第一章:基礎概念和方法

實證風險最小化 (Empirical Risk Minimization)

實證風險最小化（Empirical Risk Minimization, ERM）是機器學習中一個核心概念，其目標是降低預期的泛化誤差，即在未見過的數據上的性能表現。

$$J^*(\theta) = E_{(x,y)\sim p_{data}} L(f(x;\theta), y)$$

此數量稱為風險，這裡的"風險"是指模型預測錯誤的期望值，理想情況下應根據真實的數據分佈來計算，但在實際情況中，我們通常無法獲得或知道這個真實分佈。期望值是根據真實的底層分佈 p_{data}計算的，而不是根據經驗分佈 $\hat{p}_{data}$ 計算的。

如果我們知道真實分佈$p_{data}(\boldsymbol{x}, y)$，風險最小化將是一個可由優化演算法解決的優化任務。當我們不知道真實分佈 $p_{data}(\boldsymbol{x}, y)$，而只有訓練數據集時，我們轉而嘗試最小化在這個訓練集上的預期損失，也就是所謂的經驗風險。這是通過將真實分佈替換為訓練數據集定義的經驗分佈來實現的。經驗風險的計算涉及到訓練樣本的數量以及模型在這些樣本上的表現。這意味著用訓練集定義的經驗分佈 $\hat{p}(\boldsymbol{x}, y)$ 替代真實分佈 $p_{data}(\boldsymbol{x}, y)$。我們現在最小化經驗風險，

$$E_{x,y\sim \hat{p}_{data}(x,y)}[L(f(x;\theta), y)] = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta), y^{(i)})$$

其中m是訓練樣本的數量。基於最小化平均訓練誤差的訓練過程稱為經驗風險最小化。我們不是直接優化風險，而是優化經驗風險，並希望風險也顯著降低。

，僅僅最小化經驗風險往往會導致過度擬合，特別是在模型容量很高的情況下，因為模型可能僅僅學會記憶訓練數據，而不是學習到泛化的規律。此外，許多有用的損失函數（例如0-1損失）對於梯度下降這類優化算法來說並不友好，因為它們的導數要麼是零，要麼是未定義的，這使得直接應用梯度下降變得困難。

因此，在深度學習等高容量模型的情境中，經驗風險最小化往往不是一個完全可行的策略。相反，需要採用不同的方法，比如加入正則化項以防止過擬合，或者使用更適合的損失函數，來優化一個與真實想要最小化的風險相近但又不完全相同的量。

代理損失函數和提前停止 (Surrogate Loss Functions and Early Stopping)

在機器學習中，我們經常面臨一個挑戰：我們真正關心的損失函數（如0-1損失，用於衡量分類錯誤）很難直接優化。原因是這些損失函數在輸入維度上的複雜度很高，即使是對簡單的線性分類器，精確地最小化這些損失也是一個挑戰。

面對這個問題，機器學習實踐中常用的方法是選擇一個替代損失函數。這些代理損失函數在優化上更為友好，同時也能夠反映出我們真正想要最小化的損失。例如，負對數似然就是一個常用的代理損失函數，它允許模型估計各個類別的條件概率，從而間接地最小化預期的分類錯誤。

有趣的是，使用這些代理損失函數訓練的模型有時能學到比直接最小化0-1損失更多的信息。例如，在使用對數似然進行訓練時，測試集上的0-1損失可能會在訓練集的0-1損失達到零後還持續下降。這是因為這種訓練方法不僅僅在最小化錯誤，還在提高分類器的穩健性，從而獲

得更有信心和可靠的預測。
此外，機器學習中的訓練過程與傳統的數學優化過程有所不同。在機器學習中，我們通常會在達到某種基於提前停止的收斂標準時終止訓練。這種提前停止標準通常基於在驗證集上測量的真實底層損失函數，比如0-1損失。這意味著訓練可能會在替代損失函數的導數還相對較大時停止，這與傳統優化過程中只在導數非常小時才停止的策略形成對比。
總的來說，代理損失函數和提前停止是機器學習中兩個重要的概念，它們共同幫助機器學習模型有效地學習並避免過度擬合，同時在實際應用中實現更好的泛化能力。

批量和小批量算法 (Batch and Minibatch Algorithms)

