{|\alpha_1 - \alpha_2|} \leq \frac{\left(\frac{1}{4}\right)}{\left(\frac{e^{-\frac{1}{\nu^2}}}{4}\right)} = e^{\frac{1}{\nu^2}} \ \square$$

$\square$

Lemma 22. Garreau and Mardaoui [2021] Let $d \geq 2$. Then $||\Sigma^{-1}||_F \leq 8de^{\frac{2}{\nu^2}}$

Proof. By construction of Σ^{-1} there is 1 σ_0, $2d$ σ_1's, d σ_2's and $d^2 - d$ σ_3's due to the Gauss sum formula. By definition of the Frobenius norm and **Lemma 21** we get

$$\begin{aligned}
||\Sigma^{-1}||_F^2 &= \frac{1}{c_d^2}\left(\sigma_0^2 + 2d\sigma_1^2 + d\sigma_2^2 + (d^2 - d)\sigma_3^2\right) \\
&\leq 16e^{\frac{1}{\nu^2}}\left(\frac{9d^2}{16} + \frac{2d}{4} + 4de^{\frac{2}{\nu^2}} + (d^2 - d)e^{\frac{2}{\nu^2}}\right) \\
&\leq 64d^2e^{\frac{3}{\nu^2}}
\end{aligned}$$

Since each term in the parentheses is less than $4d^2e^{\frac{2}{\nu^2}}$ since $d \geq 2$ and $1 < e^{\frac{1}{\nu^2}}$. Taking square roots we imply the result. $\square$ $\square$

Capping $||\Gamma||$ is much easier, since the model f in the classification problem is bounded by a constant $M > 0$, especially in the case where f returns a probability

distribution (with each outcome having propability at most 1). By construction of Γ in section **5.1** we get

$$||\Gamma||^2 = \sum_{i=0}^{d} \mathbb{E}(\pi z_i f(x))^2 \leq \sum_{i=0}^{d} \mathbb{E}(\pi z_i M)^2 \leq (d+1)M^2$$

Since $z_d, \pi \leq 1$. This means that $||\Gamma|| \leq M\sqrt{d+1}$.

A.4 Convergence in probability of $\hat{\Sigma}$ and $\hat{\Gamma}$

In this section we show that given $\varepsilon > 0$:

$$\mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{op}} > \varepsilon) \to 0 \quad \text{and} \quad \mathbb{P}(||\hat{\Gamma}_n - \Gamma|| > \varepsilon) \to 0 \quad \text{as} \quad n \to \infty.$$

We do this by Hoeffding's Inequality that depends on the following lemma.

Lemma 23 (Hoeffding's)**.** Let $s > 0$ and assume $a \leq X \leq b$ almost surely. Then

$$\mathbb{E}(e^{s(X-\mathbb{E}(X))}) \leq \exp\Big(\frac{s^2(b-a)^2}{8}\Big)$$

Proof. Without loss of generality, replace $X - \mathbb{E}(X)$ with X that has $\mathbb{E}(X) = 0$. We notice that $x \mapsto e^{sx}$ is convex, which means that for any $a \leq x \leq b$ then

$$e^{sx} \leq \frac{b-x}{b-a}e^{sa} + \frac{x-a}{b-a}e^{sb}$$

In particular

$$e^{sX} \leq \frac{b-X}{b-a}e^{sa} + \frac{X-a}{b-a}e^{sb}$$

almost surely. Taking expectations

$$\mathbb{E}(e^{sX}) \leq \frac{b-\mathbb{E}(X)}{b-a}e^{sa} + \frac{\mathbb{E}(X)-a}{b-a}e^{sb} = \frac{b}{b-a}e^{sa} - \frac{a}{b-a}e^{sb}$$

Lets rewrite the right hand expression as a function $e^{L(h)}$ of $h = s(b-a)$. We do this by taking ln on the right

$$\begin{aligned}
\ln\Big(\frac{b}{b-a}e^{sa} - \frac{a}{b-a}e^{sb}\Big) &= \ln\Big(e^{sa}\Big(\frac{b}{b-a} - \frac{a}{b-a}e^{s(b-a)}\Big)\Big) \\
&= sa + \ln\Big(1 + \frac{a - ae^{s(b-a)}}{b-a}\Big) \\
&= \frac{s(b-a)a}{b-a} + \ln\Big(1 + \frac{a - ae^{s(b-a)}}{b-a}\Big) \\
L(h) :&= \frac{ha}{b-a} + \ln\Big(1 + \frac{a - ae^{h}}{b-a}\Big)
\end{aligned}$$

We then have $\mathbb{E}(e^{sX}) \leq e^{L(s(b-a))}$. Next the idea is to use **Taylor's Theorem** on $L(h)$ and cap the second order term. We compute

$$L'(h) = \frac{a}{b-a} + \frac{\frac{-ae^h}{b-a}}{1+\frac{a-ae^h}{b-a}} = \frac{a}{b-a} - \frac{ae^h}{b-ae^h} \quad \text{and} \quad L''(h) = \frac{-abe^h}{(b-ae^h)^2}$$

