libsvm code

SMO 中变量α的选择

```
int Solver::select_working_set(int &out_i, int &out_j)
{
        double Gmax = -INF;//m(alpha) =max i in I_up -y_i deltaf(alpha)
        double Gmax2 = -INF;//M(alpha) =min i in I_low -y_j deltaf(alpha)
        int Gmax_idx = -1;
        int Gmin_idx = -1;
        double obj_diff_min = INF;

        for(int t=0;t<active_size;t++)//更新GMAX,m(alpha)
                if(y[t]==+1)
                {
                        if(!is_upper_bound(t))//y-t=1 and a_t<C
                                if(-G[t] >= Gmax)
                                {
                                        Gmax = -G[t];
                                        Gmax_idx = t;
                                }
                }
                else
                {
                        if(!is_lower_bound(t))//y_t=-1 and a_t>0
                                if(G[t] >= Gmax)
                                {
                                        Gmax = G[t];
                                        Gmax_idx = t;
                                }
                }
        int i = Gmax_idx;
        const Qfloat *Q_i = NULL;
        if(i != -1)
                Q_i = Q->get_Q(i,active_size);

        for(int j=0;j<active_size;j++)//更新GMAX2 M(alpha)
        {
                if(y[j]==+1)
                {
                        if (!is_lower_bound(j))
                        {//y_j=1 and a_j>0
                                double grad_diff=Gmax+G[j];
```

```
            if (G[j] >= Gmax2)   //求 max -M (alpha)
                Gmax2 = G[j];
            if (grad_diff > 0)//-y_j delta f(alpha) <-y_i delta f(a
            {
                double obj_diff;
                double quad_coef = QD[i]+QD[j]-2.0*y[i]*Q_i[j];
                if (quad_coef > 0)//kii+kjj-2kij>0
                    obj_diff = -(grad_diff*grad_diff)/quad_
                else
                    obj_diff = -(grad_diff*grad_diff)/TAU;

                if (obj_diff <= obj_diff_min)//取可以取到最小值的j
                {
                    Gmin_idx=j;
                    obj_diff_min = obj_diff;
                }
            }
        }
    }
    else
    {
        if (!is_upper_bound(j))
        {//y_j=-1 and a_j>0
            double grad_diff= Gmax-G[j];
            if (-G[j] >= Gmax2)
                Gmax2 = -G[j];
            if (grad_diff > 0)
            {
                double obj_diff;
                double quad_coef = QD[i]+QD[j]+2.0*y[i]*Q_i[j];
                if (quad_coef > 0)
                    obj_diff = -(grad_diff*grad_diff)/quad_
                else
                    obj_diff = -(grad_diff*grad_diff)/TAU;

                if (obj_diff <= obj_diff_min)
                {
                    Gmin_idx=j;
                    obj_diff_min = obj_diff;
                }
            }
        }
    }
}

if(Gmax+Gmax2 < eps || Gmin_idx == -1)// 停止条件
    return 1;

out_i = Gmax_idx;//寻找到的 SMO 的更新的「i, j」
out_j = Gmin_idx;
return 0;
```

```
}
```

变量$alpha$的更新

```
// 更新 alpha[i] and alpha[j]
                const Qfloat *Q_i = Q.get_Q(i,active_size);
                const Qfloat *Q_j = Q.get_Q(j,active_size);

                double C_i = get_C(i);
                double C_j = get_C(j);

                double old_alpha_i = alpha[i];
                double old_alpha_j = alpha[j];

                if(y[i]!=y[j])//y_i!=y_j
                {
                        double quad_coef = QD[i]+QD[j]+2*Q_i[j];
                        if (quad_coef <= 0)
                                quad_coef = TAU;
                        double delta = (-G[i]-G[j])/quad_coef;
                        double diff = alpha[i] - alpha[j];
            //更新
                        alpha[i] += delta;
                        alpha[j] += delta;

            //修正   4种需要修正的情况
                        if(diff > 0)
                        {
                                if(alpha[j] < 0)//region 3
                                {
                                        alpha[j] = 0;
                                        alpha[i] = diff;
                                }
                        }
                        else
                        {
                                if(alpha[i] < 0)//region 4
                                {
                                        alpha[i] = 0;
                                        alpha[j] = -diff;
                                }
                        }
```

```
		if(diff > C_i - C_j)//region 1
		{
			if(alpha[i] > C_i)
			{
				alpha[i] = C_i;
				alpha[j] = C_i - diff;
			}
		}
		else
		{
			if(alpha[j] > C_j)////region 2
			{
				alpha[j] = C_j;
				alpha[i] = C_j + diff;
			}
		}
	}

	else//同理y_i==y_j 的情况
	{
		double quad_coef = QD[i]+QD[j]-2*Q_i[j];
		if (quad_coef <= 0)
			quad_coef = TAU;
		double delta = (G[i]-G[j])/quad_coef;
		double sum = alpha[i] + alpha[j];
		alpha[i] -= delta;
		alpha[j] += delta;

		if(sum > C_i)
		{
			if(alpha[i] > C_i)
			{
				alpha[i] = C_i;
				alpha[j] = sum - C_i;
			}
		}
		else
		{
			if(alpha[j] < 0)
			{
				alpha[j] = 0;
				alpha[i] = sum;
			}
		}
		if(sum > C_j)
		{
			if(alpha[j] > C_j)
			{
				alpha[j] = C_j;
				alpha[i] = sum - C_j;
			}
```

```
                        }
                        else
                        {
                                if(alpha[i] < 0)
                                {
                                        alpha[i] = 0;
                                        alpha[j] = sum;
                                }
                        }
                }
```

变量G的更新

```
// update G

                double delta_alpha_i = alpha[i] - old_alpha_i;
                double delta_alpha_j = alpha[j] - old_alpha_j;

                for(int k=0;k<active_size;k++)//只需要更新alpha_i 和alpha_j(其他的都没有改变
                {
                        G[k] += Q_i[k]*delta_alpha_i + Q_j[k]*delta_alpha_j;
                }
```