機器學習算法中的一個關鍵特點是它們經常將目標函數分解為訓練樣本的總和。這種方法使得優化算法可以基於整個成本函數的一個子集來估計成本函數的期望值，從而計算參數的每次更新。舉個例子，最大似然估計問題可以被分解為對數空間中每個樣本的總和，這等同於最大化由訓練集定義的經驗分佈的期望。
準確計算這個期望是非常昂貴的，因為它需要在整個數據集的每個樣本上評估模型。實際上，我們可以通過從數據集中隨機抽取少量樣本來計算這些期望，並僅對這些樣本取平均。這種方法的效率體現在，當使用更多樣本來估計梯度時，所獲得的回報遠小於線性增長。
在實際應用中，如果選擇基於少量樣本來快速計算梯度的近似估計，而不是慢慢計算精確的梯度，大多數優化算法會更快地收斂。這種方法特別有效，因為訓練集中的許多樣本可能對梯度的貢獻非常相似，甚至在極端情況下，訓練集中所有樣本可能都是完全相同的。
使用整個訓練集的優化算法被稱為批量或確定性梯度方法。這些方法處理一大批訓練樣本，與小批量隨機梯度下降方法形成對比，後者使用的是一小部分樣本。批量梯度下降通常意味著使用整個訓練集，而小批量通常用來描述小批量隨機梯度下降中使用的小批量樣本。
另一方面，一次只使用一個樣本的優化算法有時被稱為隨機方法或在線方法。當樣本是從持續產生的樣本流中抽取的時候，通常會使用“在線”這個術語。
用於深度學習的大多數算法都位於這兩種方法之間，使用的訓練樣本數量多於一個但少於所有。這些方法傳統上被稱為小批量或小批量隨機方法，現在通常簡單地稱為隨機方法。通過使用這些不同的策略，機器學習算法可以在計算效率和模型性能之間找到平衡。

方法	名稱	處理數據	更新參數	特點
批量梯度下降	Batch Gradient Descent	使用整個訓練集計算梯度	每次更新基於所有數據的全局信息	計算成本高，收斂平穩，適用於小數據集
隨機梯度下降	Stochastic Gradient Descent	每次隨機選擇一個樣本計算梯度	更新頻繁，每次僅基於一個樣本	更新快速，但收斂路徑不穩定，適用於大數據集
小批量梯度下降	Mini-Batch Gradient Descent	從訓練集中隨機選取一小批數據計算梯度	每次更新基於小批量數據	平衡了計算效率和收斂穩定性，適用於中大數據集

小批量(minibatch)算法的批量大小選擇受到多個因素的影響。這些因素包括:

1. 梯度估計的準確性:較大的批量可以提供對梯度的更準確估計，但提升的效果並不是線性的。換句話說，隨著批量大小的增加，每增加一個單位的樣本，對梯度估計的改善程度逐漸降低。
2. 多核心架構的利用:現代的硬體，尤其是多核心處理器，通常無法在極小的批量下得到充分利用。這意味著存在一個絕對的最小批量大小，低於此大小不會減少處理小批量的時間。
3. 記憶體限制:當批次中的所有範例並行處理時(這是常見的情況)，所需的記憶體量會隨著批量大小的增加而增加。對於許多硬體配置來說，記憶體容量可能成為限制批量大小的關鍵因素。
4. 硬體效能優化:某些硬體，特別是GPU，在處理特定大小的數組時運行效率更高。常見的是使用2的冪次大小作為批量大小，例如32、64、128或256。對於一些大型模型，有時甚至會嘗試使用16作為批量大小。

小批量訓練有許多特點和優勢，其中包括它們對學習過程的規則化效應。這種效應可能是由於小批量訓練引入了一定的噪聲，特別是當批量大小為1時，通常可以達到最佳的泛化錯誤。然而，使用小批量大小進行訓練可能需要較小的學習率，以保持算法的穩定性。由於對梯度估計的高變異性，這種訓練方法可能需要更多的步驟，從而導致總運行時間變長。

不同的機器學習算法以不同的方式從小批量中提取信息。有些算法對抽樣誤差更敏感。基於梯度$\boldsymbol{g}$ 計算更新的方法通常相對穩健，能夠處理較小的批量大小，如100。而二階方法，這些方法還使用Hessian矩陣$\boldsymbol{H}$ 並計算諸如$\boldsymbol{H}^{-1}\boldsymbol{g}$，通常需要更大的批量大小，如10000，以最小化對估計$\boldsymbol{H}^{-1}\boldsymbol{g}$ 的波動。