Notice that $-\frac{xy}{(x-y)^2} \leq \frac{1}{4} \Leftrightarrow (x+y)^2 \geq 0$ for any $x, y \in \mathbb{R}$. Setting $x = b$ and $y = ae^h$ we get $L''(h) \leq \frac{1}{4}$ for all $h \in \mathbb{R}$. We can now use **Taylor's Theorem** around 0 to see that there must be $0 < \theta < 1$ such that

$$L(h) = L(0) + hL'(0) + \frac{1}{2}h^2 L''(h\theta) \leq \frac{1}{8}h^2$$

Implying $\mathbb{E}(e^{sX}) \leq e^{\frac{s^2(b-a)^2}{8}}$. □ □

Lemma 24 (Hoeffding's Inequality)**.** Let $X_1, ..., X_n$ be i.i.d satisfying $a \leq X_i \leq b$ for $i = 1, ..., n$ and consider $S_n := X_1 + \cdots + X_n$. Then

$$\mathbb{P}(|S_n - \mathbb{E}(S_n)| \geq t) \leq 2\exp\Big(-\frac{2t^2}{n(b-a)^2}\Big)$$

Proof. Let $s > 0$, then by the Markov Inequality

$$\begin{aligned}
\mathbb{P}(S_n - \mathbb{E}(S_n) \geq t) &= \mathbb{P}(\exp(s(S_n - \mathbb{E}(S_n))) \geq \exp(st)) \\
&\leq \frac{E(\exp(s(S_n - \mathbb{E}(S_n))))}{\exp(st)} \\
&= \frac{\mathbb{E}\big(\prod_{i=1}^n \exp(s(X_i - \mathbb{E}(X_i)))\big)}{\exp(st)} \\
&= \frac{\prod_{i=1}^n \mathbb{E}(\exp(s(X_i - \mathbb{E}(X_i))))}{\exp(st)}
\end{aligned}$$

using indepence of X_i. By **Lemma 23** we get

$$\mathbb{P}(S_n - \mathbb{E}(S_n) \geq t) \leq \frac{\prod_{i=1}^n \exp(\frac{s^2(b-a)^2}{8})}{\exp(st)} = \exp(-st + \tfrac{1}{8}s^2 n(b-a)^2)$$

this is true for any $s > 0$ so we can find the minimum over s. We attain a minimum at

$$s = \frac{4t}{n(b-a)^2} \quad \text{getting} \quad \mathbb{P}(S_n - \mathbb{E}(S_n) \geq t) \leq \exp\Big(-\frac{2t^2}{n(b-a)^2}\Big)$$

In order to make it a bound on absolute value consider

$$\mathbb{P}(S_n - \mathbb{E}(S_n) \leq -t) = \mathbb{P}((-S_n) - \mathbb{E}(-S_n) \geq t)$$

We have $-b \leq -X_i \leq -a$ so by symmetry the result also holds for $-X_i$ and $-S_n$

$$\mathbb{P}(S_n - \mathbb{E}(S_n) \leq -t) \leq \exp\Big(-\frac{2(-t)^2}{n(-a-(-b))^2}\Big) = \exp\Big(-\frac{2t^2}{n(b-a)^2}\Big)$$

Together we get the desired result. □

□

The method here is a bit easier than Garreau and von Luxburg [2020] using Matrix Hoeffding.

Lemma 25. Let $\varepsilon > 0$. Then

$$\mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{op}} > \varepsilon) \leq 2(d+1)^2 \exp\Big(\frac{-2n\varepsilon^2}{(d+1)^2}\Big)$$

Proof. We use the Frobenius norm as before. Let $\varepsilon > 0$ then

$$\mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{op}} > \varepsilon) \leq \mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{F}} > \varepsilon) = \mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{F}}^2 > \varepsilon^2)$$

By construction $(\hat{\Sigma}_n)_{ij} = \frac{1}{n}\sum_{k=1}^n \pi_k z_{ki} z_{kj}$ and $\Sigma_{ij} = \mathbb{E}((\hat{\Sigma}_n)_{ij})$, so multiplying n^2.

$$= \mathbb{P}\Big(\sum_{i=0}^{d}\sum_{j=0}^{d}\Big(\sum_{k=1}^{n}\pi_k z_{ki} z_{kj} - n\Sigma_{ij}\Big)^2 > n^2\varepsilon^2\Big)$$

By a union bound argument, this probability must be less than the probability that each term is greater than $\frac{n^2\varepsilon^2}{(d+1)^2}$.

$$\leq (d+1)^2\mathbb{P}\Big(\Big(\sum_{k=1}^{n}\pi_k z_{k1} z_{k2} - n\Sigma_{ij}\Big)^2 > \frac{n^2\varepsilon^2}{(d+1)^2}\Big) = (d+1)^2\mathbb{P}\Big(\Big|\sum_{k=1}^{n}\pi_k z_{k1} z_{k2} - n\Sigma_{ij}\Big| > \frac{n\varepsilon}{d+1}\Big)$$

