1. 计算

$$\frac{\partial \ln \det A}{\partial x} = \frac{\partial \ln \det A}{\partial A}\frac{\partial A}{\partial x} = (A^{-1})^T \cdot \frac{\partial A}{\partial x}$$

2.习题 1.12

理论极限: $2^{18} = 262144$

编写代码运行结果，三列分别为`k`，假设数量，程序已运行时间，可以看到当$k = 9$时，达到理论极限

```
$ g++ cal.cpp -o cal
$ ./cal
0          1    5e-06s
1         49    6.9e-05s
2        898    0.000367s
3       8386    0.003641s
4      41743    0.033853s
5     115822    0.265686s
6     201304    1.65424s
7     248854    9.11595s
8     260788    46.5398s
9     262144    211.743s
```

```cpp
#include <iomanip>
#include <iostream>
#include <unordered_set>
#pragma GCC optimize(3, "Ofast", "inline")
using namespace std;

unordered_set<long long int> base;
unordered_set<int> counter;
clock_t start, now;

int which_melon[48] = {
    0x1, 0x2, 0x4, 0x7, 0x8, 0x10, 0x20, 0x38, 0x40, 0x80,
    0x100, 0x1c0, 0x49, 0x92, 0x124, 0x1ff, 0x200, 0x400,
    0x800, 0xe00, 0x1000, 0x2000, 0x4000, 0x7000, 0x8000,
    0x10000, 0x20000, 0x38000, 0x9200, 0x12400, 0x24800,
    0x3fe00, 0x201, 0x402, 0x804, 0xe07, 0x1008, 0x2010,
    0x4020, 0x7038, 0x8040, 0x10080, 0x20100, 0x381c0,
    0x9249, 0x12492, 0x24924, 0x3ffff
};

void find_num(int remain, long long int now, int bitslocats, int _ans)
{
    if (remain == 0)
    {
        counter.insert(_ans);
        return;
    }
    else if (remain > 48 - bitslocats)
        return;
    for (int i = bitslocats; i < 48; i++)
        find_num(remain - 1, now | (1ULL << i), i + 1, _ans | which_melon[i]);
```

```cpp
}

int function(int k)
{
    base.clear();
    find_num(k, 0, 0, 0);
    return counter.size();
}

int main()
{
    start = clock();
    for (int k = 0; k <= 18; k++)
    {
        int answer = function(k);
        now = clock();
        double now_time = (double)(now - start) / CLOCKS_PER_SEC;
        cout << k << " " << std::setw(10) << answer << "\t" << now_time << "s" << endl;
        if (answer >= 262144)
            break;
    }
    return 0;
}
```

3.

$$f(x_1,x_2)=\sqrt{\frac{1}{(2\pi)^2\det(\Sigma)}}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

其中

$$\mu=\begin{bmatrix}\mu_{x_1}\\ \mu_{x_2}\end{bmatrix}\qquad \Sigma=\begin{bmatrix}\sigma_{x_1}^2 & \rho\sigma_{x_1}\sigma_{x_2}\\ \rho\sigma_{x_1}\sigma_{x_2} & \sigma_{x_2}^2\end{bmatrix}$$

则

$$f_{x_1}(x_1)=\int_{-\infty}^{+\infty}f(x_1,x_2)dx_2=\frac{1}{\sqrt{2\pi}\sigma_x}e^{-\frac{(x_1-\mu_{x_1})^2}{2\sigma_{x_1}^2}}$$
$$P_{x_1}(x_1)=\int_{-\infty}^{x_1}f_{x_1}(t)dt=\frac{1}{\sqrt{2\pi}}e^{-\frac{x_1^2}{2}}$$

以及

$$P_{x_1|x_2}(x_1)=\frac{1}{\sqrt{2\pi(1-\rho^2)}}e^{-\frac{(x_1-\rho x_2^2)}{2(1-\rho^2)}}$$

4.

令

$$f(x)=\|x\|_p$$

由`Minkowski不等式`

$$f(\lambda x+(1-\lambda)y)\le f(\lambda x)+f((1-\lambda)y)=\lambda f(x)+(1-\lambda)f(y)$$

即$f(x)=\|x\|_p$为凸函数

5.

- 必要性

  因为$f(x)$是凸函数，所以有

$$f(tx+(1-t)y)\le tf(x)+(1-t)f(y)$$

  则有

$$f(y)\ge f(x)+\frac{f(x+(1-t)(y-x))-f(x)}{1-t}\ge f(x)+\nabla f(x)^T(y-x)$$

- 充分性

  令

$$\mu=tx+(1-t)y$$

  则有

$$\begin{aligned}f(x)\ge f(\mu)+\nabla f(t)^T(x-\mu)\\ f(y)\ge f(\mu)+\nabla f(t)^T(y-\mu)\end{aligned}$$

  分别乘以 $t,1-t$，则有

$$tf(x)+(1-t)f(y)\ge tf(\mu)+(1-t)f(\mu)=f(\mu)$$

  即

$$tf(x)+(1-t)f(y)\ge f(tx+(1-t)y)$$

习题 2.2

- 10 折交叉验证：训练集比例是均匀的，测试集同样，最后错误概率为 50%
- 留一法：只有两种情况，留下来正例，则训练集反例多，反之同理，最后错误的概率都为 100%

习题 2.4

表 2.1 分类结果混淆矩阵

真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

$$查准率\ P=\frac{TP}{TP+FP}$$

$$查全率\ R=\frac{TP}{TP+FN}$$

$$真正例率\ TPR=查全率$$

$$假正例率\ FPR=\frac{FP}{TN+FP}$$

习题 2.5

`AUC` 是 `ROC` 曲线下面的面积，有水平竖直和倾斜两种

$AUC=\frac{1}{2}\sum_{i=1}^{m-1}(x_{i+1}-x_i)(y_{i+1}+y_i)$

$l_{rank}=\frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}[\mathbb{I}(f(x^+)<f(x^-))+\frac{1}{2}\mathbb{I}(f(x^+)=f(x^-))]$

如果把横坐标定位当前反例的个数m^-，纵坐标定为当前正例的个数m^+，`AUC` 则是，当前样本坐标(x,y)，下一个样本坐标为正例则为$(x,y+1)$，反之则为$(x+1,y)$或$(x+\frac{1}{2},y+\frac{1}{2})$，此时将横纵坐标缩小$m^-m^+$，作归一化处理，可得:

$AUC=\frac{1}{m^+m^-}\sum_{x^+\in D^+}\sum_{x^-\in D^-}[\mathbb{I}(f(x^+)>f(x^-))+\frac{1}{2}\mathbb{I}(f(x^+)=f(x^-))]$

则$AUC+l_{rank}=1$

习题 2.9

卡方检验分为拟合优度检验和卡方独立性检验

$$\chi^2=\sum\frac{(O-E)^2}{E}$$

O 为实际频数，*E* 为期望频数

拟合优度检验中的自由度为$分类变量数-1$,独立性检验的自由度为$(行数-1)\times(列数-1)$

通过计算χ^2的值来评估拟合优度或独立性