Foundations of AI **Fall 2024**
Professors Greenwald and Ewing **Machine Learning**

Linear Regression, Continued

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:

$$X\boldsymbol{w} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} & 1 \\ x_{21} & x_{22} & \dots & x_{2d} & 1 \\ \vdots & \vdots & \dots & \vdots & \\ x_{n1} & x_{n2} & \dots & x_{nd} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_d \\ w_{d+1} \end{bmatrix} = \begin{bmatrix} w_1 x_{11} + w_2 x_{12} + \ldots + w_d x_{1d} + w_{d+1} \\ w_2 x_{21} + w_2 x_{12} + \ldots + w_d x_{1d} + w_{d+1} \\ \vdots \\ w_1 x_{n1} + w_2 x_{n2} + \ldots + w_d x_{nd} + w_{d+1} \end{bmatrix}$$

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

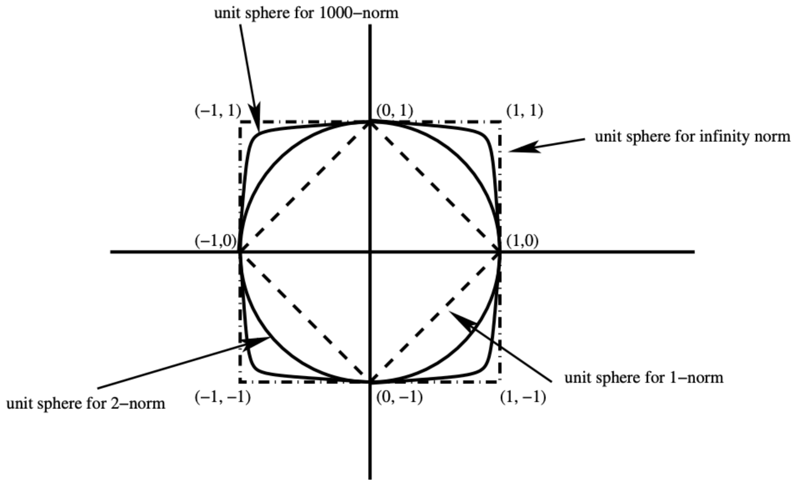

Figure 1: The feasible sets when the p-norm of a vector in $\mathbb{R}^d$ is constrained to be less than or equal to 1, for $p \in \{1, 2, 1000, \infty\}$. Image source.

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 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 $\beta > 0$,

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 \tag{1}$$

$$\text{subject to} \quad \|\boldsymbol{w}\|_2 \le \beta \tag{2}$$

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

$$\begin{aligned} & \min_{\boldsymbol{w} \in \mathbb{R}^d} \|\boldsymbol{y} - X\boldsymbol{w}\|_2^2 && (3) \\ \text{subject to} \quad & \|\boldsymbol{w}\|_1 \leq \beta && (4) \end{aligned}$$

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,[3] 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 completing 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.

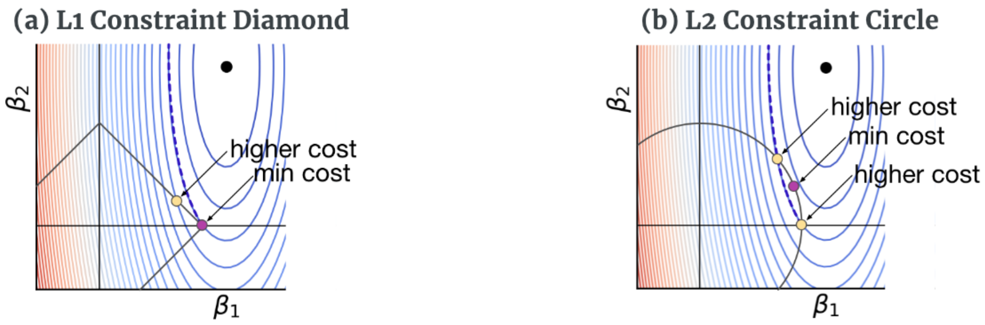

Figure 3.2. L1 and L2 constraint regions get different coefficient locations, on the diamond and circle, for the same loss function. Keep in mind that there are an infinite number of contour lines and there is a contour line exactly meeting the L2 purple dot.

Figure 2: Image source.

[3] TODO !!!