دانشکده‌ی مهندسی کامپیوتر

یادگیری ماشین

نیم‌سال اول ۱۴۰۳-۱۴۰۲

مدرس: دکتر سید ابوالفضل مطهری

درس‌نامه سوم

Naive Bayes

Naive Bayes Classification یک گروه از دسته‌بندی‌های بر پایه احتمالات است که به صورت زیر عمل می‌کند.

$$\underset{k}{\operatorname{argmax}} \left\{ \underbrace{P\left(y=k|x\right)}_{\text{MAP Rule}} \right\}$$

$$P\left(y=k|x\right) = \frac{P\left(x|y=k\right)P\left(y=k\right)}{P\left(x\right)}$$

در این شرایط تخمین عبارت $P\left(y=k\right)$ ساده است اما تخمین عبارت $P\left(x|y=k\right)$ به دلیل ابعاد بالای ماتریس سخت است. برای ساده‌سازی این عبارت اگر مقادیر x از هم مستقل باشند می‌توان عبارت بالا را به صورت زیر نوشت:

$$P\left(x|y=k\right) = \prod P\left(x_i|y=k\right)$$

لازم به ذکر است که اگر y جزو داده‌ها باشد، xها از یکدگیر مستقل هستند. در غیر این صورت مستقل نخواهند بود و وابسته‌اند چون در یک سمت مشترکند و وابستگی دارند.
در نهایت گروه‌بندی بر اساس yهای متفاوت انجام می‌گیرد.

$$\hat{P}\left(x_۱|y=۱\right),\ \hat{P}\left(x_۱|y=۲\right),\ldots,\ \hat{P}\left(x_۱|y=n\right)$$

Logistic Regression

در این قسمت می‌خواهیم جواب نهایی($P\left(y=k|x\right)$) را حدس بزنیم.

دو کلاسه:

فرض می‌کنیم که مرز بین نواحی خطی است و با یک hyperplane می‌توان فضا را به دو قسمت تبدیل کرد.
می‌خواهیم $P\left(y=۱|x\right) \lessgtr P\left(y=۲|x\right)$ مقایسه کنیم.

$$\frac{P\left(y=۱|x\right)}{P\left(y=۲|x\right)} \leqslant ۱ \Longrightarrow \log\left(\frac{P\left(y=۱|x\right)}{P\left(y=۲|x\right)}\right) \leqslant \log\left(۱\right)$$

اگر حاصل لگاریتم اول برابر با $\beta^T x$ بود، می‌توانستیم این مقدار را با صفر مقایسه کنیم و در نتیجه عبارت بالا را به صورت زیر بازنویسی می‌کنیم.

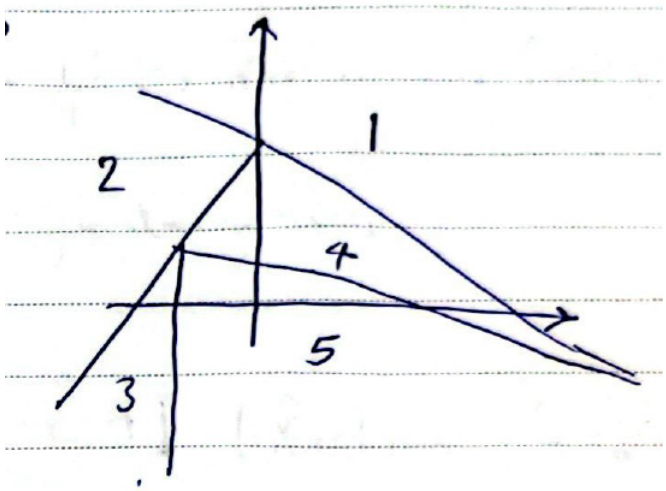
$$P\left(y=۱|x\right)=\frac{\mathrm{e}^{\beta^T x}}{۱+\mathrm{e}^{\beta^T x}}, \qquad P\left(y=۲|x\right)=\frac{۱}{۱+\mathrm{e}^{\beta^T x}}$$

حال با استفاده از تخمینگر MLE مقدار β را تخمین می‌زنیم.