Since each z_{ki} is Bernoulli distributed. Here we can use **Lemma 24** with $X_k = \pi_k z_{k1} z_{k2}$ having $0 \leq X_k \leq 1$ almost surely.

$$\mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{op}} > \varepsilon) \leq 2(d+1)^2 \exp\Big(\frac{-2n^2\varepsilon^2}{n(d+1)^2(1-0)^2}\Big) = 2(d+1)^2 \exp\Big(\frac{-2n\varepsilon^2}{(d+1)^2}\Big) \quad \square$$

□

Lemma 26. Let $\varepsilon > 0$. Assume $|f| \leq M$ for some $M > 0$. Then

$$\mathbb{P}(||\hat{\Gamma}_n - \Gamma|| > \varepsilon) \leq 2(d+1) \cdot \exp\left(\frac{-n\varepsilon^2}{2M^2(d+1)}\right)$$

.

Proof. As before we use a union bound

$$\mathbb{P}(||\hat{\Gamma}_n - \Gamma|| > \varepsilon) \leq (d+1) \cdot \mathbb{P}\Big(\Big(\frac{1}{n}\sum_{i=1}^{n} \pi_i z_{i1} f(x_i) - \mathbb{E}(\pi z_1 f(x))\Big)^2 > \frac{\varepsilon^2}{d+1}\Big)$$

We have $-M < \pi_i z_{i1} f(x_i) < M$ so **Lemma 24** gives

$$\mathbb{P}(||\hat{\Gamma}_n - \Gamma|| > \varepsilon) \leq 2(d+1) \cdot \exp\left(\frac{-2n^2\varepsilon^2}{4M^2 n(d+1)}\right) \quad \square$$

□

A.5 Putting it together

Theorem 2 Garreau and Mardaoui [2021]

Assume $f : \mathbb{R}^m \to \mathbb{R}$ is bounded by $M > 0$. Then $\hat{\beta}_n \xrightarrow{P} \beta^f$.

Specifically, given $\varepsilon > 0$ and $\eta \in (0,1)$ we have

$$\mathbb{P}(||\hat{\beta}_n - \beta^f|| \geq \varepsilon) \leq \eta$$

whenever

$$n \geq \max\left(\frac{M^2}{\varepsilon^2}, 1\right) \cdot (d+1)^2 d^4 2^{11} e^{\frac{8}{\nu^2}} \cdot \log\left(\frac{4(d+1)^2}{\eta}\right)$$

Proof. Let $\varepsilon > 0$ and $\eta \in (0,1)$. Set $\beta := \Sigma^{-1} \cdot \Gamma$ which exists by **Lemma 13**. According to **Lemma 18** we know that if $||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32$ then

$$||\hat{\beta}_n - \beta|| \leq 2||\Sigma^{-1}||_{\text{op}}||\hat{\Gamma}_n - \Gamma|| + 2||\Sigma^{-1}||_{\text{op}}^2 \cdot ||\Gamma|| \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}}$$

By **Proposition 15(iv)**, **Lemma 20**+**22**

$$||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq ||\Sigma^{-1}||_{\text{op}} \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}} \leq 8de^{\frac{2}{\nu^2}} \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}}$$

We need $||\hat{\Sigma}_n - \Sigma||_{\text{op}}$ to be less than $(25de^{\frac{2}{\nu^2}})^{-1}$ to ensure $||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32$

$$\mathbb{P}(||\hat{\Sigma}_n - \Sigma||_{\text{op}} < (25de^{\frac{2}{\nu^2}})^{-1}) \geq 1 - 2(d+1)^2 \exp\Big(\frac{-2n(25de^{\frac{2}{\nu^2}})^{-2}}{(d+1)^2}\Big)$$

Thus if we set $n_1 := \lceil \log\big(\frac{4(d+1)^2}{\eta}\big)(d+1)^2 d^2 5^4 e^{\frac{4}{\nu^2}} \rceil$ we get

$$\mathbb{P}(||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32) \geq 1 - \frac{\eta}{2}$$

for $n \geq n_1$. So on this event we have the upper bound for $||\hat{\beta}_n - \beta||$. We next find n such that the first term is less than $\frac{\varepsilon}{2}$, again by use of **Lemma 20+22**

$$2||\Sigma^{-1}||_{\text{op}}||\hat{\Gamma}_n - \Gamma|| \leq 16de^{\frac{2}{\nu^2}} \cdot ||\hat{\Gamma}_n - \Gamma||$$

