Uncertainty Quantification with Split Conformal Prediction

Thomas Davies

Edinburgh AI Society Advanced Workshop

November 4, 2024

Motivation

- Machine learning appears hugely effective across many different problems
- However we don't know how certain we are about the predictions we make
  - A self-driving car needs to know when it's uncertain about an object
  - Medical diagnosis systems must express uncertainty in critical decisions
- Deep learning is notorious for being overly confident[1][2]
- Can we quantify uncertainty with minimal assumptions?

[1] Guo et al. "On Calibration of Modern Neural Networks"

[2] Begoli et al. "The need for uncertainty quantification in machine-assisted medical decision making"

What is a Confidence Interval?

- Informally a confidence interval is an interval which is expected to typically contain the parameter being estimated.
- For example: "The average height of 10 people is 170cm $\pm$ 5cm with 95% confidence"
- The confidence level (e.g. 95%) tells us how often the interval contains the true value
  - If we repeat the experiment many times, 95% of the intervals will contain the true value
- A wider interval gives you more "confidence" but gives you less precision about the location of the parameter

The Goal

- Given any supervised machine learning algorithm that maps paired training examples to a function:

$$(X_1, Y_1), \ldots, (X_n, Y_n) \mapsto \hat{f}_{1:n} \tag{1}$$

- For a new data point X_{n+1}, we would like to construct a 95% "confidence set" for Y_{n+1}
- This is a (random) set $C(X_{n+1})$ where we have that

$$\mathbb{P}(Y_{n+1} \in C(X_{n+1})) = 0.95$$

- Without any assumptions on our learning algorithm, or underlying model this seems impossible!

Motivation: Ranks of Random Variables

- Consider some i.i.d. real valued random variables $Z_1, \ldots, Z_{n+1}$
- What is the probability that Z_{n+1} is the k-th largest?
- By symmetry (exchangeability), all the orderings are equally likely
- Therefore:

$$\mathbb{P}(\text{Rank}(Z_{n+1}) = k) = \frac{1}{n+1} \tag{2}$$

- This holds regardless of the distribution of Z_i!

A Note on Exchangeability

- Actually, we didn't need the i.i.d. assumption
- We only need exchangeability for any permutation π:

$$(Z_1, \ldots, Z_{n+1}) \stackrel{d}{=} (Z_{\pi(1)}, \ldots, Z_{\pi(n+1)})$$

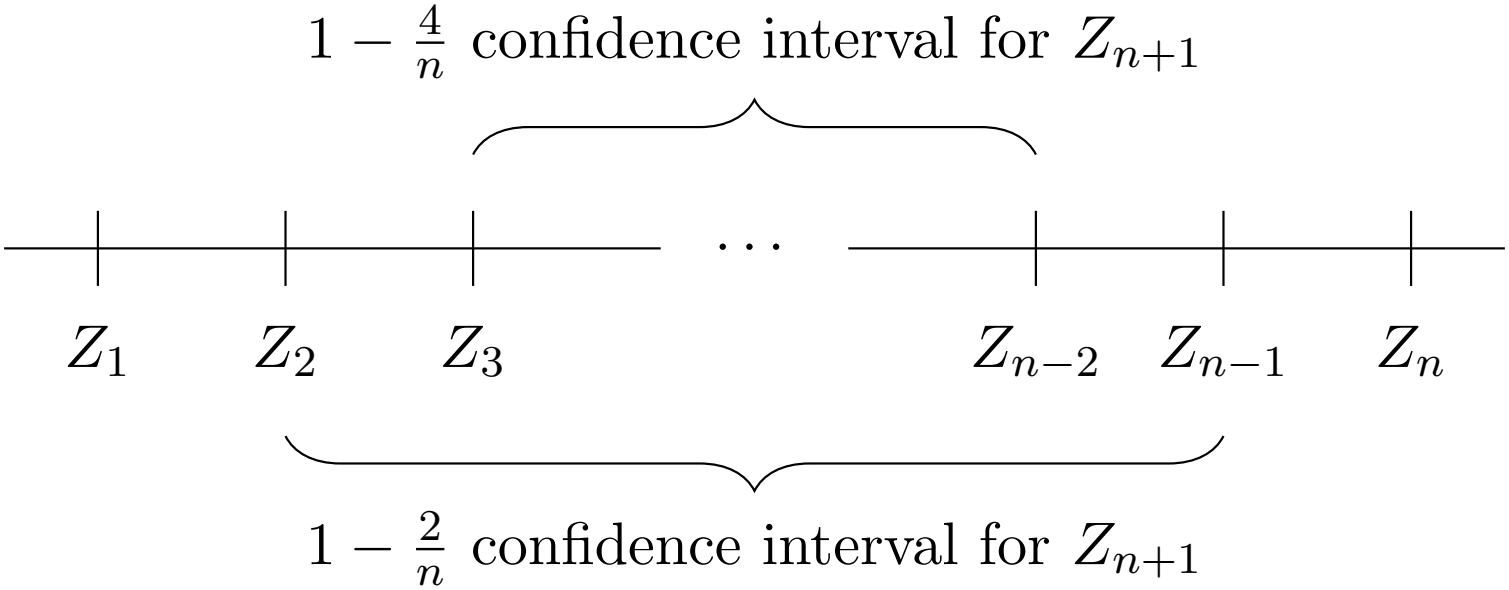
- This is much weaker than i.i.d.
- Example: Drawing without replacement from an urn
  - Not independent (draws affect future probabilities)
  - But exchangeable (order doesn't matter)