$$\begin{aligned}
P\left(D|\beta\right) &= \prod P\left(y_i|x_i,\beta\right) \Longrightarrow \sum \log\left(P\left(y_i|x_i,\beta\right)\right) = \\
&= \sum \log\left(\frac{\mathrm{e}^{\beta^T x^{(i)}}}{۱+\mathrm{e}^{\beta^T x^{(i)}}}\right) + \sum \log\left(\frac{۱}{۱+\mathrm{e}^{\beta^T x^{(i)}}}\right) \\
&\Longrightarrow \sum \left\{ y_i \log\left(\frac{\mathrm{e}^{\beta^T x^{(i)}}}{۱+\mathrm{e}^{\beta^T x^{(i)}}}\right) + \left(۱-y_i\right) \log\left(\frac{۱}{۱+\mathrm{e}^{\beta^T x^{(i)}}}\right) \right\}
\end{aligned}$$

با استفاده از مشتق، می‌توان به جواب بهینه برای مسئله بالا رسید.

چندکلاسه



برای یک مسئله kکلاسه نیاز به $k-۱$ hyperplane داریم که بتوان فضا را به k قسمت تقسیم کرد.
با مقایسه کردن هر دو فضا با یکدیگر میخواهیم ضرایب β را بدست آوریم.

$$P\left(y=i|x\right) \leqslant P\left(y=j|x\right) \Longrightarrow \log\left(\frac{P\left(y=i|x\right)}{P\left(y=j|x\right)}\right) \leqslant ۰ \Longrightarrow \beta_{ij}^T x \leqslant ۰$$

اگر در این مرحله $y = i$ را با $y = k$ مقایسه کنیم، این مدل consistent می‌باشد. یا به عبارتی دیگر یک گروه را مرجع در نظر بگیریم و بقیه را نسبت به آن مقایسه کنیم.

$$P(y = ۱|x) = \frac{e^{\beta_۱ x}}{z},\ P(y = ۲|x) = \frac{e^{\beta_۲ x}}{z}, \ldots, P(y = n|x) = \frac{۱}{z}$$

$$\text{Where} \qquad z = ۱ + \sum_{i=۱}^{k-۱} e^{\beta_i x}$$

سوال: اگر ۱۰ گروه و ۱۰۰ بعد داشته باشیم. چند پارامتر خواهیم داشت؟

۹ hyperplane خواهیم داشت که هر کدام به ۱۰۱ پارامتر نیاز خواهند داشت.

$$۹ \times ۱۰۱$$

تعریف ۱ (آنتروپی).

$$H((p_۱, \ldots, p_k)) = -\sum p_i \log(p_i)$$

تعریف ۲ (دیورژانس).

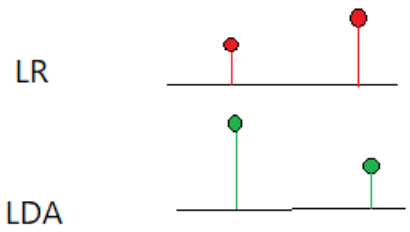
$$D((p_۱, \ldots, p_k) \,||\, (q_۱, \ldots, q_k)) = \sum p_i \log\left(\frac{p_i}{q_i}\right)$$

مقایسه مدل‌ها

ارزیابی مدل یک مرحله مهم در تعیین عملکرد بهینه و همچنین یک معیار تاثیرگذار در انتخاب مدل مورد نظر می‌باشد. به طور مثال فرض کنید که برای یک ست داده می‌خواهیم تصمیم بگیریم که از کدام یک از دو روش LDA و یا $Logistic\ Regression$ استفاده کنیم.اگر فرض کنیم که حجم داده‌ها به اندازه کافی زیاد است، می‌توان به صورت زیر عمل کرد.

ابتدا داده‌ها را به ۴ قسمت تقسیم می‌کنیم. سپس از $\frac{۱}{۴}$ داده‌ها به ترتیب برای $train$ کردن مدل‌های LDA و $Logistic\ Regression$ (توجه کنید که برای هر مدل از یک دسته داده مجزا استفاده می‌شود) و از $\frac{۱}{۴}$ داده‌ها برای $test$ کردن این مدل‌ها استفاده می‌کنیم. (در نظر داشته باشید که بین این ۴ دسته از داده‌ها اشتراکی وجود ندارد. لزوما مدلی که در $train$ نتیجه بهتری داشته باشد مدل بهتری نیست.)

آنها را با یکدیگر مقایسه کنیم. برای مثال اگر نتیجه حاصل مشابه زیر باشد ما مدل LDA را انتخاب میکنیم زیرا میتوان مشاهده کرد که مدل دوم درواقع دارای پیچیدگی بیشتری نسبت به پیچیدگی داده های ما هست و عملا خطای کمتر $train$ گمراه کننده است.



