Polynomial Regression, Regularization, and The Bias Variance Tradeoff

1 Polynomial Regression

It is straightforward to incorporate a y intercept into our matrix notation. The trick to doing so is to append an extra dimension to the parameter vector $\boldsymbol{w} \in \mathbb{R}^d$ and to likewise append an extra feature/column to X, whose value is always 1, so that w_{d+1} represents the intercept:

$$X\boldsymbol{w} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{nd} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ w_{d+1} \end{bmatrix} = \begin{bmatrix} w_1x_{11} + w_2x_{12} + \ldots + w_dx_{1d} + w_{d+1} \\ w_1x_{21} + w_2x_{22} + \ldots + w_dx_{2d} + w_{d+1} \\ \vdots \\ w_1x_{n1} + w_2x_{n2} + \ldots + w_dx_{nd} + w_{d+1} \end{bmatrix}$$

In a simple regression of y on $X = \begin{bmatrix} \boldsymbol{x} \end{bmatrix}$, i.e., only one feature, the aforementioned trick is akin to appending $\boldsymbol{x}^0 = \mathbf{1}$ to X, resulting in $\begin{bmatrix} \boldsymbol{x} & 1 \end{bmatrix}$, or $\begin{bmatrix} 1 & \boldsymbol{x} \end{bmatrix}$. More generally, it is equally possible to append $\boldsymbol{x}$ raised to any other power p to X as well: e.g., $\begin{bmatrix} 1 & \boldsymbol{x} & \boldsymbol{x}^2 & \boldsymbol{x}^3 & \dots & \boldsymbol{x}^p \end{bmatrix}$. Therefore, **polynomial regression** reduces to linear regression! In theory at least; in practice, the set of possible combinations of powers of features is massive. Assuming just two features, say $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and $p = 3$ yields the following (long) list of possibilities:[1]

$$1 \;\; \boldsymbol{x}_1 \;\; \boldsymbol{x}_1^2 \;\; \boldsymbol{x}_1^3 \;\; \boldsymbol{x}_2 \;\; \boldsymbol{x}_2^2 \;\; \boldsymbol{x}_2^3 \;\; \boldsymbol{x}_1\boldsymbol{x}_2 \;\; \boldsymbol{x}_1^2\boldsymbol{x}_2 \;\; \boldsymbol{x}_1\boldsymbol{x}_2^2 \;\; \boldsymbol{x}_1^2\boldsymbol{x}_2^2 \;\; \boldsymbol{x}_1^3\boldsymbol{x}_2 \;\; \boldsymbol{x}_1\boldsymbol{x}_2^3 \;\; \boldsymbol{x}_1^3\boldsymbol{x}_2^2 \;\; \boldsymbol{x}_1^2\boldsymbol{x}_2^3 \;\; \boldsymbol{x}_1^3\boldsymbol{x}_2^3$$

Feature selection is an important and difficult problem in machine learning. The deep learning revolution can in large part be attributed to its success at automatically identifying pertinent features.

2 Regularized Regression

Linear regression is a machine learning model with a strong bias, namely the decision to fit a *line* (as opposed to a curve) to data. Even with this inherent bias, linear regression models can have high variance. Another way to limit variance is to limit the range of the model parameters $\boldsymbol{w} \in \mathbb{R}^d$.

The p-**norm** of a vector $\boldsymbol{w} \in \mathbb{R}^d$, denoted $\|\boldsymbol{w}\|_p$, for some $p \geq 1$, is a way to gauge $\boldsymbol{w}$'s size.[2]

The most popular norm is the 2-norm, also called the **Euclidean norm**:

$$\|\boldsymbol{w}\|_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$$

[1] I probably missed some!

[2] A norm is a function from $\mathbb{R}^d \to \mathbb{R}$ that satisfies the following three properties:

1. $\|\boldsymbol{x}\| > 0$, for all $\boldsymbol{x} \neq 0$
2. $\|\alpha\boldsymbol{x}\| = |\alpha|\|\boldsymbol{x}\|$, for all $\alpha \in \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{R}^d$
3. $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$

The final property is called the **triangle inequality**, because the sum of the lengths of two sides of a triangle is at least that of the third.