Building Confidence Intervals from Ranks

- If we have $n+1$ exchangeable random variables, we know their ranks are uniform
- We can use this to build confidence intervals!
- For a $(1-\alpha)$ confidence interval:
  - We want Z_{n+1} to be between the $\lceil \frac{\alpha}{2}(n+1) \rceil$-th and $\lfloor (1-\frac{\alpha}{2})(n+1) \rfloor$-th order statistics
- This works regardless of the underlying distribution!

Visualizing the Interval

- Assuming the variables are ordered as $Z_1, Z_2, \ldots, Z_n$ we can visualise this as.



An Example

- With n = 99 calibration points and $\alpha = 0.05$:
  - Lower bound: 3rd smallest value
  - Upper bound: 97th smallest value
  - Probability new point falls in this interval $= 1 - \alpha = 0.95$
- This gives us a concrete way to build prediction intervals!
- And we still maintain our distribution-free guarantee

Building Exchangeable Variables

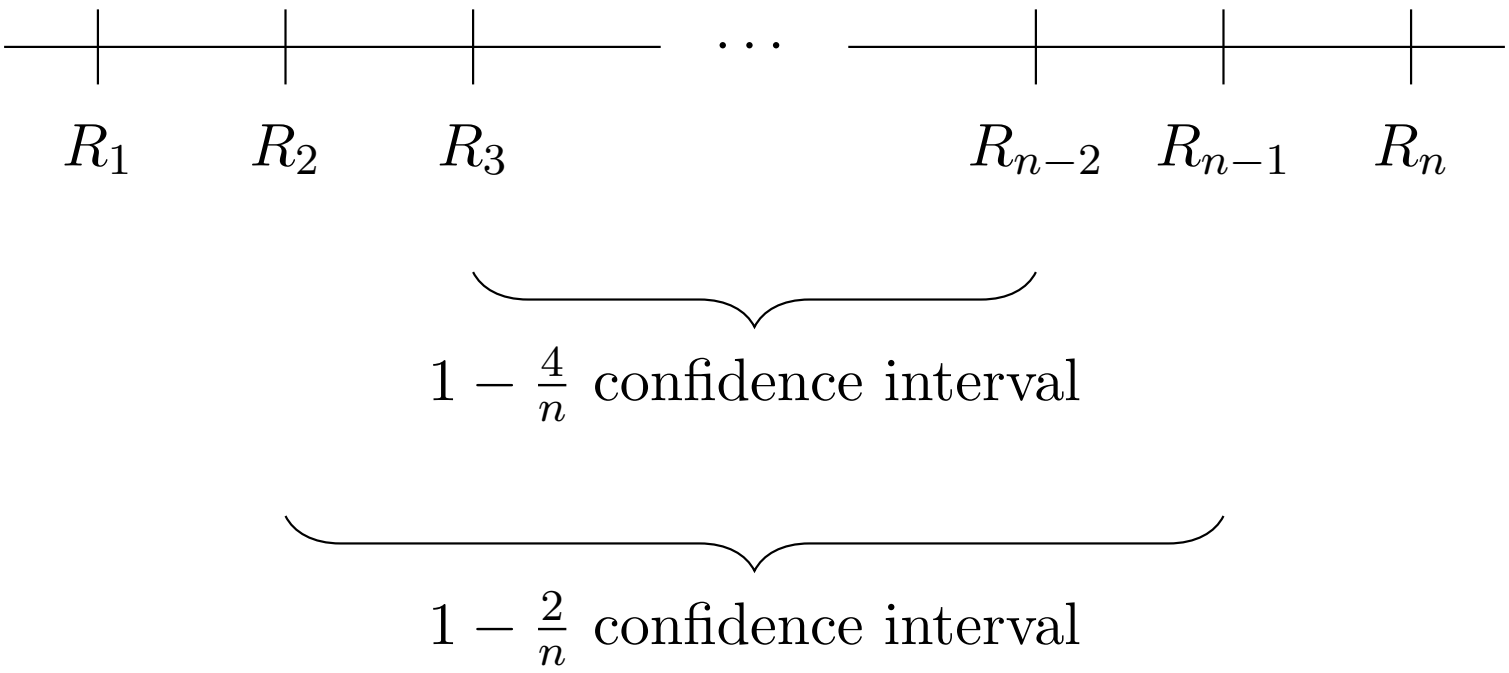
- Let's apply this to machine learning predictions by trying to build some exchangeable random variables - let's look at regression first.
- We split our training pairs into "pure training data" $\mathcal{I}_1$ and "conformal calibrating data" $\mathcal{I}_2$
- Let's train our model on $\mathcal{I}_1$ to get $\hat{f}(x)$
  and then we can evaluate our model on $\mathcal{I}_2$ to generate

