CS189: Machine Learning

UC Berkeley

Kelvin Lee

January 1, 2021

These are course notes for UC Berkeley's Spring 2021 CS189 Machine Learning, instructed by Professor Jonathan Shewchuk.

Contents

1 Regression I

The goal in machine learning is to extract a *relationship* from data. In **regression** tasks, this relationship is described by a function $y = f(\mathbf{x})$, where $y \in \mathbb{R}$ can be predicted from some collection of numerical measurements $\mathbf{x} \in \mathbb{R}^d$.

The true relationship f is unknown and the goal is to recover it as well as we can from data. The end product is a function $\hat{y} = h(\mathbf{x})$, called the **hypothesis**, that approximates f using a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each pair $(\mathbf{x}_i, y_i)$ is some possibly noisy datapoints to be learned.

Learning arbitrary functions is intractable, so we restrict ourselves to some **hypothesis class** $\mathcal{H}$ of allowable functions. In particular, we employ a **parametric model**, meaning that there is some finite-dimensional vector $\mathbf{w} \in \mathbb{R}^d$, the elements of which are known as **parameters** or **weights**, that controls the behavior of the function. That is,

$$h_{\mathrm{w}}(\mathbf{x}) = g(\mathbf{x}, \mathbf{w})$$

for some other function g. The hypothesis class is then the set of all functions induced by the possible choices of the parameters w:

$$\mathcal{H} = \{h_{\mathbf{w}} \mid \mathbf{w} \in \mathbb{R}^d\}$$

After designating a **cost function** L, which measures how poorly the predictions $\hat{y}$ of the hypothesis match the true output y, we can proceed to search for the parameters that best fit the data by minimizing this function:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\mathbf{w})$$

1.1 Ordinary Least Squares

Ordinary least squares (OLS) is a **linear regression** problem, in which we take $h_{\mathbf{w}}$ to be of the form $h_{\mathbf{w}}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$ and we want for each $i = 1, \ldots, n$ that

$$y_i \approx \hat{y}_i = h_{\mathbf{w}}(\mathbf{x}_i) = \mathbf{x}_i^\top \mathbf{w}.$$

The equivalent matrix form is

$$\underbrace{\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}}_{\mathbf{y}} \approx \underbrace{\begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix}}_{\mathbf{X}} \underbrace{\begin{bmatrix} w_1 \\ \vdots \\ w_d \end{bmatrix}}_{\mathbf{w}}$$

The matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ is called the **design matrix** and usually $n \geq d$, i.e., there are more datapoints than measurements. Thus, generally there will be no exact solution to the equation. However, we can approximate the solution by minimizing the sum of the squared errors:

$$L(\mathbf{w}) = \sum_{i=1}^{n} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 = \min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2,$$

which is an optimization problem.

1.1.1 Approach 1: Vector Calculus

Recall that the **gradient** ∇f, is a column vector containing the first-order partial derivatives of f :

$$\nabla f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}$$

If a function $L : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable, then any local optimum $\mathbf{w}^*$ satisfies $\nabla L(\mathbf{w}^*) = \mathbf{0}$. In the OLS case, we have

$$\begin{aligned} L(\mathbf{w}) &= \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \\ &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\top (\mathbf{X}\mathbf{w} - \mathbf{y}) \\ &= (\mathbf{X}\mathbf{w})^\top \mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X}\mathbf{w} + \mathbf{y}^\top \mathbf{y} \\ &= \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}. \end{aligned}$$

Recall from matrix calculus that

$$\nabla_{\mathbf{x}}(\mathbf{a}^\top \mathbf{x}) = \mathbf{a} \tag{1}$$

$$\nabla_{\mathbf{x}}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}. \tag{2}$$

The proof for the first equation is left as an exercise. Here's the proof for the second equation:

Proof. We first derive an expression for $\mathbf{x}^\top \mathbf{A}\mathbf{x}$:

$$\begin{aligned} \mathbf{x}^T \mathbf{A}\mathbf{x} &= \begin{bmatrix} x_1 & \cdots & x_n \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= \begin{bmatrix} \sum_{i=1}^n a_{i1}x_i & \cdots & \sum_{i=1}^n a_{in}x_i \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ &= \sum_{j=1}^n \sum_{i=1}^n a_{ij}x_i x_j. \end{aligned}$$

We know that

$$\frac{\partial \mathbf{x}^T \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{x}^T \mathbf{A}\mathbf{x}}{\partial x_1} \\ \vdots \\ \frac{\partial \mathbf{x}^T \mathbf{A}\mathbf{x}}{\partial x_n} \end{bmatrix}.$$

Then consider the k^{th} row in the above vector:

$$\begin{aligned}
\frac{\partial \mathbf{x}^T \mathbf{A}\mathbf{x}}{\partial x_k} &= \frac{\partial}{\partial x_k}\left(\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}x_i x_j\right) \\
&= \frac{\partial}{\partial x_k}\left(x_1\sum_{i=1}^{n} a_{i1}x_i + \cdots + x_k\sum_{i=1}^{n} a_{ik}x_i + \cdots + x_n\sum_{i=1}^{n} a_{in}x_i\right) \\
&= x_1 a_{k1} + \cdots + \left(\sum_{i=1}^{n} a_{ik}x_i + x_k a_{kk}\right) + \cdots + x_n a_{kn} \\
&= \sum_{j=1}^{n} a_{kj}x_j + \sum_{i=1}^{n} a_{ik}x_i \\
&= \Big(k^{th} \text{ row of } \mathbf{A}\Big)\,\mathbf{x} + \Big(\text{ transpose of } k^{\text{th}} \text{ column of } \mathbf{A}\Big)\,\mathbf{x} \\
&= \Big[\Big(k^{\text{th}} \text{ row of } \mathbf{A}\Big) + \Big(\text{ transpose of } k^{\text{th}} \text{ column of } \mathbf{A}\Big)\Big]\,\mathbf{x}.
\end{aligned}$$