By **Lemma 26**

$$\mathbb{P}\Big(||\hat{\Gamma}_n - \Gamma|| < \frac{\varepsilon}{2} \cdot \frac{1}{16de^{\frac{2}{\nu^2}}}\Big) \geq 1 - 2(d+1) \cdot \exp\Big(\frac{-2n\varepsilon^2}{(d+1)^2 \cdot 2^{10} d^2 e^{\frac{4}{\nu^2}}}\Big) \geq 1 - \frac{\eta}{2}$$

for $n \geq n_2 := \lceil \log\big(\frac{4(d+1)}{\eta}\big)(d+1)^2 \cdot 2^9 d^2 e^{\frac{4}{\nu^2}} \cdot \frac{1}{\varepsilon^2} \rceil$ And similarly we see for the second term that by use of **Lemma 20+22**, and since $||\Gamma|| \leq M\sqrt{d+1}$ we have

$$2||\Sigma^{-1}||_{\text{op}}^2||\Gamma|| \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}} \leq 2^5 d^2 e^{\frac{4}{\nu^2}} \cdot M\sqrt{d+1} \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}}$$

By **Lemma 25** we get

$$\mathbb{P}\Big(||\hat{\Sigma}_n - \Sigma||_{\text{op}} < \frac{\varepsilon}{2} \cdot \frac{1}{2^5 d^2 e^{\frac{4}{\nu^2}} \cdot M\sqrt{d+1}}\Big) \geq 1 - 2(d+1)^2 \exp\Big(\frac{-n\varepsilon^2}{2^{11} d^4 e^{\frac{8}{\nu^2}} \cdot M^2 \cdot (d+1)}\Big)$$

$$\geq 1 - \frac{\eta}{2}$$

for $n \geq n_3 := \lceil \log\big(\frac{4(d+1)^2}{\eta}\big)(d+1) \cdot 2^{11} d^4 e^{\frac{8}{\nu^2}} M^2 \cdot \frac{1}{\varepsilon^2} \rceil$.

Notice that both n_2 and n_3 are less than $n_4 := \lceil \log\big(\frac{4(d+1)^2}{\eta}\big)(d+1)^2 \cdot 2^{11} d^4 e^{\frac{8}{\nu^2}} M^2 \cdot \frac{1}{\varepsilon^2} \rceil$

So for $n \geq n_4$ we have

$$\mathbb{P}\Big(2||\Sigma^{-1}||_{\text{op}}||\hat{\Gamma}_n - \Gamma|| + 2||\Sigma^{-1}||_{\text{op}}^2 \cdot ||\Gamma|| \cdot ||\hat{\Sigma}_n - \Sigma||_{\text{op}} \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2}\Big) \geq 1 - \frac{\eta}{2}$$

and for $n \geq n_1$

$$\mathbb{P}(||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32) \geq 1 - \frac{\eta}{2}$$

Lets call the two events A and B. We have $A \cap B \subset (||\hat{\beta}_n - \beta|| < \varepsilon)$ so if $n \geq \max\{n_1, n_4\}$:

$$\mathbb{P}(||\hat{\beta}_n - \beta|| \geq \varepsilon) \leq \mathbb{P}(A^c) + \mathbb{P}(B^c) \leq \frac{\eta}{2} + \frac{\eta}{2} = \eta. \quad \square$$

$\square$

The following 3 results are entirely my own.

Corollary 4 β^f is the WLS estimator of the fixed version of Z.

Proof. We have the expression for Σ:

$$\Sigma_{jk} = \begin{cases} \alpha_0, & j = k = 0 \\ \alpha_1, & k = 0 \text{ or } j = 0 \text{ or } k = j > 0 \\ \alpha_2, & \text{otherwise} \end{cases}$$

With α given by **Definition 7**. So lets show that the fixed version denoted $\bar{\Sigma}$ is equal to Σ in these cases. First, if $j = k = 0$ then by definition

$$\bar{\Sigma}_{00} = \frac{1}{2^d} \sum_{i=1}^{2^d} \pi_i = \frac{1}{2^d} \sum_{s=0}^{d} \binom{d}{s} \psi(s/d) = \alpha_0$$

Since we run through all combinations of z_i, there will be $\binom{d}{s}$ counts of value $\psi(s/d)$. Similarly if $j = 0$ or $k = 0$ or $k = j$ then by definition

$$\bar{\Sigma}_{jk} = \frac{1}{2^d} \sum_{i=1}^{2^d} \pi_i z_{ij} = \frac{1}{2^d} \sum_{s=0}^{d} \binom{d-1}{s} \psi(s/d) = \alpha_1$$

as before since the terms will be 0 unless $z_{ij} = 1$, and there will be $\binom{d-1}{s}$ combinations of these. The argument is the same for the last case. This means that $\bar{\Sigma} = \Sigma$, and the same argument will show that $\bar{\Gamma} = \Gamma$, implying $\bar{\beta} = \beta^f$ $\square$

Lemma 27. $\hat{\Sigma}_n^{-1} \xrightarrow{P} \Sigma^{-1}$ with respect to $||\cdot||_{\text{op}}$

