1-1

Generative: در این روش اول توزیع احتمالاتی هر کلاس را تخمین میزنیم و سپس تابع discriminant را حساب میکنیم.

Discriminant: در این روش صریحا تابع discriminant را حساب میکنیم و نیاز به در نظر گرفتن توزیع احتمالاتی نیست.

1-2

وقتی تابع Discriminant خطی باشد و با معادله‌ی زیر داده شده باشد، داریم:

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0,$$

جهت ابر صفحه با بردار نرمال w و موقعیت آن با بایاس w0مشخص می‌شود.

1-3

در واقع برای حل مسائل یادگیری، آن‌ها را به مسئله‌ی کاهش تابع هزینه تبدیل میکنیم و با تعریف مناسب این تابع، معیار مناسبی برای سنجش یادگیری داریم.

1-4

فرض میکنیم c کلاس داریم.

یکی در مقابل همه: در این روش به مساله به چشم c مساله‌ی دو کلاسه نگاه میکنیم و برای هر کلاسی به تابعی دست میابیم تا طبقه‌بندی را انجام دهیم. مشکل این روش وجود نواحی مبهم است. نواحی ای که هیچکدام از توابع ما آنرا جزو دسته‌ی خود در نظر نمیگیرند.

یکی در مقابل دیگری:

در این روش به ازای هر دو کلاس یک تابع داریم و در کل c * (c -1) / 2 تابع داریم. مشکل این روش نیز وجود نواحی مبهم است.

ماشین خطی:

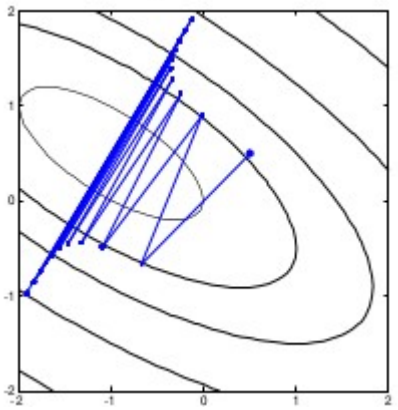
در این روش c تابع خطی discriminant داریم که برای هر نقطه هر کدام از تابع‌ها امتیازی حساب کرده و با مقایسه‌ی امتیازها طبقه‌بندی صورت میگیرد.

1-5

ماشین خطی قسمت قبل

1-6

در صورتی که نرخ یادگیری مقدار بالایی داشته باشد در هر آپدیت الگوریتم گام‌هاب بلندتری برداشته و حتی ممکن است اندازه‌ی گام از فاصله نقطه‌ی حاضر از نقطه‌ی کمینه بیشتر باشد و مدل اطراف نقطه‌ی بهینه در حال حرکت باشد و هرگز همگرا نشود.



1-7

وابسته به معکوس‌پذیری ماتریس هسیان است. همچنین این پروسه بسیار زمانگیر است و از نظر زمانی نسبت به تعداد ویژگی‌ها درجه سوم است.

2-1

با استفاده از کرنل خطی SVM به نتایج زیر رسیدیم.

(0 = جوان، 1 = میانسال، 2=پیر)

	0	1	2
0	55	16	1
1	2	124	0
2	13	39	0

	precision	recall	f1-score	support
0	0.79	0.76	0.77	72
1	0.69	0.98	0.81	126
2	0.00	0.00	0.00	52
accuracy			0.72	250
macro avg	0.49	0.58	0.53	250
weighted avg	0.58	0.72	0.63	250

متاسفانه در این حالت مدلمان هیچ پیشبینی ای برای کلاس پیر نداشته است. دلیل این مشکل یکی نبودن مقیاس فیچرهای مختلف است که در محاسبه‌ی فاصله‌ها، فیچرهای با مقیاس بزرگتر نقش اصلی را ایفا خواهند کرد.

2-2

کرنل rbf نیز تقریبا به نتایج مشابهی رسید با این تفاوت که از نظر زمانی بسیار سریع‌تر از حالت قبل بود. (kernel tricks)

	0	1	2
0	54	18	0
1	3	123	0
2	13	39	0

	precision	recall	f1-score	support
0	0.77	0.75	0.76	72
1	0.68	0.98	0.80	126
2	0.00	0.00	0.00	52
accuracy			0.71	250
macro avg	0.48	0.58	0.52	250
weighted avg	0.57	0.71	0.62	250

2-3

پس از نرمالایز کردن دیگر نه اندازه و مقیاس فیچرها بلکه اطلاعات هر فیچر است که در یادگیری تاثیرگذار میشود.