$$R_i = |\hat{f}(X_i) - Y_i| \text{ for } (X_i, Y_i) \in \mathcal{I}_2$$

- These R_i are now exchangeable!!! So we can build a confidence interval for our next prediction using the ranking technique as before.

Visualizing the Residuals

- We know that the residuals must be positive so we only need to worry about one side of the interval.
- Assuming the residuals are ordered as $R_1, R_2, \ldots, R_n$ we can visualise this as.



The Split Conformal Prediction Algorithm

- For a new point X_{n+1}, compute $R_{n+1}(y) = |\hat{f}(X_{n+1}) - y|$ for candidate values y
- Our confidence set $C(X_{n+1})$ is all values of y where $R_{n+1}(y)$ is "not too large" compared to the calibration scores
- Specifically: $y \in C(X_{n+1})$ if $R_{n+1}(y)$ is smaller than the $\lceil (1-\alpha)(n+1) \rceil$-th largest calibration score
- This gives us valid $(1-\alpha)$ coverage by the rank arguments above!

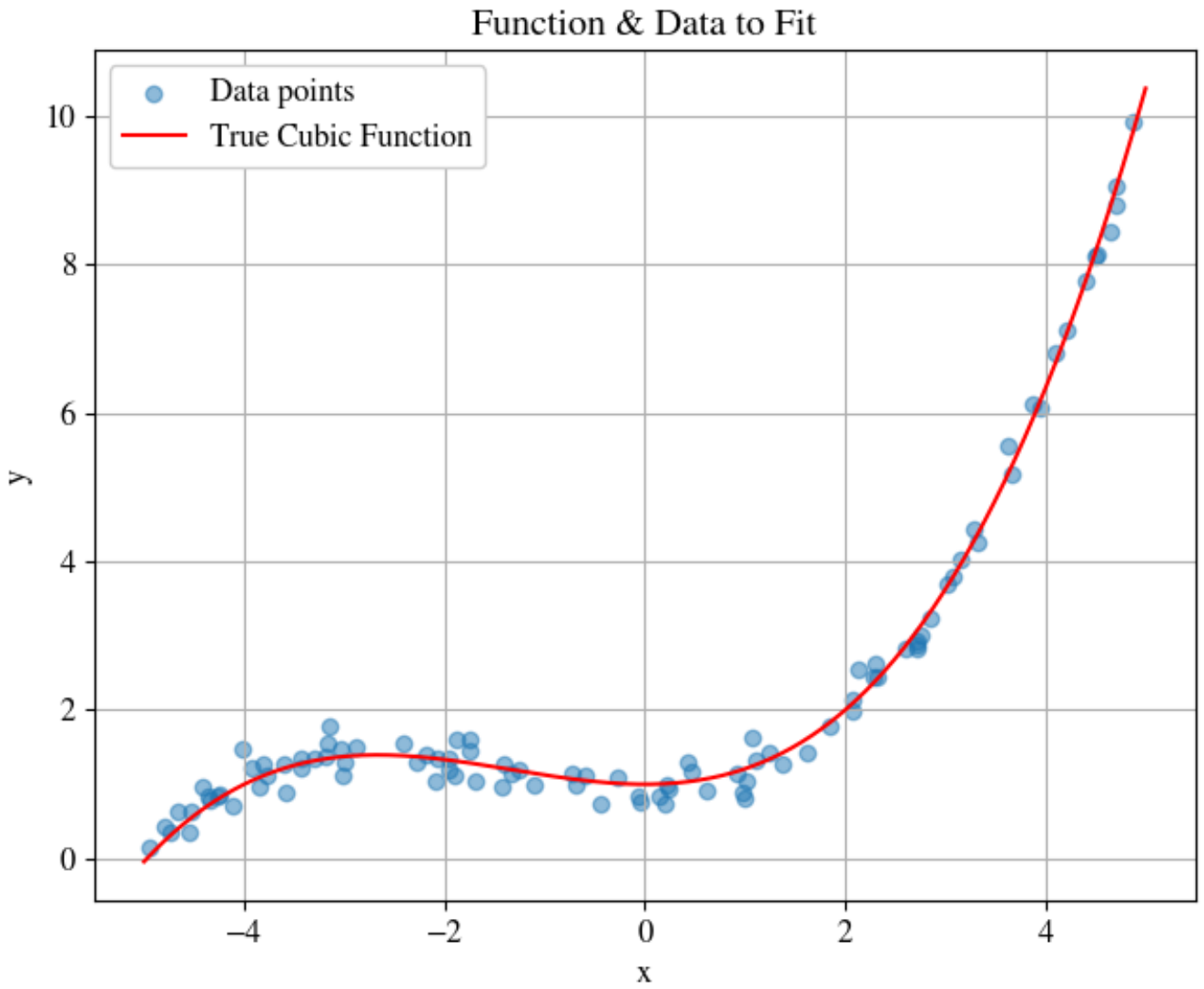
A Toy Example

- Let's run a synthetic experiment to demonstrate conformal prediction
- We'll generate data from:
  - X drawn from Uniform(-5, 5)
  - Y follows a cubic relationship with noise:

$$Y = 1 + \frac{1}{6}X^2 + \frac{1}{24}X^3 + \epsilon$$

    where $\epsilon \sim \mathcal{N}(0, 0.2)$
- This gives us ground truth to test drive our prediction intervals