Proof. Let $\varepsilon > 0$ and $\eta \in (0, 1)$. Rewrite

$$\begin{aligned} ||\hat{\Sigma}_n^{-1} - \Sigma^{-1}||_{\text{op}} &= ||(\Sigma + (\hat{\Sigma}_n - \Sigma))^{-1} - \Sigma^{-1}||_{\text{op}} \\ &= ||\Sigma^{-1}((I + \Sigma^{-1}(\hat{\Sigma}_n - \Sigma))^{-1} - I)||_{\text{op}} \\ &\leq ||\Sigma^{-1}||_{\text{op}} \cdot ||(I + \Sigma^{-1}(\hat{\Sigma}_n - \Sigma))^{-1} - I||_{\text{op}} \end{aligned}$$

by **Lemma 20+22** $||\Sigma^{-1}||_{\text{op}} \leq ||\Sigma^{-1}||_F \leq 8de^{\frac{2}{\nu 2}}$. Next, if we assume that

$$||\Sigma^{-1}(\hat{\Sigma}_n - \Sigma)||_{\text{op}} \leq 0.32$$

Then **Lemma 17** implies that

$$||(I + \Sigma^{-1}(\hat{\Sigma}_n - \Sigma))^{-1} - I||_{\text{op}} \leq 2||\hat{\Sigma}_n - \Sigma||$$

Setting $n \geq n_1$ from the proof of **Theorem 2** ensures that the condition is satisfied

$$\mathbb{P}(||(I + \Sigma^{-1}(\hat{\Sigma}_n - \Sigma))^{-1} - I)||_{\text{op}} \leq 2||\hat{\Sigma}_n - \Sigma||) > 1 - \frac{\eta}{2}$$

Now by **Lemma 25** we can find n_2 such that for all $n \geq n_2$

$$\mathbb{P}\Big(||\hat{\Sigma}_n - \Sigma||_{\text{op}} \leq \frac{\varepsilon}{2||\Sigma^{-1}||_{\text{op}}}\Big) > 1 - \frac{\eta}{2}$$

Notice that on the intersection of these events, we have

$$\begin{aligned} ||\hat{\Sigma}_n^{-1} - \Sigma^{-1}||_{\text{op}} &\leq ||\Sigma^{-1}||_{\text{op}} \cdot ||(I + \Sigma^{-1}(\hat{\Sigma}_n - \Sigma))^{-1} - I||_{\text{op}} \\ &\leq 2||\Sigma^{-1}||_{\text{op}}||\hat{\Sigma}_n - \Sigma|| \leq \varepsilon \end{aligned}$$

so setting $n \geq \max\{n_1, n_2\}$ a union bound argument yields

$$\mathbb{P}(||\hat{\Sigma}_n^{-1} - \Sigma^{-1}||_{\text{op}} > \varepsilon) \leq \frac{\eta}{2} + \frac{\eta}{2} = \eta$$

□

Before we show the asymptotic theorems, we can define similar to **Definition 7**

Definition 28. Let $\gamma_0 = \mathbb{E}(\pi^2)$ and for $1 \leq p \leq d$ define $\gamma_p = \mathbb{E}(\pi^2 z_1 \cdots z_p)$

.

Lemma 29. Let $0 \leq p \leq d$. Then for any $\nu > 0$ we have

$$\gamma_p = \frac{1}{2^d} \sum_{s=0}^{d} \binom{d-p}{s} \psi(s/d)^2$$

Proof. Exactly the same as for **Lemma 8**. We get

$$\begin{aligned}\gamma_p &= \mathbb{E}(\pi^2 z_1 \cdots z_p) \\ &= \frac{1}{2^d}\sum_{s=0}^{d}\binom{d}{s}\mathbb{E}(\pi^2 z_1 \cdots z_p \mid S = s) \\ &= \frac{1}{2^d}\sum_{s=0}^{d}\binom{d}{s}\psi(s/d)^2\mathbb{P}(z_1 = 1, ..., z_p = 1 \mid S = s) \\ &= \frac{1}{2^d}\sum_{s=0}^{d}\binom{d-p}{s}\psi(s/d)^2\end{aligned}$$

□

Theorem 5. $\hat{\beta}_n \xrightarrow{P} \beta^*$ and consequently $\beta^f = \beta^*$. So $\hat{\beta}_n$ is **weakly consistent**.