شکل ۱: مثال نتیجه مقایسه دو مدل

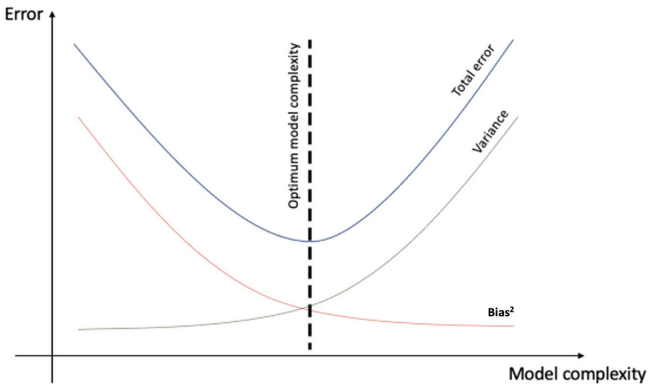
می‌توان از تعمیم دادن این ایده به تعداد بیشتری مدل استفاده کرد. در حالت کلی با توجه به ضرایب β و تعداد پارامترهای به کار رفته می‌توانیم بی‌نهایت مدل مختلف داشته باشیم و خطای مدل‌ها را به صورت زیر محاسبه کنیم:

$$M_1 \rightarrow \frac{1}{n}\sum L(y_i, f_1(x_i))$$

$$M_2 \rightarrow \frac{1}{n}\sum L(y_i, f_2(x_i))$$

$$\vdots$$

این روش زمانی به پاسخ صحیح خواهد رسید و برای تمامی M ها تخمین خوبی ارائه خواهد داد که تعداد داده ها زیاد باشد. نموداری از خطای $test$ و $train$ برحسب $complexity$ تشکیل می‌دهیم. بنابراین در نهایت مدلی مد نظر ما است که هم در $train$ و هم در $test$ خطای کمی داشته باشد و عملکرد بهتر در تنها یکی از این بخش‌ها کافی نیست. در این مقایسه ممکن است پیچیدگی مدل در مقابل عملکرد آن قرار بگیرد. با افزایش پیچیدگی مدل می‌توان در داده‌های $train$ به نتایج بهتری دست یافت (خطای بایاس را کاهش می‌دهیم) اما همین افزایش پیچیدگی می‌تواند در داده‌های $test$ به نتایج ضعیفتر منجر شود (خطای واریانس افزایش می‌یابد). در حقیقت در چنین حالتی، ما از مدلی استفاده کرده‌ایم که از مدل واقعی پیچیده‌تر بوده و به اصطلاح $Overfit$ رخ داده است.



شکل ۲: نمودار دقت مدل با توجه به پیچیدگی آن

به همین منظور واجب است که در مدل خود پیچیدگی را کنترل کنیم. به متغیری که از آن به منظور کنترل پیچیدگی مدل استفاده می‌کنیم $hyperparameter$ می‌گویند و آن را با نماد α نمایش می‌دهیم. به طور مثال مدل

خطی زیر را در نظر بگیرید:

$$f_\beta = \sum_{i=۰}^{p} \beta_i X_i$$

$$s.t. \quad \sum_{i=۰}^{p} \beta_i^۲ \leqslant \alpha$$

در اینجا با استفاده از متغیر α می‌توانیم پیچیدگی مدل را کنترل کنیم به طوری که با افزایش α پیچیدگی افزایش و با کاهش آن، کاهش می‌یابد. توجه کنید که برای داده‌های *typical* مقدار α تک پارامتری است و می‌توان از تعدادی از نمونه‌ها به عنوان داده *train* استفاده کرد، در حالی که اگر ابعاد α زیاد باشد، ممکن است چنین کاری عملی نباشد.