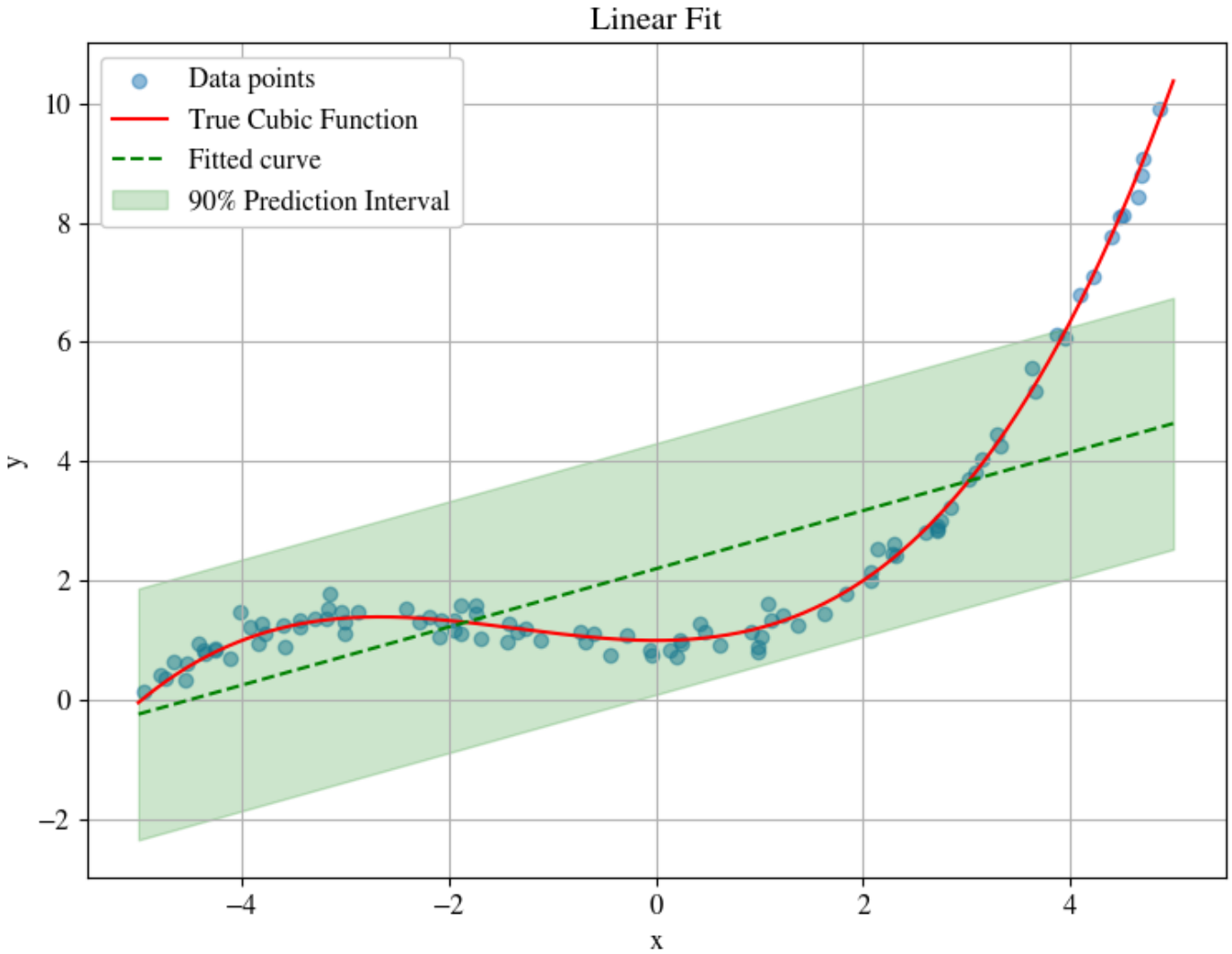
Visualizing the Data



Linear Model with Conformal Intervals

- Let's fit a linear model to this data - deliberately underfit!
- We should see that our conformal confidence interval gives a reasonably large interval
- The green region shows the 90% prediction interval
- Notice how:
  - The intervals are valid despite model misspecification
  - The intervals are very wide - we can't predict well so have high uncertainty