Proof. Let $\varepsilon > 0$ and $\eta \in (0, 1)$. By **Lemma 1** and **Assumption A3** we have

$$\hat{\beta}_n = (Z^T W Z)^{-1} Z^T W y = \beta^* + (Z^T W Z)^{-1} Z^T W u \tag{10}$$

Lets show the difference in β vanishes in probability. Recalling that $\hat{\Sigma}_n = \frac{1}{n} Z^T W Z$

$$||\hat{\beta}_n - \beta^*|| \leq ||\hat{\Sigma}_n^{-1}||_{\text{op}} \cdot ||\frac{1}{n} Z^T W u||$$

By **Lemma 27** we have $||\hat{\Sigma}_n^{-1}||_{\text{op}} \xrightarrow{P} ||\Sigma^{-1}||_{\text{op}}$ and the second factor can be written

$$\tfrac{1}{n} Z^T W u = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} Z_i^T \pi_i u_i$$

where Z_i is the i'th row of Z. Set $X_i := \pi_i u_i Z_i$. Then X_i is i.i.d and we have

$$\mathbb{E}(X_i) = \mathbb{E}(u_i \mid Z_i) \cdot \mathbb{E}(\pi_i \cdot Z_i) = \mathbf{0} \tag{11}$$

$$\mathbb{V}(X_i) = \mathbb{E}(Z_i Z_i^T \pi_i^2 u_i^2) = \underbrace{\mathbb{E}(u_i^2 \mid Z_i)}_{=\sigma^2} \cdot \mathbb{E}(\pi_i^2 Z_i Z_i^T) < \infty \tag{12}$$

using $\pi_i \in \sigma(Z_i)$ as well as **A3**. By **multivariate WLLN** Hansen [2022] (**Theorem 4.25**) we have $||\frac{1}{n} Z^T W u|| \xrightarrow{P} 0$.

$$||\hat{\beta}_n - \beta^*|| \leq ||\hat{\Sigma}_n^{-1}||_{\text{op}} \cdot ||\frac{1}{n} Z^T W u|| \xrightarrow{P} ||\Sigma^{-1}||_{\text{op}} \cdot 0 = 0$$

By uniqueness of P-limits we have $\beta^* = \beta^f$.

□

Theorem 6 $\hat{\beta}_n \overset{\text{as}}{\sim} \mathcal{N}\big(\beta^*, \frac{\sigma^2}{n}\Sigma^{-1}\Omega\Sigma^{-1}\big)$ for a constant covariance matrix $\Omega \in \mathbb{R}^{(d+1)\times(d+1)}$.

Proof. Using (10) we have

$$\sqrt{n}(\hat{\beta}_n - \beta^*) = \hat{\Sigma}_n^{-1} \cdot \tfrac{1}{\sqrt{n}} ZWu$$

as before $\hat{\Sigma}_n^{-1} \xrightarrow{P} \Sigma^{-1}$ wrt. $||\cdot||_{\text{op}}$. Splitting up the second factor into sums of X_i and using (11) and (12), we can use the **multivariate Laplace CLT** Hansen [2022] (**Theorem 7.7**) to get

$$\frac{1}{n} ZWu \overset{\text{as}}{\sim} \mathcal{N}\big(\mathbf{0}, \frac{\sigma^2}{n}\Omega\big)$$

with

$$\Omega_{jk} = \begin{cases} \gamma_0, & j = k = 0 \\ \gamma_1, & k = 0 \text{ or } j = 0 \text{ or } k = j > 0 \\ \gamma_2, & \text{otherwise} \end{cases}$$

Limit Normal Product Rule Cameron [2013] now implies that $\hat{\beta}_n \overset{\text{as}}{\sim} \mathcal{N}\big(\beta^*, \frac{\sigma^2}{n}\Sigma^{-1}\Omega\Sigma^{-1}\big)$ □

B LIME Image Explainer

B.1 Imports

```
# pip install LIME torch
from LIME.LIME_image import LIMEImageExplainer
from sklearn.linear_model import LinearRegression

import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, TensorDataset

import tensorflow as tf
from skimage.segmentation import mark_boundaries
```

```
15 from skimage.color import gray2rgb, rgb2gray
16
17 import matplotlib
18 import matplotlib.pyplot as plt
19 import scipy.stats as stats
```

B.2 Convolutional Neural Net

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(
            in_channels=1,
            out_channels=32,
            kernel_size=4,
            stride=2)
        self.conv2 = nn.Conv2d(
            in_channels=32,
            out_channels=64,
            kernel_size=4,
            stride=2)
        self.relu1 = nn.ReLU()
        self.relu2 = nn.ReLU()
        self.relu3 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.25)
        self.dropout2 = nn.Dropout(0.5)
        self.softmax = nn.Softmax(dim=1)
        self.layer3 = nn.Linear(64*5*5, 128)
        self.layer4 = nn.Linear(128, 10)
    def forward(self, x):
        x = x.view(x.size(0), 1, 28, 28)
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.conv2(x)
        x = self.relu2(x)
        x = self.dropout1(x)
        x = x.view(x.size(0), -1)
        x = self.layer3(x)
        x = self.relu3(x)
        x = self.dropout2(x)
        x = self.layer4(x)
        x = self.softmax(x)
        return x
```

B.3 Training and Evaluation