在實際應用中，隨機選擇小批量數據是非常重要的。要從一組樣本中計算無偏(unbiased) 的預期梯度估計，需要這些樣本是獨立的。同時，我們希望連續的梯度估計彼此獨立，因此連續的小批量樣本也應該互相獨立。這意味著在數據集的順序具有一定意義的情況下，需要在選擇小批量之前對樣本進行洗牌。在一些非常大的數據集上，每次構建小批量時真正隨機抽取樣本可能是不切實際的。但通常只需將數據集的順序洗牌一次，然後以這種方式存儲即可。

小批量隨機梯度下降的一個有趣動機是，它遵循真實泛化誤差的梯度，只要沒有重複的例子(examples)。通過從數據生成分佈p_{data} 抽取一個小批量樣本$\{x^{(i)}, \ldots, x^{(m)}\}$及其對應目標，我們可以計算該小批量的損失相對於參數的梯度，獲得泛化誤差精確梯度的無偏估計器，並更新模型參數以減少泛化誤差。

當然，這種方法只有在不重用樣本時才適用。然而，通常最好是多次遍歷訓練集，除非訓練集非常大。當使用多個這樣的周期時，只有第一個周期遵循泛化誤差的無偏梯度。但由於降低了訓練誤差，額外的周期通常提供足夠的好處，以抵消訓練誤差和測試誤差之間增加的差距。隨著一些數據集的迅速增長，超過了計算能力的增長速度，機器學習應用中只使用每個訓練樣本一次，甚至只完成對訓練集的不完整遍歷，變得越來越普遍。當使用極大的訓練集時，過擬合不是問題，因此欠擬合和計算效率成為主要關注點。

第二章:優化挑戰和策略

在機器學習領域，一般的優化任務極為困難。傳統上，為了避免這種困難，機器學習通常通過精心設計目標函數和約束條件，確保優化問題保持convex。然而，在訓練神經網絡時，我們不得不面對更複雜的一般non-convex優化問題。

不良系統 (Ill-conditioning)

當優化convex function時，會出現一些挑戰。其中最顯著的是Hessian矩陣$\boldsymbol{H}$的不良條件。這是大多數數值優化中非常普遍的問題，無論是convex的還是其他的。不良條件問題普遍認為存在於神經網絡訓練問題中。不良條件可以可以從SGD陷入困境來表現，即使非常小的步驟也會增加成本函數。

回想一下方程式4.9，成本函數的second-order Taylor series展開預測了梯度下降步驟 $-\epsilon \boldsymbol{g}$ 將增加到成本中。

$$\frac{1}{2}\epsilon^2 \boldsymbol{g}^T \boldsymbol{H}\boldsymbol{g} - \epsilon \boldsymbol{g}^T \boldsymbol{g}$$

當 $\frac{1}{2}\epsilon^2 \boldsymbol{g}^T \boldsymbol{H}\boldsymbol{g}$ 超過 $\epsilon \boldsymbol{g}^T \boldsymbol{g}$ 時，梯度的不良條件成為問題。在許多情況下，梯度範數 $\boldsymbol{g}^T \boldsymbol{g}$ 在整個學習過程中並未顯著縮減，但$\boldsymbol{g}^T \boldsymbol{H}\boldsymbol{g}$項增長超過一個數量級。結果是，儘管存在強烈的梯度，學習變得非常緩慢，因為必須縮小學習率來補償更強的曲率。

儘管在神經網絡訓練之外的其他設置中存在不良條件，但在其他情境中用於對抗它的一些技術對神經網絡來說適用性較低。例如，Newton's method是一種用於最小化 條件較差的Hessian矩陣的convex函數 的絕佳工具，但在應用於神經網絡之前，Newton's method也需要做些修改。

局部最小值 (Local Minima)

局部最小值在convex optimization問題中扮演著關鍵角色。在這種情況下，局部最小值總是全局最小值。然而，在nonconvex function，如神經網絡中，存在多個局部最小值。這些網絡，尤其是那些包含多個等效參數化潛在變量的模型，由於模型可識別性問題而產生多個局部最小值。