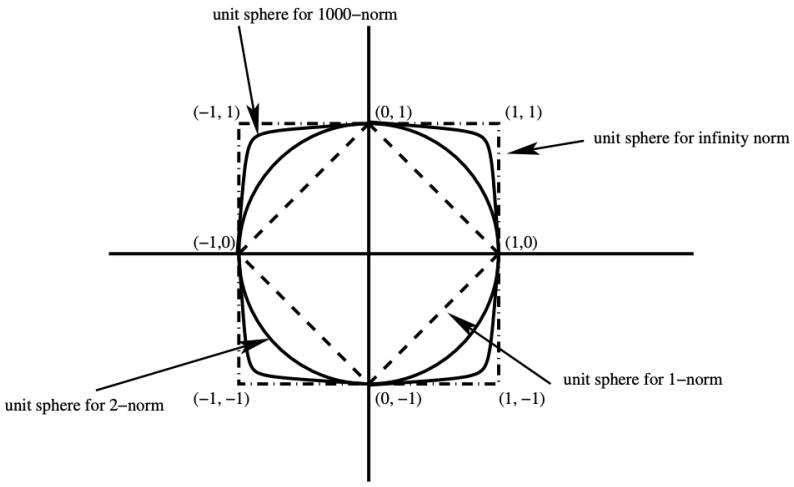

Figure 1: The feasible sets when the p-norm of a vector in $\mathbb{R}^d$ is constrained to be less than or equal to 1, for $p \in \{1, 2, 1000, \infty\}$. Image source.

Other common choices include the 1-norm:

$$\|\boldsymbol{w}\|_1 = \sum_{i=1}^{d} |w_i|$$

and the ∞-norm:

$$\|\boldsymbol{w}\|_\infty = \max_{i \in \{1,\ldots,d\}} |w_i|$$

Nomenclature An optimization problem with decision variables $\boldsymbol{z} \in \mathbb{R}^d$ is called **constrained** when its solution must lie in some smaller subset, say C, of $\mathbb{R}^d$. This smaller subset is called the **feasible set**. (In an **unconstrained** optimization problem, the solution can be found anywhere in $\mathbb{R}^d$.) Figure 1 depicts the feasible sets when the p-norm of a vector in $\mathbb{R}^d$ is constrained to be less than or equal to 1, for $p \in \{1, 2, 1000, \infty\}$.

Recall the ordinary least squares (OLS) objective:

$$\text{loss}(\boldsymbol{w}) = (\boldsymbol{y} - X\boldsymbol{w})^T(\boldsymbol{y} - X\boldsymbol{w}) = \sum_{i=1}^{n} (\boldsymbol{y} - X\boldsymbol{w})_i^2$$

In other words,

$$\text{loss}(\boldsymbol{w}) = \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2$$

Observe that OLS is an unconstrained optimization problem, as $\boldsymbol{w}$ can take on any value in $\mathbb{R}^d$. **Regularized regression** is a constrained optimization problem, with the same objective, but a limited domain for $\boldsymbol{w}$.

Ridge regression uses the 2-norm as a regularizer: for some $\beta > 0$,

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2$$
$$\text{subject to } \|\boldsymbol{w}\|_2 \leq \beta$$

LASSO uses the 1-norm: for some $\beta > 0$,

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2$$
$$\text{subject to } \|\boldsymbol{w}\|_1 \leq \beta$$

The choice of β is a choice of how much bias to include in the model.

Like OLS, ridge regression has a closed-form solution. This solution can be found by first reformulating the ridge regression constrained optimization problem in terms of its Lagrangian,and then following the same steps as last time to solve OLS. The closed-form solution of ridge regression generalizes that of OLS:

$$\boldsymbol{w} = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}$$

On the other hand, LASSO does not have a closed-form solution. The difficulty arises from the fact that the absolute value function is not differentiable at zero. As a result, LASSO is generally solved using a generalization of gradient descent called *subgradient* descent, as the subgradient exists everywhere. The subgradient, however, is not unique, so this algorithm is usually slower to converge than gradient descent.