```
#Download MNIST train and test dataset
mnist_train = datasets.MNIST(root='./data', download=True, train=True,
    transform=transforms.Compose([transforms.ToTensor()]))

mnist_testset = datasets.MNIST(root='./data', download=True, train=False,
    transform=transforms.Compose([transforms.ToTensor()]))

train_loader = torch.utils.data.DataLoader(
                    mnist_train, batch_size=256, shuffle=True)
test_loader = torch.utils.data.DataLoader(
                    mnist_testset, batch_size=1, shuffle=True)
#Init net, optimizer, criterion
net = Net()
optimizer = torch.optim.Adam(net.parameters(), 0.001)
criterion = nn.CrossEntropyLoss()
#Train loop on training dataset
for j in range(2):
  for batch, (inputs, labels) in enumerate(train_loader):
    outputs = net(inputs)
    loss = criterion(outputs, labels)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    if(batch % 200 == 0):
      print(f"Epoch: {j} at loss: {loss}")

#Evaluate on testset
def evaluate(net, mnist_testset):
  net.eval()
  error = 0
  for i, (input, label) in enumerate(test_loader):
    if torch.argmax(net(input)).item() != label.item():
      error += 1
  return error/len(test_loader)
print(f"Errorrate: {evaluate(net, mnist_testset)}")
```

B.4 LIME Image Linear

```
#Compute Weighted Linear Regression (LIME with linear regressor
def hat_beta_n(x, net, d, n=10000):

  #get original prediction
  x_out = net(x.view(1,1,28,28))
  x_pred = toValue(x_out)

  #Create an z: (n,d) matrix of bernoulli variables
  z = stats.bernoulli.rvs(0.5, size=(n,d), random_state=2)
  z[0] = np.ones(d)

  #initialise vectors y and pi_i
  y = []
  pii = []

  for i in range(n):
    #trudge original x and forward pass
    xi = trudge_input(x, segments, z[i])
    yi = net(xi.view(1,1,28,28))[0][x_pred].item()

    #Append to vectors pi_i and y
    pii.append(compute_psi(1-sum(z[i].astype(float))/len(z[i])))
    y.append(yi)

  y = np.array(y).reshape(n,1)
  W = np.diag(pii)
  Z = np.concatenate((np.ones(n).reshape(n,1), np.array(z.astype(float))), axis=1)

  hat_beta = np.matmul(np.matmul(np.matmul(np.linalg.inv(np.matmul(np.matmul(Z.T, W), Z)), Z.T), W)
  return hat_beta
```

B.5 Asymptotic coefficients β^f

```
def expected_beta(x, net, d):
  #get original prediction
  x_out = net(x.view(1,1,28,28))
  pred_x = toValue(x_out)

  #Init vars for E(pi_i f(x)) and E(pi_i z_j f(x))
  E_pi_fx = 0
  E_pi_zfx = np.zeros(d)

  #iterate over every binary combination of trudges
  for i in range(2**d):
    #turn i into binary shape with length d
    bit = np.array([int(x) for x in "{0:b}".format(i)])
    zeros = np.zeros(d-len(bit))
    zi = np.concatenate((zeros, bit)).astype(float)

    #trudge original x and forward pass
    xi = trudge_input_mean(x, grid(x, splits=4), zi)
    yi = net(xi.view(1,1,28,28))[0][pred_x].item()
    pii = compute_psi(1-sum(zi.astype(float))/len(zi))
    E_pi_fx += yi*pii
    E_pi_zfx += zi*yi*pii

  #Calculate beta according to Corollary 3

  beta0 = (compute_sigma_0(d)*E_pi_fx+compute_sigma_1(d)*sum(E_pi_zfx))
    /(compute_dencst(d)* 2**d)
  betaj = [(compute_sigma_1(d)*E_pi_fx+compute_sigma_2(d)*E_pi_zfx[j]
            +compute_sigma_3(d)*(sum(E_pi_zfx)-E_pi_zfx[j]))/
                (compute_dencst(d)* 2**d) for j in range(d)]
  return np.insert(betaj, 0, beta0).reshape(d+1,1)

  #Here I recycled def of sigma, dencst etc. from Garreaus LIME on github

```

C LIME Text Explainer

C.1 Initialization

```
# pip install transformers
#Import official LIME implemenation
from LIME.LIME_text import LIMETextExplainer

#Further imports not mentioned in LIME Image
from sklearn.linear_model import Lasso
from transformers import GPT2LMHeadModel, GPT2Tokenizer

#Initialize pre-trained GPT-2 model
model = GPT2LMHeadModel.from_pretrained('gpt2')

#Initialize tokenizer
tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
tokenizer.pad_token = tokenizer.eos_token
tokenizer.padding_side="left"

#Current instance sentence to be interpreted
sentence = "UK Parliament has implemented new"
words = sentence.split(" ")

#Set (n,d)
n = 500
d = len(words)
```

C.2 General utilities

```
#Generate fixed design Z, (small d)
def all_comb(d):
  lenst ='{0:0'+str(d)+'b}'
  return np.flip([[int(i) for i in list(lenst.format(j))] for j in range(2**d)], a
```