برای کرنل خطی داریم:

	0	1	2
0	53	10	9
1	8	108	10
2	9	18	25

	precision	recall	f1-score	support
0	0.76	0.74	0.75	72
1	0.79	0.86	0.82	126
2	0.57	0.48	0.52	52
accuracy			0.74	250
macro avg	0.71	0.69	0.70	250
weighted avg	0.74	0.74	0.74	250

که به وضوح شرایط در مورد کلاس سوم بهبود یافته است.

همچنین کرنل rbf:

	0	1	2
0	53	10	9
1	8	108	10
2	9	18	25

	precision	recall	f1-score	support
0	0.64	0.71	0.67	72
1	0.71	0.81	0.76	126
2	0.65	0.33	0.44	52
accuracy			0.68	250
macro avg	0.67	0.61	0.62	250
weighted avg	0.68	0.68	0.66	250

-6

در حالت one vs one، مدل هیچ خطایی ندارد.

	0	1	2
0	11	0	0
1	0	11	0
2	0	0	16

Confusion Matrix

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	11
2	1.00	1.00	1.00	16
accuracy			1.00	38
macro avg	1.00	1.00	1.00	38
weighted avg	1.00	1.00	1.00	38

Classification Report

```
[[[27  0]
  [ 0 11]]

 [[27  0]
  [ 0 11]]

 [[22  0]
  [ 0 16]]]
```

Multi label Confusion Matrix

1-1

Generative: در این روش اول توزیع احتمالاتی هر کلاس را تخمین میزنیم و سپس تابع discriminant را حساب میکنیم.

Discriminant: در این روش صریحا تابع discriminant را حساب میکنیم و نیاز به در نظر گرفتن توزیع احتمالاتی نیست.

1-2

وقتی تابع Discriminant خطی باشد و با معادله‌ی زیر داده شده باشد، داریم:

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0,$$

جهت ابر صفحه با بردار نرمال w و موقعیت آن با بایاس w0مشخص می‌شود.

1-3

در واقع برای حل مسائل یادگیری، آن‌ها را به مسئله‌ی کاهش تابع هزینه تبدیل میکنیم و با تعریف مناسب این تابع، معیار مناسبی برای سنجش یادگیری داریم.

1-4

فرض میکنیم c کلاس داریم.

یکی در مقابل همه: در این روش به مساله به چشم c مساله‌ی دو کلاسه نگاه میکنیم و برای هر کلاسی به تابعی دست میابیم تا طبقه‌بندی را انجام دهیم. مشکل این روش وجود نواحی مبهم است. نواحی ای که هیچکدام از توابع ما آنرا جزو دسته‌ی خود در نظر نمیگیرند.

یکی در مقابل دیگری:

در این روش به ازای هر دو کلاس یک تابع داریم و در کل c * (c -1) / 2 تابع داریم. مشکل این روش نیز وجود نواحی مبهم است.

ماشین خطی:

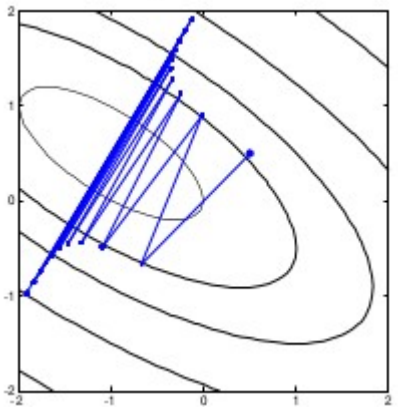
در این روش c تابع خطی discriminant داریم که برای هر نقطه هر کدام از تابع‌ها امتیازی حساب کرده و با مقایسه‌ی امتیازها طبقه‌بندی صورت میگیرد.

1-5

ماشین خطی قسمت قبل

1-6

در صورتی که نرخ یادگیری مقدار بالایی داشته باشد در هر آپدیت الگوریتم گام‌هاب بلندتری برداشته و حتی ممکن است اندازه‌ی گام از فاصله نقطه‌ی حاضر از نقطه‌ی کمینه بیشتر باشد و مدل اطراف نقطه‌ی بهینه در حال حرکت باشد و هرگز همگرا نشود.



1-7

وابسته به معکوس‌پذیری ماتریس هسیان است. همچنین این پروسه بسیار زمانگیر است و از نظر زمانی نسبت به تعداد ویژگی‌ها درجه سوم است.

2-1

با استفاده از کرنل خطی SVM به نتایج زیر رسیدیم.