While both ridge regression and LASSO impose bias, and thereby limit variance, their behavior is notably different. Take, for example, the vectors $(1, 0)$ and $(1/\sqrt{2}, 1/\sqrt{2})$. While $\|(1,0)\|_2 = \|(1/\sqrt{2}, 1/\sqrt{2})\|_2 = 1$, only $(1, 0)$ has 1-norm 1; $\|(1/\sqrt{2}, 1/\sqrt{2})\|_1 = \sqrt{2}$. As a result, LASSO has a tendency to select features, by completely zeroing out less important features—not just assigning them small coefficients.

Figure 2 is a depiction of ridge regression and LASSO. The OLS minimum is the dark black dot. As this point lies outside the feasible set, the optimal point for each problem lies on the boundary of the feasible set. But only for LASSO does it fall on a corner; the coefficient in the y direction is completely zeroed out.

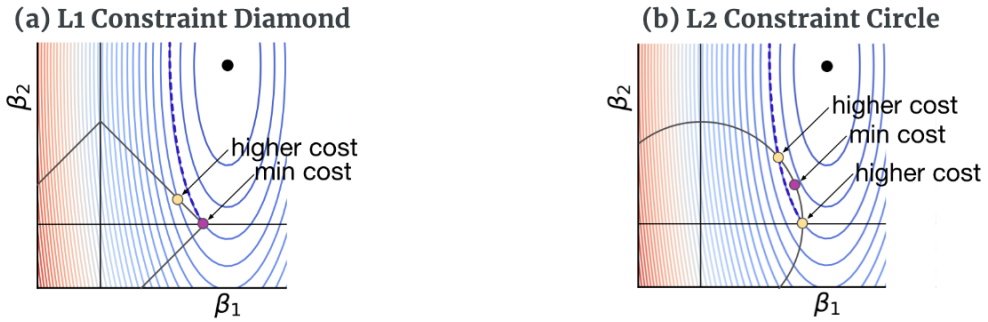

Figure 3.2. L1 and L2 constraint regions get different coefficient locations, on the diamond and circle, for the same loss function. Keep in mind that there are an infinite number of contour lines and there is a contour line exactly meeting the L2 purple dot.

Figure 2: Image source.

3 Bias-Variance Tradeoff

Machine learning models make predictions that deviate from the true values due to various sources of error. Understanding these sources of error is crucial for building effective models and diagnosing their shortcomings.

3.1 Definitions

Consider a supervised learning setting where we have a target variable y that depends on features $\boldsymbol{x}$ according to some unknown true relationship:

$$y = f(\boldsymbol{x}) + \epsilon$$

where $f(\boldsymbol{x})$ is the true function we wish to learn and ϵ represents irreducible noise with $\mathbb{E}[\epsilon] = 0$ and $\text{Var}(\epsilon) = \sigma^2$.

Given a training dataset $\mathcal{D}$, we train a model that produces predictions $\hat{f}(\boldsymbol{x}; \mathcal{D})$. Because different training datasets will yield different models, we can think of $\hat{f}$ as a random variable that depends on the particular training data drawn.

The **expected prediction error** for a model at a point $\boldsymbol{x}$ can be decomposed as:

$$\mathbb{E}_{\mathcal{D}}\left[(y - \hat{f}(\boldsymbol{x}; \mathcal{D}))^2\right] = \text{Bias}^2[\hat{f}(\boldsymbol{x})] + \text{Var}[\hat{f}(\boldsymbol{x})] + \sigma^2$$

Let us define each component:

Bias measures the average deviation of our model's predictions from the true values:

$$\text{Bias}[\hat{f}(\boldsymbol{x})] = \mathbb{E}_{\mathcal{D}}[\hat{f}(\boldsymbol{x}; \mathcal{D})] - f(\boldsymbol{x})$$

High bias indicates that the model is systematically missing relevant patterns in the data. This typically occurs when the model is too simple to capture the underlying structure—a phenomenon called **underfitting**. For example, fitting a linear model to data generated by a polynomial function will result in high bias.