Repeat this for each row and we have the desired result

$$\nabla_{\mathbf{x}}(\mathbf{x}^\top \mathbf{A}\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}.$$

□

Now back to where we were at, using the results above, we can compute the gradient of L:

$$\begin{aligned}
\nabla L(\mathbf{w}) &= \nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}) \\
&= \nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w}) - 2\nabla_{\mathbf{w}}(\mathbf{w}^\top \mathbf{X}^\top \mathbf{y}) + \underbrace{\nabla_{\mathbf{w}}(\mathbf{y}^\top \mathbf{y})}_{\mathbf{0}}, \\
&= 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y}
\end{aligned}$$

where the last line uses equation (2) from above and the symmetry of $\mathbf{X}^\top \mathbf{X}$ to obtain $\mathbf{X}^\top \mathbf{X} + (\mathbf{X}^\top \mathbf{X})^\top = 2\mathbf{X}^\top \mathbf{X}$. By setting this gradient to $\mathbf{0}$, we conclude that any optimum $\mathbf{w}^*_{\text{OLS}}$ satisfies

$$\mathbf{X}^\top \mathbf{X}\mathbf{w}^*_{\text{OLS}} = \mathbf{X}^\top \mathbf{y}.$$

If $\mathbf{X}$ is full rank, then $\mathbf{X}^\top \mathbf{X}$ is as well ($n \geq d$), so we can solve for a unique solution

$$\boxed{\mathbf{w}^*_{\text{OLS}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}}.$$

Remark. Note that $\nabla L(\mathbf{w}^*) = \mathbf{0}$ is a *necessary* but not *sufficient* condition for optimality. Recall from calculus that a critical point could be a local minimum, a local maximum, or a saddle point. In this case, we have a **convex** objective function and so any critical point is a global minimum. We can show that L is convex by computing the **Hessian** of L, which is

$$\nabla^2 L(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X}$$

and showing that it is positive semi-definite:

$$\mathbf{w}^\top (2\mathbf{X}^\top \mathbf{X})\mathbf{w} = 2(\mathbf{X}\mathbf{w})^\top \mathbf{X}\mathbf{w} = 2\|\mathbf{X}\mathbf{w}\|_2^2 \geq 0.$$

1.1.2 Approach 2: Orthogonal projection

We can also use orthogonal projections to obtain the same solution. Recall that if V is an inner product space and S a subspace of V, then any $\mathbf{v} \in V$ can be decomposed uniquely in the form

$$\mathbf{v} = \mathbf{v}_s + \mathbf{v}_\perp$$

where $\mathbf{v}_s \in S$ and $\mathbf{v}_\perp \in S^\perp$. Here $S^\perp$ is the **orthogonal complement** of S, which is the set of vectors that are perpendicular to every vector in S. The following is a proof of this statement, feel free to skip it for those who are not interested.

Proof. We want to show that $V = S \oplus S^\perp$ for any finite-dimensional subspace S of V. First we show that

$$V = S + S^\perp.$$

Suppose $\mathbf{v} \in V$. Let $\mathbf{e}_1, \ldots, \mathbf{e}_m$ be an orthonormal basis of S. Then

$$\mathbf{v} = \underbrace{\langle \mathbf{v}, \mathbf{e}_1 \rangle \, \mathbf{e_1} + \cdots + \langle \mathbf{v}, \mathbf{e}_m \rangle \, \mathbf{e_m}}_{\mathbf{v}_S} + \underbrace{\mathbf{v} - \langle \mathbf{v}, \mathbf{e}_1 \rangle \, \mathbf{e}_1 - \cdots - \langle \mathbf{v}, \mathbf{e}_m \rangle \, \mathbf{e}_m}_{\mathbf{v}_\perp}$$

Clearly $\mathbf{v}_S \in S$. Since $\mathbf{e}_1, \ldots, \mathbf{e}_m$ is an orthonormal list, for each $j = 1, \ldots, m$ we have

$$\begin{aligned} \langle \mathbf{v}_\perp, \mathbf{e}_j \rangle &= \langle \mathbf{v}, \mathbf{e}_j \rangle - \langle \mathbf{v}, \mathbf{e}_j \rangle \\ &= \mathbf{0}, \end{aligned}$$

which implies that $\mathbf{v}_\perp$ is orthogonal to every vector in S, i.e. $\mathbf{v}_\perp \in S^\perp$. Thus we have $\mathbf{v} = \mathbf{v}_S + \mathbf{v}_\perp$, completing our first part of the proof.

Now we need to show that $S \cap S^\perp = \{0\}$. First we prove the following lemma:

Lemma 1. If S is a subset of V, then $S \cap S^\top \subset \{0\}$.