```
# Given design Z trudge sentence
def trudge(sentence, Z):
  words = sentence.split(" ")
  texts = []
  for i in range(Z.shape[0]):
    trudged = ""
    for j in range(Z.shape[1]-1):
      if bool(Z[i,j]):
        trudged += words[j] + " "

    if bool(Z[i,Z.shape[1]-1]):
        trudged += words[Z.shape[1]-1]

    if(len(trudged)>0):
      if(trudged[-1] == " "):
        trudged = trudged[:-1]
    texts.append(trudged)
  return texts

# Appends interaction columns to design Z
def append_interaction_columns(arr1):
  arr=arr1
  for i in range(arr1.shape[1]):
    for j in range(i+1, arr1.shape[1]):
      arr = np.concatenate(
        (arr, np.expand_dims(
                np.multiply(arr[:, i], arr[:, j]), axis=1)), axis=1
        )
  return arr
#Returns string list of words and mutual interactions
def include_interactions(words):
  ls = list(words)
  for i in range(len(words)):
    for j in range(i+1, len(words)):
      add = words[i]+"-"+words[j]
      ls.append(add)
  return ls
```

C.3 GPT2 model

```
#Propagate tokens through GPT2 model
def GPT_prob(token, model):
  out = model.generate(**token, max_new_tokens=1,
  return_dict_in_generate=True, output_scores=True)

  prob_labels = torch.softmax(out["scores"][-1], dim=-1)
  top_label = np.argsort(prob_labels[0])[-1]
  name_top_label= tokenizer.decode(top_label)

  print(f"Top label:{name_top_label} \nTop P: {round(prob_labels[0,
  top_label].item(), 3)}\n")

  return prob_labels[:, top_label]

#Compute psi values for weights
def compute_psi(t,nu=0.25):
  return np.exp(-np.square(1.0-np.sqrt(1.0 - t))/(2*nu**2))

#Compute weights for regression
def pi(Z):
  return compute_psi(np.subtract(np.ones(Z.shape[0]), np.sum(Z,
                       axis=1)/Z.shape[1]))

#Print results of regression
def print_results(feature_names, regressor, Z, y, weights,
                    interaction=False):
  rounded = [round(c, 5) for c in regressor.coef_]
  coefs = dict(zip(feature_names, rounded))

  print(f"Intercept: {round(regressor.intercept_, 5)}
  print(f"Coefficients: {coefs}")
  print(f"Weighted R^2: {round(regressor.score(Z, y,
        sample_weight=weights),5)}")
  return

```

```
#Run LIME on a language model
def LIME_TEXT(model, token, feature_names, Z, model_regressor,
                interaction=False):
  y = GPT_prob(token, model)
  weights = pi(Z)

  if interaction:
    Z = append_interaction_columns(Z)

  Z = Z.astype(float)

  reg = model_regressor.fit(Z, y, sample_weight=weights)
  print_results(feature_names, reg, Z, y, weights)
  return
```

C.4 Examples

```
#EXAMPLE 1: RANDOM DESIGN Z, Attention LIME
Z = stats.bernoulli.rvs(0.5, size=(n,d), random_state=7)
Z[0] = np.ones(d)

token = tokenizer([sentence]*n, return_tensors="pt")
token["attention_mask"] = torch.from_numpy(Z)

#Linear Regression (no interaction)
LIME_TEXT(model, token, words, Z, LinearRegression(fit_intercept=True))

```

```
#EXAMPLE 2: Fixed design Z (n=2**d)
Z = all_comb(d)

#Remove words according to Z
texts = trudge(sentence, Z)
token = tokenizer(texts, return_tensors="pt", padding='longest')

```

```
#Include interactions
features = include_interactions(words)

#Run LIME with LASSO Regression lambda=.01 including interactions
LIME_TEXT(model, token, features, Z,
        model_regressor=Lasso(alpha=.01, fit_intercept=True),
        interaction=True)
```

C.5 Official LIME

```
#For documentation see

# LIME_base.py
# LIME_text.py
# LIME_image.py

# at https://github.com/marcotcr/LIME/tree/master/LIME

def OfficialLIME_GPT2(texts):
    token = tokenizer(texts, return_tensors="pt", padding='longest')
    out = model.generate(
            **token,
            max_new_tokens=1,
            return_dict_in_generate=True,
            output_scores=True)

    prob_labels = torch.softmax(out["scores"][-1], dim=-1)
    top_label = np.argsort(prob_labels[0])[-1:] #<-- only diff to GPT2

    return np.array(prob_labels[:, top_label])
```

```
explainer = LIMETextExplainer()

exp = explainer.explain_instance(
        text_instance=sentence,
        classifier_fn=OfficialLIME_GPT2,
        top_labels=1,
        num_features=5,
        num_samples=500,
        model_regressor=LinearRegression(fit_intercept=True))

print(f"Intercept: {exp.intercept}")
print(f"Coefficients: {exp.as_list(label=0)}")
print(f"score: {exp.score}")

```