Variance measures how much the model's predictions vary across different training datasets:

$$\text{Var}[\hat{f}(\boldsymbol{x})] = \mathbb{E}_{\mathcal{D}}\left[(\hat{f}(\boldsymbol{x}; \mathcal{D}) - \mathbb{E}_{\mathcal{D}}[\hat{f}(\boldsymbol{x}; \mathcal{D})])^2\right]$$

High variance indicates that the model is overly sensitive to the particular training data used. This typically occurs when the model is too complex and fits noise in the training data rather than the underlying signal—a phenomenon called **overfitting**. For example, fitting a high-degree polynomial to a small dataset will result in high variance.

Irreducible Error σ^2 represents the noise inherent in the data that cannot be reduced by any model. This error is due to factors not captured by the features $\boldsymbol{x}$ or random measurement noise.

3.2 The Tradeoff

The term "tradeoff" arises because, in practice, reducing bias often increases variance and vice versa. This relationship stems from model complexity:

Simple models (low complexity):

- High bias: Cannot capture complex patterns in the data
- Low variance: Predictions are stable across different training sets
- Example: Linear regression on nonlinear data

Complex models (high complexity):

- Low bias: Can fit intricate patterns in the data
- High variance: Predictions fluctuate significantly with different training sets
- Example: High-degree polynomial regression

The optimal model complexity minimizes the total expected error, which requires balancing bias and variance. This optimal point typically lies somewhere between the extremes of underfitting and overfitting.

3.3 Regularization and the Bias-Variance Tradeoff

Regularization provides a principled way to navigate the bias-variance tradeoff. By constraining model parameters, regularization intentionally introduces bias to achieve a more substantial reduction in variance.

Consider again the ridge regression optimization problem:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2$$
$$\text{subject to } \|\boldsymbol{w}\|_2 \leq \beta$$

The regularization parameter β (or equivalently, λ in the Lagrangian formulation) controls the bias-variance tradeoff:

Small β (strong regularization):

- Forces $\boldsymbol{w}$ to be small, limiting the model's flexibility
- *Increases bias*: Model cannot fit training data as closely
- *Decreases variance*: Model is less sensitive to particular training samples
- Risk: Underfitting if β is too small

Large β (weak regularization):

- Allows $\boldsymbol{w}$ to take on larger values, increasing model flexibility
- *Decreases bias*: Model can better approximate the true function
- *Increases variance*: Model may fit noise in the training data
- Risk: Overfitting if β is too large

As $\beta \to \infty$, ridge regression approaches ordinary least squares (no regularization). As $\beta \to 0$, the model approaches the trivial solution $\boldsymbol{w} = \mathbf{0}$.

The closed-form solution for ridge regression, $\boldsymbol{w} = (X^T X + \lambda I)^{-1} X^T \boldsymbol{y}$, makes the regularization effect explicit. The term λI added to $X^T X$ shrinks the coefficients toward zero, with the strength of shrinkage controlled by λ (where λ is inversely related to β). This shrinkage reduces variance at the cost of introducing bias.

LASSO and feature selection: LASSO regularization (ℓ_1-norm) not only manages the bias-variance tradeoff but also performs automatic feature selection. By driving some coefficients exactly to zero, LASSO effectively reduces model complexity, which can simultaneously reduce both bias (by removing irrelevant

features that add noise) and variance (by simplifying the model). This makes LASSO particularly valuable when dealing with high-dimensional data where many features may be irrelevant.

In practice, the optimal choice of regularization strength is typically determined through **cross-validation**, where the model is trained and evaluated on multiple subsets of the data to estimate its generalization performance.