Proof of lemma 1. Suppose S is a subset of V and $\mathbf{v} \in S \cap S^\perp$. Then $\langle \mathbf{v}, \mathbf{v} \rangle = 0$, which implies that $\mathbf{v} = \mathbf{0}$. Thus $S \cap S^\perp \subset \{0\}$ □

Now using lemma 1 we know that $S \cap S^\perp = \{0\}$. Along with the first part, we conclude that $V = S \oplus S^\perp$. □

The **orthogonal projection** onto S, denoted P_S, is the linear operator that maps $\mathbf{v}$ to $\mathbf{v}_S$ in the decomposition above. Recall the property that

$$\|\mathbf{v} - P_S\mathbf{v}\| \leq \|\mathbf{v} - \mathbf{s}\|$$

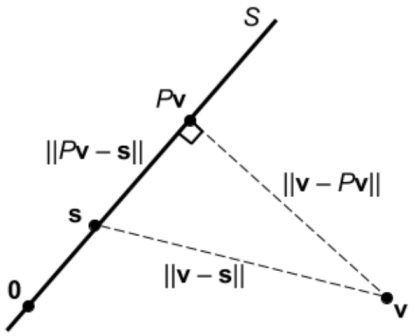
for all $\mathbf{s} \in S$, with equality if and only if $\mathbf{s} = P_s\mathbf{v}$. Rewriting the equation above, we actually have

$$P_S\mathbf{v} = \arg\min_{\mathbf{s} \in S} \|\mathbf{v} - \mathbf{s}\|.$$

Proof. Using the Pythagorean theorem, we get

$$\|\mathbf{v} - \mathbf{s}\|^2 = \|\underbrace{\mathbf{v} - P_S\mathbf{v}}_{\in S^\perp} + \underbrace{P_S\mathbf{v} - \mathbf{s}}_{\in S}\|^2 = \|\mathbf{v} - P_S\mathbf{v}\|^2 + \|P_S\mathbf{v} - \mathbf{s}\|^2 \geq \|\mathbf{v} - P_S\mathbf{v}\|^2$$

with equality if and only if $\|P_S\mathbf{v} - \mathbf{s}\|^2 = 0$, i.e. $\mathbf{s} = P_S\mathbf{v}$. Taking square roots on both sides gives $\|\mathbf{v} - \mathbf{s}\| \geq \|\mathbf{v} - P_S\mathbf{v}\|$ as desired. □



We have the following for the OLS case,

$$\mathbf{w}^*_{\text{OLS}} = \arg\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2.$$

Observe that $P_{\text{range }(\mathbf{X})}\mathbf{y} = \mathbf{X}\mathbf{w}^*_{\text{OLS}}$. The projected point $\mathbf{X}\mathbf{w}^*_{\text{oLs}}$ is always unique. If $\mathbf{X}$ is full rank ($n \geq d$), then the optimum $\mathbf{w}^*_{\text{oLs}}$ is also unique because the columns of X are linearly independent. We now need the following lemma to solve for $\mathbf{w}^*_{\text{OLS}}$:

Lemma 2.

$$\text{null}(\mathbf{X}^\top) = \text{range}(\mathbf{X})^\perp$$

Proof. Suppose $\mathbf{X}$ is a mapping from V to W and let $\mathbf{w} \in W$. Then

$$\begin{aligned}
\mathbf{w} \in \text{ null}(\mathbf{X}^\top) &\iff \mathbf{X}^\top\mathbf{w} = \mathbf{0} \\
&\iff \langle \mathbf{v}, \mathbf{X}^\top\mathbf{w}\rangle = 0 \text{ for all } \mathbf{v} \in V \\
&\iff \langle \mathbf{X}\mathbf{v}, \mathbf{w}\rangle = 0 \text{ for all } \mathbf{v} \in V \\
&\iff \mathbf{w} \in \text{range}(\mathbf{X})^\perp.
\end{aligned}$$

□

Since we are projecting onto range($\mathbf{X}$), we have $\mathbf{y} - P\mathbf{y} \perp \text{range}(\mathbf{X})$, i.e. $\mathbf{y} - \mathbf{X}\mathbf{w}^*_{\text{OLS}} \in \text{null}\left(\mathbf{X}^\top\right)$ by lemma 2. Finally we have

$$\mathbf{X}^\top\left(\mathbf{y} - \mathbf{X}\mathbf{w}^*_{\text{OLS}}\right) = \mathbf{0} \implies \mathbf{X}^\top\mathbf{X}\mathbf{w}^*_{\text{OLS}} = \mathbf{X}^\top\mathbf{y}.$$