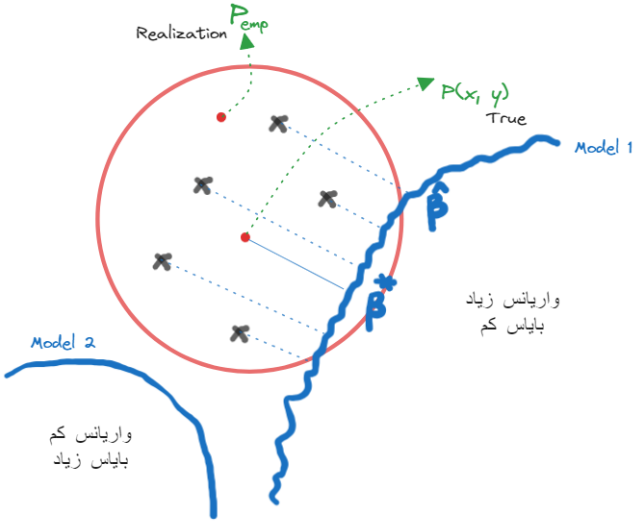
Variance and Bias

$$x : F(x)$$
$$[x_۱, x_۲, ..., x_m]$$
$$F_{emp}(x) = \frac{\sum_{i=۱}^{m} ۱[x_i \leqslant x]}{m}$$
$$f_{emp}(x) = \frac{۱}{m} \sum_{i=۱}^{m} \delta(x - x_i)$$

حال سوال مهمی پیش می آید، اینکه اگر قرار باشد resampling صورت بگیرد و دوباره همانقدر قبلی برداریم چند درصد داده‌های جدید با قبلی یکسان خواهد بود؟
مشاهده میشود که با میل‌کردن *sample size* به بی‌نهایت، مقدار عبارت زیر به $0.63m$ میل میکند که نشان می‌دهد بیش از نصف داده‌ها تکراری خواهند بود.

$$۱ - (\frac{m-۱}{m})^m \to \lim_{m\to\infty} (۱ - (\frac{m-۱}{m})^m) = m(۱ - e^{-۱}) \approx ۰/۶۳m$$

$$P_{emp} : D = \{(x_i, y_i)\}$$



$$Predict : \hat{Y} = f_{\beta}(x)$$

هدف:

$$\min_{\hat{Y}=f_{\beta}(x)} E(L(Y, \hat{Y}))$$

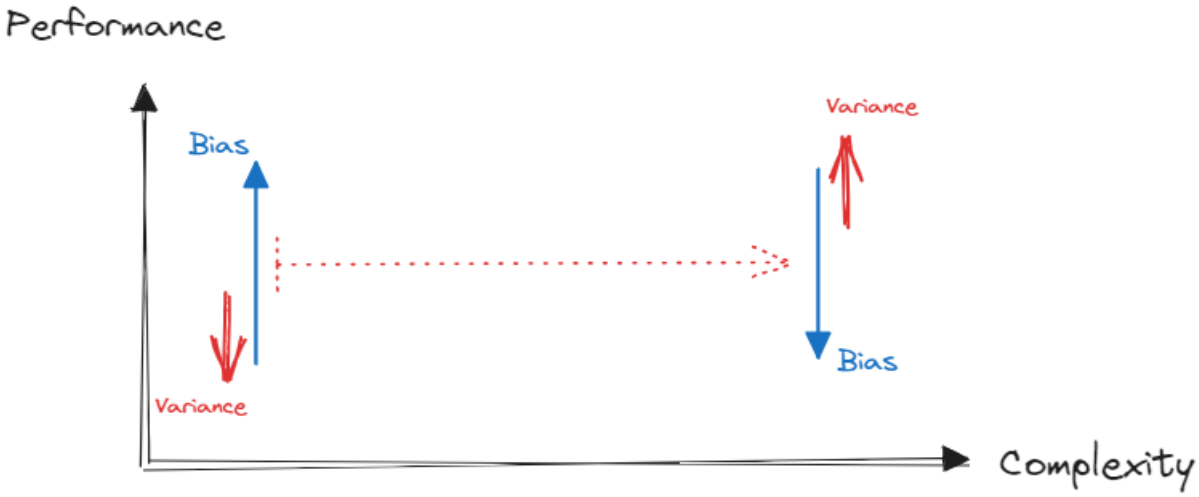
حالا یک Err_T هم داریم که نشان دهنده میزان خطا بر روی داده های آموزش می باشد.

$$\begin{aligned} Err_T &= E[(Y - \hat{Y})^2 | T] = E[(f(x) + \epsilon - f_{\beta}(x))^2 | T] \\ &= E[\epsilon + f(x) - \overline{f_{\beta}}(x) + \overline{f_{\beta}}(x) - f_{\beta}(x)] | T \\ &= \delta_E^2 + bias^2 + Var \end{aligned}$$

بایاس نشان دهنده فاصله ایست که تا خروجی اصلی داریم.
و واریانس مقدار آزادی و حرکتی که مدل ما می تواند انجام بدهد.