一個典型的例子是神經網絡，其中權重空間的對稱性和非線性激活函數如整流線性單元（ReLU）和最大化網絡增加了非可識別性。例如，通過調整神經網絡的權重和偏差，可以創造出效果相同但參數不同的多個網絡配置。這些非可識別性原因導致神經網絡的成本函數可能有大量甚至不可數的局部最小值。

然而，這些由不可識別性產生的局部最小值在成本函數值上是相等的。這意味著，盡管存在多個局部最小值，但它們並不是nonconvex問題的一種有問題的形式。實際上，對於大型神經網絡，大多數局部最小值的成本函數值都相對較低，因此找到真正的全局最小值並不總是必要的。

過去，一些從業者認為局部最小值是神經網絡優化中的一個常見難題。但現在的觀點有所轉變，專家們開始懷疑對於足夠大的網絡，許多局部最小值的成本函數值都足夠低，使得尋找一個成本低但不是最小的點變得更加重要。總之，局部最小值的角色和影響在不同類型的優化問題中有所不同，特別是在涉及覆雜模型如神經網絡時。

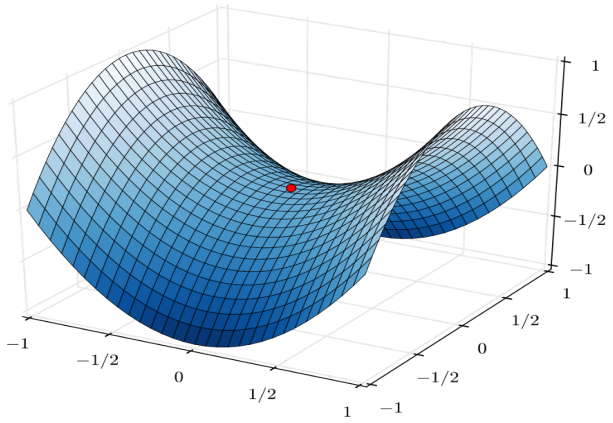
高原、鞍點和其他平坦區域 (Plateaus, Saddle Points, and Other Flat Regions)

在許多高維nonconvex function的研究中，相較於局部極值（最小值或最大值），鞍點這種具有零梯度的點更為常見。鞍點的特點在於其Hessian矩陣包含正負特徵值。在這些點上，沿著正特徵值方向的點比鞍點有更高的成本，而沿著負特徵值方向則有較低的成本。

鞍點可以被理解為成本函數沿某個截面的局部最小值，而沿另一個截面則是局部最大值。隨著維度的增加，局部最小值變得罕見，而鞍點則變得更加普遍。對於高維空間的函數

$f : R^n \to R$

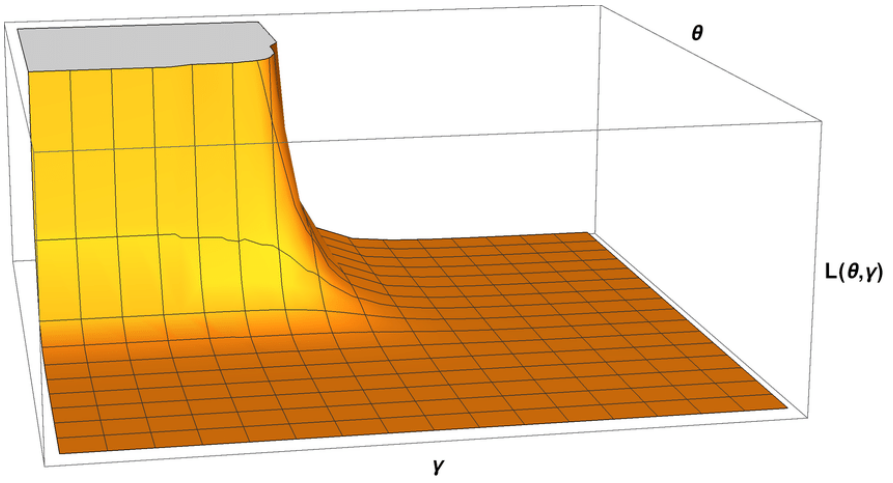
，鞍點與局部最小值的比例隨維度n的增長而呈指數增加。



局部最小值的Hessian矩陣只有正特徵值，而鞍點的Hessian矩陣則是正負特徵值的混合。在高維空間中，所有特徵值全為正或全為負的概率極低，因此鞍點成為主導。此外，許多隨機函數在達到較低成本區域時，其Hessian的特徵值傾向於正值，意味著低成本的局部最小值比高成本更為常見。