1.2 Ridge Regression

While Ordinary Least Squares can be used for solving linear least squares problems, it falls short due to numerical instability and generalization issues. Numerical instability arises when the features of the data are close to collinear (leading to linearly dependent feature columns), causing the input matrix $\mathbf{X}$ to lose its rank or have singular values that very close to 0. Why are small singular values bad? Let us illustrate this via the singular value decomposition (SVD) of X:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

where $\mathbf{U} \in \mathbb{R}^{n\times n}, \mathbf{\Sigma} \in \mathbb{R}^{n\times d}, \mathbf{V} \in \mathbb{R}^{d\times d}$. In the context of OLS, we must have that $\mathbf{X}^\top\mathbf{X}$ is invertible, or equivalently, $\text{rank}\left(\mathbf{X}^\top\mathbf{X}\right) = \text{rank}\left(\mathbf{X}^\top\right) = \text{rank}(\mathbf{X}) = d$. Assuming that $\mathbf{X}$ and $\mathbf{X}^\top$ are full column rank d, we can express the SVD of $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{U}\begin{bmatrix}\mathbf{\Sigma}_d \\ \mathbf{0}\end{bmatrix}\mathbf{V}^\top$$

where $\Sigma_d \in \mathbb{R}^{d \times d}$ is a diagonal matrix with strictly positive entries. Now let's try to expand the $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$ term in OLS using the SVD of $\mathbf{X}$

$$\begin{aligned}\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} &= \left(\mathbf{V} \begin{bmatrix}\boldsymbol{\Sigma}_d & \mathbf{0}\end{bmatrix} \mathbf{U}^\top \mathbf{U} \begin{bmatrix}\boldsymbol{\Sigma}_d \\ \mathbf{0}\end{bmatrix} \mathbf{V}^\top\right)^{-1} \\ &= \left(\mathbf{V} \begin{bmatrix}\boldsymbol{\Sigma}_d & \mathbf{0}\end{bmatrix} \mathbf{I} \begin{bmatrix}\boldsymbol{\Sigma}_d \\ \mathbf{0}\end{bmatrix} \mathbf{V}^\top\right)^{-1} \\ &= \left(\mathbf{V}\boldsymbol{\Sigma}_d^2\mathbf{V}^{\mathrm{T}}\right)^{-1} = \left(\mathbf{V}^\top\right)^{-1} \left(\boldsymbol{\Sigma}_d^2\right)^{-1} \mathbf{V}^{-1} = \mathbf{V}\boldsymbol{\Sigma}_d^{-2}\mathbf{V}^\top\end{aligned}$$

This means that $\left(\mathbf{X}^\top \mathbf{X}\right)^{-1}$ will have singular values that are the squared inverse of the singular values of $\mathbf{X}$, potentially leading to extremely large singular values when the singular value of $\mathbf{X}$ are close to 0. Such excessively large singular values can be very problematic for numerical stability purposes. In addition, abnormally high values to the optimal $\mathbf{w}$ solution would prevent OLS from generalizing to unseen data.

There is a very simple solution to these issues: penalize the entries of $\mathbf{w}$ from becoming too large. We can do this by adding a penalty term constraining the norm of $\mathbf{w}$. For a fixed, small scalar $\lambda > 0$, we now have:

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

Note that the λ in our objective function is a hyperparameter that measures the sensitivity to the values in $\mathbf{w}$. Just like the degree in polynomial features, λ is a value that we must choose arbitrarily through validation. Let's expand the terms of the objective function:

$$\begin{aligned}L(\mathbf{w}) &= \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 \\ &= \mathbf{w}^\top \mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{w}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y} + \lambda \mathbf{w}^\top \mathbf{w}\end{aligned}$$

Finally take the gradient of the objective and find the value of $\mathbf{w}$ that achieves $\mathbf{0}$ for the gradient:

$$\begin{aligned}\nabla_{\mathrm{w}} L(\mathbf{w}) &= \mathbf{0} \\ 2\mathbf{X}^\top \mathbf{X}\mathbf{w} - 2\mathbf{X}^\top \mathbf{y} + 2\lambda \mathbf{w} &= \mathbf{0} \\ \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right) \mathbf{w} &= \mathbf{X}^\top \mathbf{y} \\ \mathbf{w}^*_{\mathrm{RIDGE}} &= \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \mathbf{y}.\end{aligned}$$

This value is guaranteed to achieve the (unique) global minimum, because the objective function is strongly convex. To show that f is strongly convex, it suffices to compute the Hessian of f, which in this case is

$$\nabla^2 L(\mathbf{w}) = 2\mathbf{X}^\top \mathbf{X} + 2\lambda \mathbf{I}$$

and show that this is positive definite (PD)

$$\forall \mathbf{w} \neq \mathbf{0}, \mathbf{w}^\top \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right) \mathbf{w} = (\mathbf{X}\mathbf{w})^\top \mathbf{X}\mathbf{w} + \lambda \mathbf{w}^\top \mathbf{w} = \|\mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2 > 0$$

since the Hessian is positive definite, we can equivalently say that the eigenvalues of the Hessian are strictly positive and that the objective function is strongly convex. A useful property of strongly convex functions is that they have a unique optimum point, so the solution to ridge regression is unique. We cannot make such guarantees about ordinary least squares, because the corresponding Hessian could have eigenvalues that are 0. Let us explore the case in OLS when the Hessian has a 0 eigenvalue. In this context, the term $\mathbf{X}^\top \mathbf{X}$ is not invertible, but this does not imply that no solution exists! In OLS, there always exists a solution, and when the Hessian is PD that solution is unique; when the Hessian is PSD, there are infinitely many

solutions. (There always exists a solution to the expression $\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}$, because the range of $\mathbf{X}^\top\mathbf{X}$ and the range space of $\mathbf{X}^\top$ are equivalent; since $\mathbf{X}^\top\mathbf{y}$ lies in the range of $\mathbf{X}^\top$, it must equivalently lie in the range of $\mathbf{X}^\top\mathbf{X}$ and therefore there always exists a $\mathbf{w}$ that satisfies the equation $\mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y}$.) The technique we just described is known as ridge regression. Note that now the expression $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ is invertible, regardless of rank of $\mathbf{X}$. Let's find $\left(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}\right)^{-1}$ through SVD Now with our slight tweak, the matrix $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ has become full rank and thus invertible. The singular values have become $\frac{1}{\sigma^2+\lambda}$ and $\frac{1}{\lambda}$, meaning that the singular values are guaranteed to be at most $\frac{1}{\lambda}$, solving our numerical instability issues. Furthermore, we have partially solved the overfitting issue. By penalizing the norm of x, we encourage the weights corresponding to relevant features that capture the main structure of the true model, and penalize the weights corresponding to complex features that only serve to fine tune the model and fit noise in the data.