Model Complexity

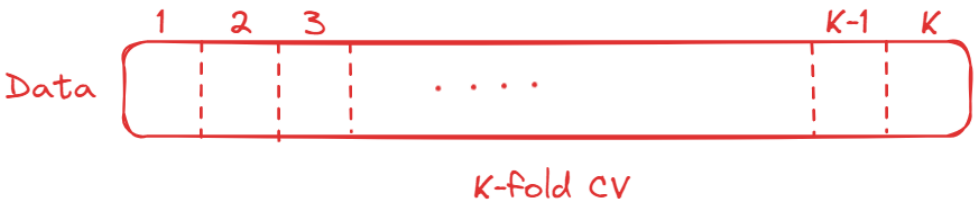
اگر پیچیدگی مدل ما زیاد باشد، ما واریانس زیاد و بایاس کمی خواهیم داشت. اگر ما سعی کنیم تا مدل را **simplify** کنیم یعنی از پیچیدگی آن بکاهیم، به نظر می آید که واریانس را کمتر و بایاس را زیادتر کرده‌ایم؛ البته باید توجه داشته باشیم که همیشه اینگونه نیست.
پس در نتیجه معمولا:



یکی از راه حل هایی که پیش رو داریم تقسیم کردن همه دیتا به قسمت های مختلفی است. بهتر است حدود ۵۰ درصد دیتا را به داده های train، و مابقی آن ها را بین داده های test و validation نصف کنیم. (البته در اهداف و داده های مختلف می توانیم به روش های دیگر این تقسیم را انجام بدهیم)

Cross Validation

راه قبلی برای وقتی که تعداد داده های زیاد و کافی داشته باشیم مشکلی ندارد و خوب است ولی اگر داده های کمی داشته باشیم باید سراغ راه حل های دیگری برویم.



در این حالت K بار باید عمل training انجام شود و به این شکل، K مدل خواهیم داشت. هر بار یک قسمت را برای validation و بقیه قسمت ها را برای train انتخاب می کنیم.

برای انتخاب مدل نهایی راه های مختلفی وجود دارد که یکی از آن ها را در اینجا می‌بینیم:

$$f(x) = \frac{۱}{K} \sum_{i=۱}^{K} f_{\beta i}(x)$$

و validation این مدل برابر با:

$$f(x) = \frac{۱}{m} \sum_{i=۱}^{m} L(Y_i, \hat{Y}_i)$$

مقدار جواب پیشبینی شده همان نتیجه f_β هستش که از روی یک قسمت از داده ها بدست آمده که در آن دفعه برای validation استفاده شده است.

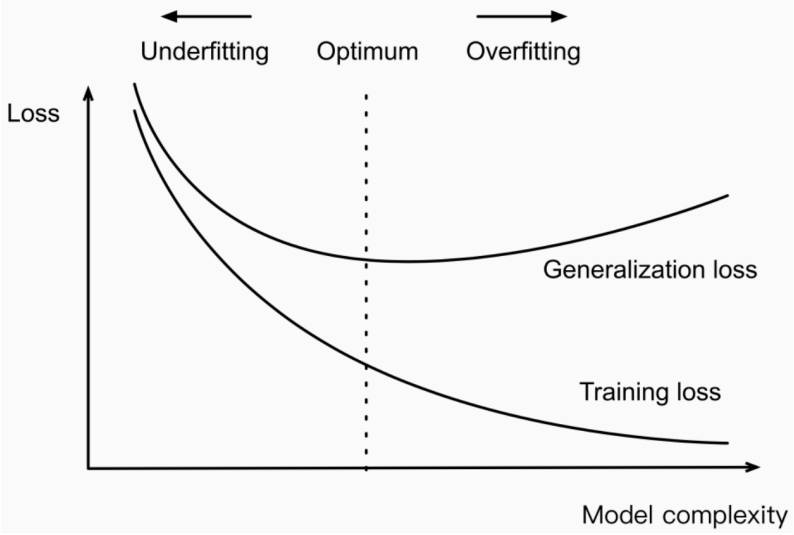
اگر نمونه ای که در هر دفعه برای validation قرار می دهیم فقط یکی باشد به این حالت LOOCV می گوییم.

Regularization

فرض کنید داریم: $D = \{(x_i, y_i) | i \in \{۱, ۲, \ldots, n\}\}$

مدل ما (همان $f_\beta(x)$) می‌تواند parametric یا non parametric باشد.