對於訓練算法來說，鞍點的存在帶來了新的挑戰。對於僅依賴梯度信息的算法，如何處理鞍點附近的低梯度情況是一個問題。雖然梯度下降算法在許多情況下似乎能夠逃脫鞍點，但對於牛頓法這種設計來解決梯度為零點的算法，鞍點則可能構成問題，因為它可能會跳到鞍點上。這也是為什麼在高維空間的神經網絡訓練中，二階方法並沒有取代梯度下降的原因之一。高維空間中的鞍點和其他平坦區域對於優化算法的影響和挑戰，是一個值得深入研究的領域。

懸崖和梯度爆炸 (Cliffs and Exploding Gradients)



神經網絡中，尤其是多層網絡，經常會出現極為陡峭的區域，這些區域類似於懸崖，主要是由於多個大權重相乘所致。在這些極陡峭的懸崖結構的表面上，梯度更新步驟可能會使參數移動極遠，通常會完全跳過懸崖結構。無論是從上方還是下方接近懸崖，都可能是危險的，但幸運的是，這些嚴重後果可以通過使用梯度裁剪啟發式方法來避免。

當傳統的梯度下降算法提出進行非常大的步驟時，梯度裁剪啟發式方法介入，以減少步長，使跳出梯度指示的大致最陡下降方向的區域變得不太可能。在循環神經網絡的成本函數中，懸崖結構尤為常見，因為這類模型涉及許多因素的乘積，每個時間步都有一個因素。因此，長時間序列會導致極端的乘法累積。

這種懸崖結構的存在對神經網絡的訓練提出了特殊的挑戰，尤其是在控制梯度更新時。梯度裁剪方法的引入，為防止算法在這些陡峭區域中做出過大步伐提供了有效的解決方案，這對於維持神經網絡訓練的穩定性和效率至關重要。

長期依賴 (Long-term Dependencies)

在深度學習領域，神經網絡面臨著梯度消失和爆炸的挑戰，尤其是在計算圖變得極其深入的情況下。前饋網絡和遞歸網絡都是這種問題的典型例子。在遞歸網絡中，反覆使用相同的參數會導致這些問題特別明顯。假設計算圖包含一條重複乘以矩陣W的路徑，隨著時間的推移，這相當於乘以W^t。假設W有一個特徵分解$W = V diag(\lambda) V^{-1}$，則乘以 W^t 可以寫成

$$W^t = \left(V \mathrm{diag}(\boldsymbol{\lambda}) V^{-1}\right)^t = V \mathrm{diag}(\boldsymbol{\lambda})^t V^{-1}.$$

如果W的特徵值不接近絕對值1，則梯度會在大於1時爆炸，或小於1時消失。這導致梯度消失問題，使得難以確定參數該如何調整以改善成本函數；而梯度爆炸則可能導致學習過程不穩定。促使梯度裁剪的懸崖結構正是爆炸性梯度的一個例子。值得注意的是，由於前饋網絡在每個層使用不同的矩陣W，與遞歸網絡相比，它們在一定程度上能避免梯度消失和爆炸問題。

不精確梯度 (Inexact Gradients)

在優化算法的設計中，通常假定我們能夠獲得精確的梯度或海森矩陣。然而，在實際應用中，我們通常只能獲得這些量的噪聲估計或甚至是有偏估計。此外，有時我們想要最小化的目標函數本身就是難以處理的，其梯度也同樣難以確定。在這些情況下，我們只能近似估計梯度。為了應對梯度估計中的不完美，設計了各種神經網絡優化算法。這些算法考慮到梯度估計可能存在的不精確性，並試圖對此進行校正。此外，通過選擇一個更容易近似的替代損失函數來規避問題，也是一種常用策略。這種替代損失函數比真實損失更容易近似，從而使得優化過程更加可行和有效。

總的來說，這些策略的目標是在面對梯度估計的不確定性和計算上的挑戰時，仍然能夠有效地進行神經網絡的優化。這些方法的發展和改進，對於神經網絡在各種複雜問題中的應用至關重要，使得即使在不理想的條件下也能達到良好的優化效果。