1.3 Feature Engineering

The least-squares optimization problem

$$\min_{\mathbf{w}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2$$

represents the best-fit *linear* model via a projection of $\mathbf{y}$ onto the subspace spanned by the columns of $\mathbf{X}$. However, note that the true relationship $y = f(\mathbf{x})$ can be nonlinear, so nonlinear models are useful as well. In fact, we can do this under the framework of linear least-squares by augmenting the data with new **features**. Specifically, we design some **feature map** $\boldsymbol{\phi} : \mathbb{R}^\ell \to \mathbb{R}^d$ that maps $\mathbf{x} \in \mathbb{R}^\ell$ into a vector of features $\boldsymbol{\phi}(\mathbf{x})$, which is a linear combination of **basis functions** ϕ_j. The hypothesis function is as follows:

$$h_{\mathbf{w}}(\mathbf{x}) = \sum_{j=1}^{d} w_j \phi_j(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}).$$

Note that although the model is still linear with respect to the features, it is nonlinear with respect to the original data if $\boldsymbol{\phi}$ is nonlinear.
We replace $\mathbf{X} \in \mathbb{R}^{n\times\ell}$ by $\boldsymbol{\Phi} \in \mathbb{R}^{n\times d}$, which has $\boldsymbol{\phi}(\mathbf{x}_i)^\top$ as its ith row, and use least-squares:

$$\min_{\mathbf{w}} \|\boldsymbol{\Phi}\mathbf{w} - \mathbf{y}\|_2^2.$$

Example 1.1 (Fitting Ellipses). Let's use least-squares to estimate the parameters of an ellipse from data. Assume that we have n data points $\mathcal{D} = \{(x_{1,i}, x_{2,i})\}_{i=1}^n$, which may be noisy (i.e. could be off the actual orbit). Our goal is to determine the relationship between x_1 and x_2. We assume that the ellipse from which the points were generated has the form

$$w_1 x_1^2 + w_2 x_2^2 + w_3 x_1 x_2 + w_4 x_1 + w_5 x_2 = 1$$

where the coefficients $w_1, \ldots, w_5$ are the parameters we wish to estimate. We formulate the problem with least-squares:

$$\min_{\mathbf{w}} \|\mathbf{\Phi w} - \mathbf{1}\|_2^2$$

where

$$\Phi = \begin{bmatrix} x_{1,1}^2 & x_{2,1}^2 & x_{1,1}x_{2,1} & x_{1,1} & x_{2,1} \\ x_{1,2}^2 & x_{2,2}^2 & x_{1,2}x_{2,2} & x_{1,2} & x_{2,2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1,n}^2 & x_{2,n}^2 & x_{1,n}x_{2,n} & x_{1,n} & x_{2,n} \end{bmatrix}$$

In this case, the feature map ϕ is given by

$$\phi(x) = \left(x_1^2, x_2^2, x_1 x_2, x_1, x_2\right)$$

Note that there is no "target" vector $\mathbf{y}$ here, so this is not a traditional regression problem, but it still fits into the framework of least-squares.

2 Regression II

2.1 Probabilistic Model

In **supervised learning**, we assume that there exists a true underlying model mapping inputs to outputs:

$$f : \mathbf{x} \to f(\mathbf{x}),$$

which is unknown, and our goal is to find a hypothesis model that best represents it. The only information given is a dataset

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$$

where $\mathbf{x}_i \in \mathbb{R}^d$ is the input and $y_i \in \mathbb{R}$ is the observation, a noisy version of the true output $f(\mathbf{x}_i)$

$$Y_i = f(\mathbf{x}_i) + Z_i.$$

Assume that $\mathbf{x}_i$ is a fixed value (which implies that $f(\mathbf{x}_i)$ is fixed as well), while Z_i is a random variable (Y_i as well). Assume that Z_i has zero mean, because otherwise there would be systematic bias in our observations. The Z_i's could be Gaussian, uniform, Laplacian, etc.. In most contexts, we assume that they are i.i.d Gaussians: $Z_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \sigma^2\right)$. We can therefore say that Y_i is a random variable whose probability distribution is given by

$$Y_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(f(\mathbf{x}_i), \sigma^2\right)$$

Now that we have defined the model and data, we wish to find a hypothesis model h_θ (parameterized by θ) that best captures the relationships in the data, while possibly taking into account prior beliefs that we have about the true model. We can represent this as a probability problem, where the goal is to find the optimal model that maximizes our probability.