همچنین تابع هزینه [۱] را اگر با نمادگذاری $L(y, \hat{y})$ نمایش دهیم، هدف ما پیدا کردن β است که $\min \sum_{i=۱}^{n} L(y_i, \hat{y_i})$ را می‌دهد. منظور از «تعمیم» این است که مدل در داده‌های جدید که در مجموعه داده آموزشی نبوده (همان داده‌های تست) عملکرد خوبی داشته باشد. می‌خواهیم تاثیر complexity مدل را بر روی خطا بر روی داده‌های تست بررسی کنیم.



همانطورکه می‌بینیم با افزایش complexity خطای prediction نیز افزایش می‌یابد در نتیجه باید میزان com-plexity مدل را کنترل کنیم و این کار را می‌توانیم با استفاده از regularization انجام دهیم. Regularization

[۱] loss function

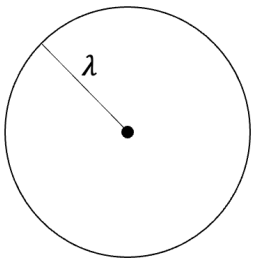
می‌تواند explicit یا implicit باشد. در explicit یک ناحیه را در نظر می‌گیریم؛ مانند $۰ \leqslant \beta_۱ \leqslant ۵$ یا $|\beta_۱| < ۰/۱$ و یا $\sum |\beta_i| < ۰/۷$
Regularization به صورت implicit در فرایند بهینه‌سازی اعمال می‌شود. در واقع الگوریتم را طوری در نظر می‌گیریم که از یک جا شروع کند و فقط ناحیه اطراف را جستجو کند. در نتیجه همه‌ی فضا در نظر گرفته نشده است. و complexity کمتر شده است. از مثال های آن می‌توان به drop out و early stopping اشاره کرد.
اگر مدل را بصورت زیر در نظر بگیریم:

$$f_\beta(x) = \sum_{i=۱}^{n} \beta_i x_i$$

انواع Regularization

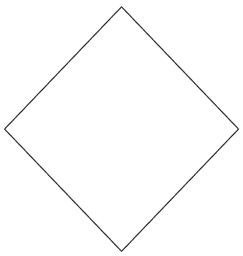
- Ridge Regularization

$$||\beta||_۲^۲ = \sum_{i=۱}^{n} \beta_i^۲ \leqslant \lambda$$



- Lasso Regularization

$$||\beta||_۱ = \sum_{i=۱}^{n} \beta_i \leqslant \lambda$$



- Subset Selection

$$||\beta||_۰ = \{\beta_i \neq ۰\} \leqslant \lambda$$

- Elastic Net

$$||\beta||_۲^۲ \leqslant \lambda_۱$$

$$||\beta||_۱ \leqslant \lambda_۲$$

Lasso به صورت خودکار یک سری از ضرایب را صفر می‌کند.
اگر داشته باشیم:

$$\min g(\beta)$$
$$\text{such that } h(\beta) \leqslant \lambda$$

می‌توانیم تابع هدف را به صورت زیر بازنویسی کنیم:

$$\min_{\beta} g(\beta) + \mu h(\beta)$$

در این صورت به جای تغییر λ می‌توانیم μ ها راتغییر دهیم. چون μ ، λ متناظر هم هستند و یعنی به ازای هر μ یک λ وجود دارد و برعکس.
به طور مثال در lasso داریم:

$$\min_{\beta} \frac{1}{n} \sum ||y_i - \beta_i x_i||^2$$
$$\text{s.t} \quad ||\beta||_1 \leqslant \lambda$$

که معادلا برابر است با:

$$min_{\beta} \frac{1}{n} \sum ||y_i - \sum \beta_i x_i||^2 + \mu ||\beta||_1$$

هر چه λ بیشتر شود پیچیدگی بیشتر می‌شود اما هر چه μ بیشتر شود پیچیدگی کمتر می‌شود.
در فضای non parametric نیز می‌توان constraint گذاشت:
به طور مثال:
بین همه f ها جستجو شود به طوری که:

$$constrain = \int f(x)^2 dx \leqslant \lambda$$

معادلا می‌توان نوشت:

$$\min \sum L(y_i, \hat{y_i}) + \mu \int f(x)^2 dx$$

در شبکه‌های عصبی معمولا از روش‌هایی مانند Early Stopping، Drop Out، Gradient Descent استفاده می‌شود و این باعث وجود نوعی Implicit Regularization در آن‌ها می‌گردد.