(0 = جوان، 1 = میانسال، 2=پیر)

	0	1	2
0	55	16	1
1	2	124	0
2	13	39	0

	precision	recall	f1-score	support
0	0.79	0.76	0.77	72
1	0.69	0.98	0.81	126
2	0.00	0.00	0.00	52
accuracy			0.72	250
macro avg	0.49	0.58	0.53	250
weighted avg	0.58	0.72	0.63	250

متاسفانه در این حالت مدلمان هیچ پیشبینی ای برای کلاس پیر نداشته است. دلیل این مشکل یکی نبودن مقیاس فیچرهای مختلف است که در محاسبه‌ی فاصله‌ها، فیچرهای با مقیاس بزرگتر نقش اصلی را ایفا خواهند کرد.

2-2

کرنل rbf نیز تقریبا به نتایج مشابهی رسید با این تفاوت که از نظر زمانی بسیار سریع‌تر از حالت قبل بود. (kernel tricks)

	0	1	2
0	54	18	0
1	3	123	0
2	13	39	0

	precision	recall	f1-score	support
0	0.77	0.75	0.76	72
1	0.68	0.98	0.80	126
2	0.00	0.00	0.00	52
accuracy			0.71	250
macro avg	0.48	0.58	0.52	250
weighted avg	0.57	0.71	0.62	250

2-3

پس از نرمالایز کردن دیگر نه اندازه و مقیاس فیچرها بلکه اطلاعات هر فیچر است که در یادگیری تاثیرگذار میشود.

برای کرنل خطی داریم:

	0	1	2
0	53	10	9
1	8	108	10
2	9	18	25

	precision	recall	f1-score	support
0	0.76	0.74	0.75	72
1	0.79	0.86	0.82	126
2	0.57	0.48	0.52	52
accuracy			0.74	250
macro avg	0.71	0.69	0.70	250
weighted avg	0.74	0.74	0.74	250

که به وضوح شرایط در مورد کلاس سوم بهبود یافته است.

همچنین کرنل rbf:

	0	1	2
0	53	10	9
1	8	108	10
2	9	18	25

	precision	recall	f1-score	support
0	0.64	0.71	0.67	72
1	0.71	0.81	0.76	126
2	0.65	0.33	0.44	52
accuracy			0.68	250
macro avg	0.67	0.61	0.62	250
weighted avg	0.68	0.68	0.66	250

-6

در حالت one vs one، مدل هیچ خطایی ندارد.

	0	1	2
0	11	0	0
1	0	11	0
2	0	0	16

Confusion Matrix

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	11
2	1.00	1.00	1.00	16
accuracy			1.00	38
macro avg	1.00	1.00	1.00	38
weighted avg	1.00	1.00	1.00	38

Classification Report

```
[[[27  0]
  [ 0 11]]

 [[27  0]
  [ 0 11]]

 [[22  0]
  [ 0 16]]]
```

Multi label Confusion Matrix

در حالت one vs rest، خطا وجود دارد.

	0	1	2
0	11	0	0
1	0	11	0
2	0	1	15

Confusion Matrix

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	0.92	1.00	0.96	11
2	1.00	0.94	0.97	16
accuracy			0.97	38
macro avg	0.97	0.98	0.97	38
weighted avg	0.98	0.97	0.97	38

Classification Report

```
[[[27  0]
  [ 0 11]]

 [[26  1]
  [ 0 11]]

 [[22  0]
  [ 1 15]]]
```

Multi label Confusion Matrix

خطای موجود مربوط به پیشبینی یک گیاه از کلاس سوم به عنوان کلاس دوم است. این خطا احتمالا به علت تعداد کمتر طبقه‌بندهای دودویی است و اینکه در حالت قبل چون هر دیتا میان هر دو جفت از کلاس‌ها چک می‌شد و سپس نتیجه‌گیری میشد.

-7

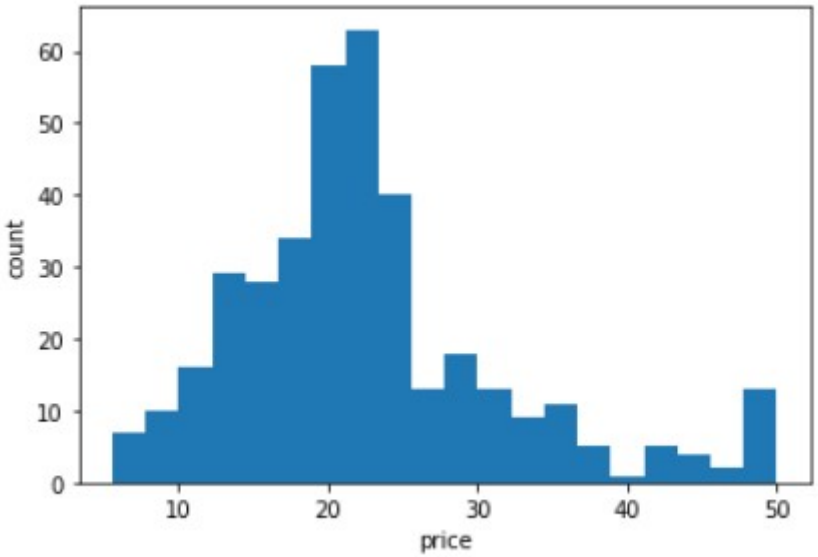
با استفاده از SVR با کرنل خطی که بهترین نتیجه را نیز داشته است برای دیتاهای عادی داشتیم:

```
rmse: 5.461468081706728
mae: 3.4141732933560305
```