2.2 Maximum Likelihood Estimation

In **Maximum Likelihood Estimation** (MLE), the goal is to find the hypothesis model that maximizes the probability of the data, or the **likelihood**. Suppose the set of models are parameterized with θ. The problem can be expressed as

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) = p(\text{data} = \mathcal{D} \mid \text{true model} = h_{\boldsymbol{\theta}}).$$

Substituting the representation of $\mathcal{D}$ we have

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = p(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{\theta})$$

Since logarithms are monotonic functions, we can further simplify the problem by working with the **log likelihood** $\ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y})$

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = \arg\max_{\theta} \log \mathcal{L}(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}).$$

Using independence, we can decompose the log likelihood:

$$\ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) = \log p(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n, \boldsymbol{\theta}) = \log \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) = \sum_{i=1}^{n} \log \left[p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) \right].$$

Note that each $p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta})$ comes from a Gaussian

$$Y_i \mid \boldsymbol{\theta} \sim \mathcal{N}(h_{\boldsymbol{\theta}}(\mathbf{x}_i).\sigma^2)$$

Thus we have

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MLE}} &= \arg\max_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathbf{X}, \mathbf{y}) \\
&= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log \left[p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) \right] \\
&= \arg\max_{\boldsymbol{\theta}} - \left(\sum_{i=1}^{n} \frac{\left(y_i - h_{\boldsymbol{\theta}}(\mathbf{x}_i) \right)^2}{2\sigma^2} \right) - n \log \sqrt{2\pi}\sigma \\
&= \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{n} \frac{\left(y_i - h_{\boldsymbol{\theta}}(\mathbf{x}_i) \right)^2}{2\sigma^2} \right) + n \log \sqrt{2\pi}\sigma \\
&= \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left(y_i - h_{\boldsymbol{\theta}}\left(\mathbf{x}_i \right) \right)^2
\end{aligned}$$

Now consider the case of regression − the hypothesis has the form $h_\theta(\mathbf{x}_i) = \mathbf{x}_i^\top \boldsymbol{\theta}$, where $\boldsymbol{\theta} \in \mathbb{R}^d$. Then the problem becomes

$$\hat{\boldsymbol{\theta}}_{\text{MLE}} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\theta})^2$$

which is the OLS problem. This shows that OLS and MLE for regression lead to the same answer. We conclude that MLE is a probabilistic justification for why using squared error (which is the basis of OLS) is a good metric for evaluating a regression model.

2.3 Maximum a Posteriori

In **Maximum a Posteriori** (MAP) Estimation, the goal is to find the model for which the data maximizes the probability of the model, or the **posterior**:

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} p(\text{true model} = h_{\boldsymbol{\theta}} \mid \text{data} = \mathcal{D}).$$

We maximize the posterior using **Bayes' Rule**:

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\max_{\boldsymbol{\theta}} p(\text{true model} = h_\theta \mid \text{data} = \mathcal{D}) \\
&= \arg\max_{\boldsymbol{\theta}} \frac{p(\text{data} = \mathcal{D} \mid \text{true model} = h_\theta) \cdot p(\text{true model} = h_\theta)}{p(\text{data} = \mathcal{D})} \\
&= \arg\max_{\boldsymbol{\theta}} p(\text{data} = \mathcal{D} \mid \text{true model} = h_\theta) \cdot p(\text{true model} = h_\theta) \\
&= \arg\max_{\boldsymbol{\theta}} \log p(\text{data} = \mathcal{D} \mid \text{ true model } = h_\theta) + \log p(\text{true model} = h_\theta) \\
&= \arg\min_{\theta} - \log p(\text{data} = \mathcal{D} \mid \text{true model} = h_\theta) - \log p(\text{true model} = h_\theta) \\
&= \arg\min_{\boldsymbol{\theta}} - \left(\sum_{i=1}^{n} \log \left[p(y_i \mid \mathbf{x}_i, \boldsymbol{\theta}) \right] \right) - \log[p(\boldsymbol{\theta})]
\end{aligned}$$

MAP is similar to MLE except that the former has an extra [prior term $p(\text{true model} = h_\theta)$, which has the effect of favoring certain models over others *a priori*, regardless of the dataset. MLE is a special case of MAP when the prior is uniform. Assume as before that the noise terms are i.i.d. Gaussians: $N_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, \sigma^2\right)$. Also, assume for the prior term $P(\boldsymbol{\Theta})$ that the components θ_j are i.i.d. Gaussians:

$$\theta_j \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\theta_{j_0}, \sigma_h^2)$$

Then we have

$$\begin{aligned}
\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\min_{\boldsymbol{\theta}} \left(\frac{\sum_{i=1}^{n} \left(y_i - h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) \right)^2}{2\sigma^2} \right) + \left(\frac{\sum_{j=1}^{d} \left(\theta_j - \theta_{j_0} \right)^2}{2\sigma_h^2} \right) \\
&= \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{n} \left(y_i - h_{\boldsymbol{\theta}} \left(\mathbf{x}_i \right) \right)^2 \right) + \frac{\sigma^2}{\sigma_h^2} \left(\sum_{j=1}^{d} \left(\theta_j - \theta_{j_0} \right)^2 \right).
\end{aligned}$$

Let's look again at the case for linear regression to illustrate the effect of the prior term when $\theta_{j_0} = 0$. In this context, we refer to the linear hypothesis function $h_\theta(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^\top \boldsymbol{\theta} \right)^2 + \frac{\sigma^2}{\sigma_h^2} \sum_{j=1}^{d} \theta_j^2$$

This is just the Ridge Regression problem! We just proved that Ridge Regression and MAP for regression lead to the same answer! We can simply set $\lambda = \frac{\sigma^2}{\sigma_2^2}$. We conclude that MAP is a probabilistic justification for adding the penalized ridge term in Ridge Regression.