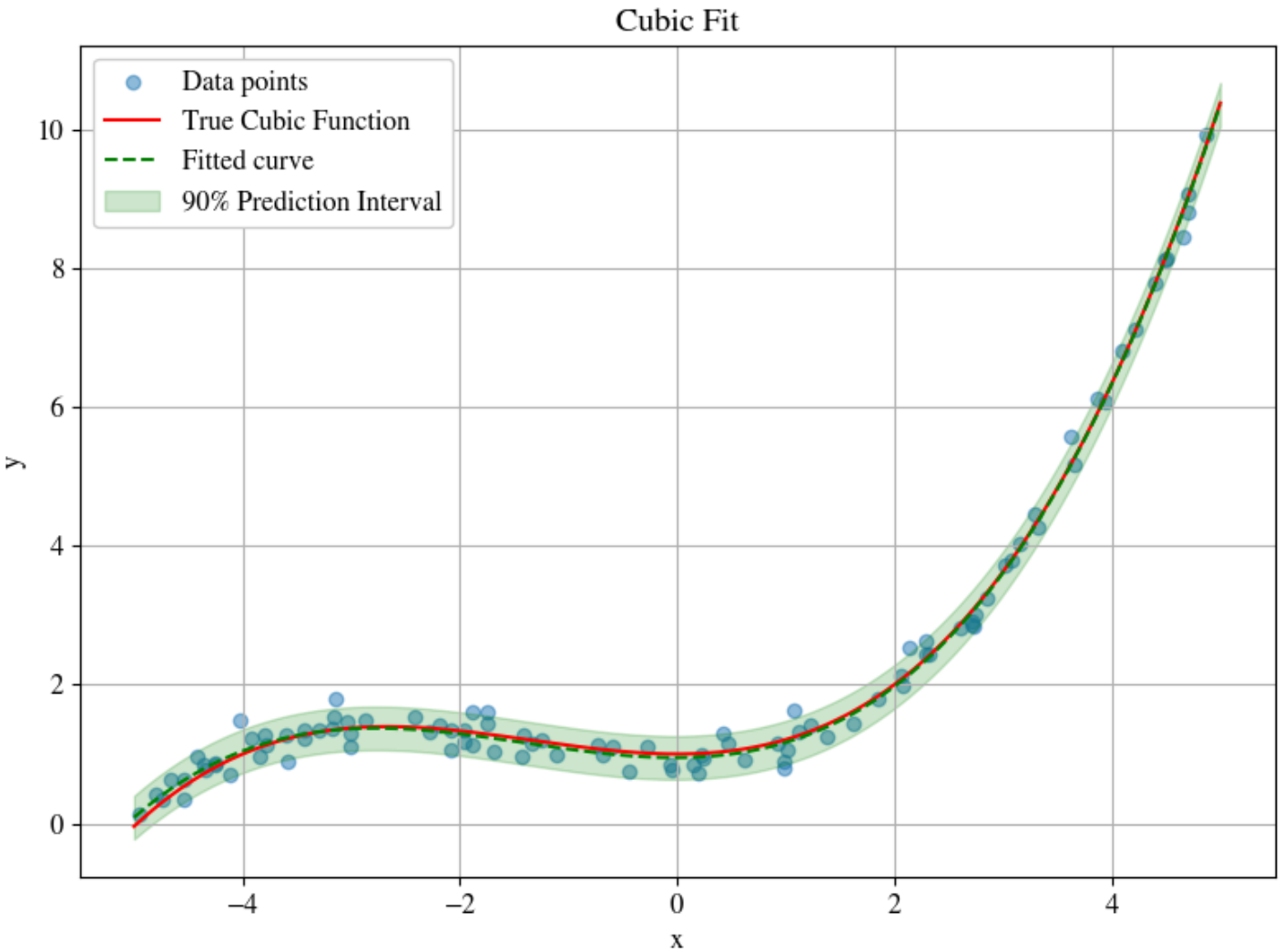
Linear Fit with Conformal Intervals



Linear Model with Conformal Intervals

- Let's now fit a cubic model to this data - this should give a much better fit
- We should see a small prediction interval

Cubic Fit with Conformal Intervals



Confidence Interval Size vs Model Complexity

Polynomial Degree	CI Width (lower is better)
0 (Constant)	5.03
2 (Linear)	3.82
2 (Quadratic)	2.40
3 (Cubic)	0.552
4 (Quartic)	0.583
5 (Quintic)	0.601

- Notice how the confidence interval width:
  - Decreases as we move from underfitting (degree 0) to good fit (degree 3)
  - Starts increasing again with higher degrees due to overfitting
- This shows that we can indeed use this to do model selection!

Code

All of the conformal prediction was 9 lines of code!

```
def get_conformal_predictor(x, y, degree=3):
    x_train, x_calib, y_train, y_calib = train_test_split(
        x, y, test_size=0.5, random_state=42
    )

    coeffs = np.polyfit(x_train, y_train, degree)

    y_pred_calib = np.polyval(coeffs, x_calib)
    residuals = np.abs(y_calib - y_pred_calib)

    def confinterval(x_new, alpha=0.9):
        y_pred = np.polyval(coeffs, x_new)
        q = np.quantile(residuals, alpha)
        return y_pred - q, y_pred + q

    return coeffs, confinterval
```