局部/全局結構之間的差異 (Poor Correspondence Between Local/Global Structure)

在深度學習的優化過程中，我們面臨許多挑戰，尤其是與單點損失函數的特性相關的挑戰。即使在某點克服問題，如果最佳改進方向不指向低成本區域，模型表現仍可能不佳。許多專家認為，訓練神經網絡耗時的主要原因在於到達解決方案所需的軌跡長度。

優化困難的研究通常集中在訓練是否能達到全局最小點、局部最小點或鞍點。然而，實際情況是，神經網絡往往不會到達任何這些關鍵點。它們通常不會進入梯度較小的區域，而且這樣的關鍵點甚至可能根本不存在。舉例來說，像 $-\log p(y|\boldsymbol{x};\boldsymbol{\theta})$ 這樣的損失函數可能沒有全局最小點，而是隨著模型對其預測變得更有信心，損失函數的值會逐漸接近某一特定值。

對於使用softmax的分類器，當模型能正確分類訓練集中的每個實例時，負對數似然性會非常接近零，但實際上無法達到零。同理，實數值模型如$p(y|\boldsymbol{x}) = \mathrm{N}(y; f(\boldsymbol{\theta}), \beta^{-1})$的負對數似然性理論上可以無限接近負無窮，這發生在模型能夠完美預測所有訓練集目標值的情況下，此時學習演算法會無限制地增加β值。
然而，即使在沒有局部最小值或鞍點的情況下，局部優化有時也無法找到一個好的成本函數值。這通常是由於初始化在“山”的錯誤一側，無法跨越它。雖然在高維空間中學習演算法通常可以繞過這樣的障礙，但這需要的路徑可能非常長，導致訓練時間過長。
未來的研究需要更深入了解影響學習軌跡長度的因素，並更全面地描述學習過程的結果。當前的許多研究方向聚焦於為具有複雜全局結構的問題找到良好的初始點，而非開發能夠進行非局部移動的演算法。

雖然局部下降可能定義出通往有效解的較短路徑，但我們往往無法精準遵循這條路徑。原因在於我們只能大致估計目標函數的某些性質，比如梯度，並且這些估計往往帶有偏見或方差。此外，目標函數本身可能存在問題，例如條件不良或梯度不連續，使得能夠良好模擬目標函數的梯度區域非常有限。
在某些情況下，局部信息可能完全無法提供有效指引，例如當函數處於寬廣平坦的區域，或者當我們落在某個關鍵點上。這些情況下，局部下降無法定義通往解決方案的路徑。有時局部移動過於貪婪，可能會導致我們沿著下坡但偏離解決方案的路徑前進，或者走上一條不必要長的軌跡。
目前，我們還不完全了解哪些問題對於神經網絡優化最為關鍵，這仍是一個活躍的研究領域。不過，如果能夠在一個局部下降能夠有效工作且與解決方案直接相連的區域內初始化學習，那麼許多這類問題都可能被避免。這種觀點促使研究者探索使用傳統優化演算法時選擇良好初始點的重要性。

優化的理論極限 (Theoretical Limits of Optimization)

許多理論研究表明，對於神經網絡設計的任何優化算法都存在性能上的限制。然而，這些理論結果通常對於神經網絡在實際應用中的使用影響不大。

一些理論成果僅適用於神經網絡輸出離散值的情況。而大多數神經網絡單元輸出的是平滑增長的值，這使得通過局部搜索進行優化成為可能。另一些理論結果指出，存在某些無法處理的問題類別，但要判斷特定問題是否屬於這一類別往往很困難。
還有其他理論成果顯示，對於特定大小的網絡找到解決方案是無法實現的。然而，在實踐中，我們通常可以通過使用更大的網絡來輕鬆找到解決方案，因為更大的網絡有更多的參數設置對應於可接受的解決方案。此外，在神經網絡訓練的背景下，我們通常不需要找到函數的精確最小值，只需降低其值以獲得良好的泛化錯誤率。
對於優化算法是否能夠實現這一目標的理論分析極為困難。因此，為優化算法的性能開發更現實的界限仍然是機器學習研究的一個重要目標。這些研究有助於更深入地理解神經網絡的優化過程，並指導實際應用中的算法設計和選擇。