2.4 Multivariate Gaussian

Again, just as in MLE, notice that we implicitly condition on the $\mathbf{x}_i$'s because we treat them as constants. Also, let us assume as before that the noise terms are i.i.d. Gaussians: $N_i \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma^2\right)$. For the prior term $P(\Theta)$, we assume that the components θ_j are i.i.d. Gaussians:

$$\theta_j \overset{\text{iid}}{\sim} \mathcal{N}\left(\theta_{j_0}, \sigma_h^2\right)$$

Using this specific information, we now have:

$$\begin{aligned}\hat{\boldsymbol{\theta}}_{\text{MAP}} &= \arg\min_{\boldsymbol{\theta}} \left(\frac{\sum_{i=1}^{n} \left(y_i - h_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right)\right)^2}{2\sigma^2} \right) + \left(\frac{\sum_{j=1}^{d} \left(\theta_j - \theta_{j_0}\right)^2}{2\sigma_h^2} \right) \\ &= \arg\min_{\boldsymbol{\theta}} \left(\sum_{i=1}^{n} \left(y_i - h_{\boldsymbol{\theta}}\left(\mathbf{x}_i\right)\right)^2 \right) + \frac{\sigma^2}{\sigma_h^2} \left(\sum_{j=1}^{d} \left(\theta_j - \theta_{j_0}\right)^2 \right)\end{aligned}$$

Let's look again at the case for linear regression to illustrate the effect of the prior term when $\theta_{j_0} = 0$. In this context, we refer to the linear hypothesis function $h_\theta(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{x}$

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^\top \boldsymbol{\theta}\right)^2 + \frac{\sigma^2}{\sigma_h^2} \sum_{j=1}^{d} \theta_j^2$$

This is just the Ridge Regression problem! We just proved that Ridge Regression and MAP for regression lead to the same answer! We can simply set $\lambda = \frac{\sigma^2}{\sigma_2^2}$. We conclude that MAP is a probabilistic justification for adding the penalized ridge term in Ridge Regression. 2. A random vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_k)^\top$ is JG if $\sum_{i=1}^{k} a_i Z_i$ is normally distributed for every $a = (a_1, a_2, \ldots, a_k)^\top \in \mathbb{R}^k$ 3. (Non-degenerate case only) A random vector $\mathbf{Z} = (Z_1, Z_2, \ldots, Z_k)^\top$ is JG if Where $\boldsymbol{\Sigma} = \mathbb{E}\left[(\mathbf{Z} - \boldsymbol{\mu})(\mathbf{Z} - \boldsymbol{\mu})^\top\right] = \mathbb{E}\left[(\mathbf{RU})(\mathbf{RU})^\top\right] = \mathbf{R}\mathbb{E}\left[\mathbf{U}\mathbf{U}^\top\right]\mathbf{R}^\top = \mathbf{R}I\mathbf{R}^\top = \mathbf{R}\mathbf{R}^\top$ $\boldsymbol{\Sigma}$ is also called the covariance matrix of $\mathbf{Z}$. Note that all of these conditions are equivalent. In this note we will start by showing a proof that (1) $\implies$ (3).*We willlleaveitasanexercisetoprovetherestoftheimplicationsneededto* (3) In the context of the noise problem we defined earlier, we are starting with condition (1), ie. $\mathbf{Z} = \mathbf{RU}$ (in this case $k = l = n$), and we would like to derive the probability density of $\mathbf{Z}$. Note that here we removed the μ from consideration because in machine learning we always assume that the noise has a mean of 0. We leave it as an exercise for the reader to prove the case for an arbitrary μ We will first start by relating the probability density function of U to that of Z. Denote $f_{\text{U}}(\mathbf{u})$ as the probability density for $\mathbf{U} = \mathbf{u}$, and similarly denote $f_{\mathbf{Z}}(\mathbf{z})$ as the probability density for $\mathbf{Z} = \mathbf{z}$. One may initially believe that $f_{\text{U}}(\mathbf{u}) = f_{\mathbf{Z}}(\mathbf{Ru})$, but this is NOT true. Remember that since there is a change of variables from U to Z, we must make sure to incorporate the change of variables constant, which in this case is the absolute value of the determinant of $\mathbf{R}$. Incorporating this constant, we will have the correct formula:

$$f_{\mathbf{U}}(\mathbf{u}) = |\det(\mathbf{R})| f_{\mathbf{Z}}(\mathbf{Ru})$$

Let's see why this is true, with a simple 2D geometric explanation. Define U space to be the 2D space with axes U_1 and U_2. Now take any arbitrary region $\mathbf{R}'$ in $\mathbf{U}$ space (note that this $\mathbf{R}'$ is different from the matrix $\mathbf{R}$ that relates $\mathbf{U}$ to $\mathbf{Z}$). As shown in the diagram below, we have some off-centered circular region $\mathbf{R}'$ and we would like to approximate the probability that $\mathbf{U}$ takes a value in this region. We can do so by taking a Riemann sum of the density function $f_{\text{U}}(.)$ over smaller and smaller squares that make up the region $\mathbf{R}'$: Mathematically, we have that

$$p(\mathbf{U} \subseteq \mathbf{R}' = \iint_{\mathbf{R}'} f_{\mathbf{U}}\left(u_1, u_2\right) du_1 du_2 \approx \sum \sum_{\mathbf{R}'} f_{\mathbf{U}}\left(u_1, u_2\right) \Delta u_1 \Delta u_2$$

Now, let's apply the linear transformation $\mathbf{Z} = \mathbf{RU}$, mapping the region $\mathbf{R}'$ in $\mathbf{U}$ space, to the region $T\left(\mathbf{R}'\right)$ in $\mathbf{Z}$ space.