و برای دیتاهای نرمالایز شده با مقداری بهبود داشتیم:

```
rmse: 5.1577021512426136
mae: 3.266267141067641
```

که با توجه به توزیع قیمت‌ها خطای مناسبی است.



اگر به جای Hard Margin مقداری خطا را تحمل کرده و از روش Soft Margin استفاده کنیم و متغیر ξ را به مقدار بگیریم که نشان دهنده میزان تحمل خطاست به جای نامعادلات قبلی داریم:

$$\begin{cases} w^T x_i + b \geq 1 - \xi_i; & y_i = 1 \\ w^T x + b \leq -1 + \xi_i; & y_i = -1 \\ \xi_i \geq 0 & \forall i \end{cases}$$

حال با کمینه کردن تابع زیر می‌توان به w های مناسب رسید

$$\frac{1}{2}|w^2| + \frac{C}{2}\sum_i \xi_i^2$$

که در آن C میزان تحمل خطا را کنترل می‌کند. به این صورت که با افزایش C خطاهای کمتری را تحمل کنیم. توان دو ξ و ضریب $\frac{1}{2}$ نیز پس از مشتق گیری محاسبات را آسان می‌کند.

④ سه نقطه در دو کلاس داریم

1: (0.5, 1) , (1, 0.5)

-1: (1, 1.5)

می خواهیم ضریب لاگرانژ هر نقطه را با شرایط $\alpha_i > 0$, $\sum_{i=1}^{n} \alpha_i y_i = 0$ به دست آوریم. با توجه به لیبل داده ها داریم

$\alpha_1 + \alpha_2 - \alpha_3 = 0$

حال با توجه به محدودیت های بالا باید تابع زیر را بیشینه کرد.

$$\alpha^T 1_n - \frac{1}{2} \alpha^T Y G Y \alpha$$

$$\Longrightarrow [\alpha_1 \quad \alpha_2 \quad \alpha_3] \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} - \frac{1}{2} [\alpha_1 \quad \alpha_2 \quad \alpha_3] \begin{bmatrix} y_1 & 0 & 0 \\ 0 & y_2 & 0 \\ 0 & 0 & y_3 \end{bmatrix} G \begin{bmatrix} y_1 & 0 & 0 \\ 0 & y_2 & 0 \\ 0 & 0 & y_3 \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix}$$

$$G = \begin{bmatrix} x_1^T x_1 & x_1^T x_2 & x_1^T x_3 \\ x_2^T x_1 & x_2^T x_2 & x_2^T x_3 \\ x_3^T x_1 & x_3^T x_2 & x_3^T x_3 \end{bmatrix}$$ همچنین برای G داریم

$$G = \begin{bmatrix} 1.25 & 1 & 2 \\ 1 & 1.25 & 1.75 \\ 2 & 1.75 & 3.25 \end{bmatrix}$$

با جایگذاری داریم:

$\Longrightarrow = \alpha_1 + \alpha_2 + \alpha_3 - \frac{5}{8}\alpha_1(\alpha_1 + 2\alpha_2 - 4\alpha_3) + 1.25\alpha_2^2 + 3.25$

$$= \alpha_1 + \alpha_2 - \alpha_3 - \frac{1}{2} [\alpha_1 \quad \alpha_2 \quad -\alpha_3] G \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ -\alpha_3 \end{bmatrix}$$

$\Longrightarrow \alpha_1 = 4 \quad \alpha_2 = 0 \quad \alpha_3 = 4$

$$w = \sum \alpha_i y_i x_i = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$$

$$y_i = w^T x_i + b \xrightarrow{x_1} 1 = \begin{bmatrix} -2 & -2 \end{bmatrix} \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} + b$$

$$\Longrightarrow b = 4$$

$$\boxed{-2x_1 + -2x_2 + 4 = 0}$$

معادله خط جداساز