The graph on the right is now $\mathbf{Z}$ space, the 2D space with axes Z_1 and Z_2. Assuming that the matrix $\mathbf{R}$ is invertible, there is a one-to-one correspondence between points in $\mathbf{U}$ space to points in $\mathbf{Z}$ space. As we can note in the diagram above, each unit square in $\mathbf{U}$ space maps to a parallelogram in $\mathbf{Z}$ space (in higher dimensions, we would use the terms hypercube and

parallelepiped). Recall the relationship between each unit hypercube and the parallelepiped it maps to:

$$\text{Area(parallelepiped)} = |\det(\mathbf{R})| \cdot \text{ Area(hypercube)}$$

In this 2D example, if we denote the area of each unit square as $\Delta u_1 \Delta u_2$, and the area of each unit parallelepiped as ΔA, we say that

$$\Delta A = |\det(\mathbf{R})| \cdot \Delta u_1 \Delta u_2$$

Now let's take a Riemann sum to find the probability that $\mathbf{Z}$ takes a value in $T(\mathbf{R}')$:

$$\begin{aligned} p(\mathbf{Z} \subseteq T(\mathbf{R}') &= \iint_{T(\mathbf{R}')} f_{\mathbf{Z}}(z_1, z_2)\, dz_1 dz_2 \\ &\approx \sum \sum_{T(\mathbf{R}')} f_{\mathbf{Z}}(\mathbf{z}) \Delta A \\ &= \sum \sum_{\mathbf{R}'} f_{\mathbf{Z}}(\mathbf{Ru}) |\det(\mathbf{R})| \Delta u_1 \Delta u_2 \end{aligned}$$

Note the change of variables in the last step: we sum over the squares in U space, instead of parallelograms in $\mathbf{R}$ space. So far, we have shown that (for any dimension n)

$$p(\mathbf{U} \subseteq \mathbf{R}' = \int \ldots \iint_{\mathbf{R}'} f_{\mathbf{U}}(\mathbf{u}) du_1 du_2 \ldots du_n$$

Mathematically, we have that

$$p(\mathbf{U} \subseteq \mathbf{R}' = \iint_{\mathbf{R}'} f_{\mathbf{U}}(u_1, u_2)\, du_1 du_2 \approx \sum \sum_{\mathbf{R}'} f_{\mathbf{U}}(u_1, u_2)\, \Delta u_1 \Delta u_2$$

Now, let's apply the linear transformation $\mathbf{Z} = \mathbf{RU}$, mapping the region $\mathbf{R}'$ in $\mathbf{U}$ space, to the region $T(\mathbf{R}')$ in $\mathbf{Z}$ space.

The graph on the right is now $\mathbf{Z}$ space, the 2D space with axes Z_1 and Z_2. Assuming that the matrix $\mathbf{R}$ is invertible, there is a one-to-one correspondence between points in $\mathbf{U}$ space to points in $\mathbf{Z}$ space. As we can note in the diagram above, each unit square in $\mathbf{U}$ space maps to a parallelogram in $\mathbf{Z}$ space (in higher dimensions, we would use the terms hypercube and parallelepiped). Recall the relationship between each unit hypercube and the parallelepiped it maps to:

$$\text{Area(parallelepiped)} = |\det(\mathbf{R})| \cdot \text{ Area(hypercube)}$$

In this 2D example, if we denote the area of each unit square as $\Delta u_1 \Delta u_2$, and the area of each unit parallelepiped as ΔA, we say that

$$\Delta A = |\det(\mathbf{R})| \cdot \Delta u_1 \Delta u_2$$

Now let's take a Riemann sum to find the probability that $\mathbf{Z}$ takes a value in $T(\mathbf{R}')$:

$$\begin{aligned} p(\mathbf{Z} \subseteq T(\mathbf{R}') &= \iint_{T(\mathbf{R}')} f_{\mathbf{Z}}(z_1, z_2)\, dz_1 dz_2 \\ &\approx \sum \sum_{T(\mathbf{R}')} f_{\mathbf{Z}}(\mathbf{z}) \Delta A \\ &= \sum \sum_{\mathbf{R}'} f_{\mathbf{Z}}(\mathbf{Ru}) |\det(\mathbf{R})| \Delta u_1 \Delta u_2 \end{aligned}$$

Note the change of variables in the last step: we sum over the squares in U space, instead of parallelograms in $\mathbf{R}$ space. So far, we have shown that (for any dimension n)

$$p(\mathbf{U} \subseteq \mathbf{R}' = \int \ldots \iint_{\mathbf{R}'} f_{\mathbf{U}}(\mathbf{u}) du_1 du_2 \ldots du_n$$