Lecture 9: Machine Learning: Structure & Latents

Structure Max-Margin extends binary classifier methods so they can be applied to learn the parameters of an MRF, HMM, SFCG or other model.

Recall standard SVM for binary classification:

\[ R(\lambda) = \frac{1}{2} \| \lambda \|^2 + C \sum_{i=1}^{m} \max\{0, 1 - y_i \phi(d_i) \} \]

Training Data \( \{(y_i, d_i)\} \)

\( y_i \in \{\pm 1\} \)

\( C \geq 0 \)

\( \phi(d_i) \)

\( \phi(d) \) = 1 if \( d \) is a plane. For \( d = 0 \)

Decision rule:

\[ \hat{y}_i(\lambda) = \arg \max_{y} y \cdot \phi(d_i) = \sgn \sum_{i=1}^{m} \phi(d_i) \]

The task is to minimize \( R(\lambda) \) w.r.t. \( \lambda \) which maximizes the 'margin' \( \| \lambda \| \).

Here is a more general formulation that can be used if the output variable \( y \) is a vector \( y = (y_1, \ldots, y_m) \) i.e., it could be the state of an MRF, an HMM, or a SFCG:

\[ R(\lambda) = \frac{1}{2} \| \lambda \|^2 + C \sum_{i=1}^{m} \Delta(y_i; \hat{y}_i(\lambda)) \]

Decision rule:

\[ \hat{y}_i(\lambda) = \arg \max_{y} y \cdot \phi(d_i, y) \]

the error function \( \Delta(y_i; \hat{y}_i(\lambda)) \) is any measure of distance between the true solution \( y_i \) and the estimate \( \hat{y}_i(\lambda) \)

\( \Rightarrow \) to obtain binary value: (i) Select \( y_i = y_i \in \{\pm 1\} \)

(ii) \( \hat{y}_i(\lambda) = \max(0, 1 - y_i \cdot 2 \cdot \phi(d_i)) \)

Margin loss \( \sqrt{\phi(d_i)} \) because the function is 0 linearly increasing with \( \phi(d_i) \) when the point is on the right side of the margin.

\( \hat{y}_i(\lambda) = \arg \max_{y} y \cdot \phi(d_i) \)
The more general formulation is:

\[ \Omega(2) = \frac{1}{2} J_i^2 + \sum_{i=1}^{m} \Delta(y_i; \hat{y}_i(2)) \]

\[ \hat{y}_i = \arg \max_{y_i} 2 \cdot \phi(y_i; y_i) \quad \text{for binary classification} \]

This requires an inference algorithm, inference only has to compute max \( y_i; \phi(y) \), so is trivial.

Also need to be able to minimize \( \Omega(2) \) to find \( \hat{y}_i \) hard because the error term \( \Delta(y_i; \hat{y}_i(2)) \) is a highly complicated function of \( \hat{y}_i \).

Modify \( \Omega(2) \) to an upper bound \( \Omega(2) \)

\[ \Omega(2) = \frac{1}{2} J_i^2 + \sum_{i=1}^{m} \max \{ \Delta(y_i; \hat{y}_i) + 2 \cdot \phi(d_i; \hat{y}_i) \} \]

which is convex in \( \hat{y}_i \)

This has a single minimum.

To get this bounds use two steps:

\[ \text{(Step 1)} \quad \max \{ \Delta(y_i; \hat{y}_i) + 2 \cdot \phi(d_i; \hat{y}_i) \} \]

\[ \hat{y}_i \geq \Delta(y_i; \hat{y}_i(2)) + 2 \cdot \phi(d_i; \hat{y}_i(2)) \]

\[ \rightarrow \hat{y}_i \geq \max \Delta + 2 \phi \]

\[ \text{(Step 2)} \quad 2 \cdot \phi(d_i, \hat{y}_i(2)) \geq 2 \cdot \phi(d_i, y_i) \]

Note: bounds are tight because if we can find a good solution then \( y_i \approx \hat{y}_i(2) \).

How to minimize \( \Omega(2) \)?

Several algorithms (hot topic)

Some in dual space - (the original SVM for binary problems).

Simple: Stochastic gradient descent

- take derivative of \( \Omega(2) \) w.r.t. \( \hat{y}_i \)

\[ \hat{y}_i = y_i - \eta \left( \Delta(y_i; \hat{y}_i) - \phi(d_i, y_i) \right) \]

where \( \eta = \arg \max \{ \Delta(y_i; \hat{y}_i) + 2 \cdot \phi(d_i; \hat{y}_i) \} \)

Note: inference algorithm must be adapted to compute this.
How to extend to models with latent (hidden) variables? Denote these variables by \( h \).

Want decision rule

\[
(\hat{y}, \hat{h}) = \arg \max_{(y, h)} \mathcal{L}(d, y, h) \quad \text{inference algorithm}
\]

Training data \( \{(d_i, y_i^i) : i = 1, \ldots, N\} \). The hidden variables are not known.

Loss function

\[
\Delta(y_i; \hat{y}_i(h, \hat{h}(h)))
\]

depends on the truth \( y_i \), the estimate of \( \hat{y}_i(h, \hat{h}(h)) \) from the model.

\[
R(z) = \frac{1}{2} \| z \|^2 + C \sum_{i=1}^n \Delta(y_i; \hat{y}_i(h, \hat{h}(h)))
\]

Replace \( R(z) \) by

\[
\tilde{R}(z) = \frac{1}{2} \| z \|^2 + C \sum_{i=1}^n \left( \max_{h \in H} \left\{ \Delta(y_i; \hat{y}_i(h, \hat{h}(h))) + \alpha \cdot \phi(d_i, y_i, h) \right\} - \max_{h \in H} \alpha \cdot \phi(d_i, y_i, h) \right)
\]

\[
\tilde{R}(z) = f(z) + g(z)
\]

convex

To show convexity and concavity

\[
\sup_{z \in H} \tilde{R}(z) = \frac{1}{2} \| \sum_{i=1}^n \max_{h \in H} \phi(d_i, y_i, h) \|
\]

Convex if

\[
\tilde{R}(z) \leq \alpha \tilde{R}(z_1) + (1-\alpha) \tilde{R}(z_2)
\]

\[
\tilde{R}(z_1 + (1-\alpha) z_2) = \frac{1}{2} \| z_1 + (1-\alpha) z_2 \|^2 \max_{g_i} \phi(d_i, y_i, g_i)
\]

\[
\tilde{R}(z_1 + (1-\alpha) z_2) = \alpha \frac{1}{2} \| z_1 \|^2 \max_{g_i} \phi(d_i, y_i, g_i)
\]

\[
+ (1-\alpha) \frac{1}{2} \| z_2 \|^2 \max_{g_i} \phi(d_i, y_i, g_i)
\]

but

\[
\max_{g_i} \phi(d_i, y_i, g_i) \geq \max_{g_i} \left\{ \phi(d_i, y_i, g_i), \phi(d_i, y_i, g_i) \right\}
\]

Hence \( f(\cdot) \) is convex

and \( g(\cdot) \) is concave.
Apply CCCP algorithm.

Two staged: step 1. \[ \frac{\partial g(\lambda)}{\partial h} = - \frac{1}{2} \phi'(d_i, y_i, h^*) \]

where \( h^* = \arg \max \phi'(d_i, y_i, h) \)

\( \lambda^t \) current estimate of \( \lambda \).

Step 2. Some

\[ \lambda^{t+1} = \arg \min_{\lambda} \left( f(\lambda) + \lambda \cdot \frac{2}{\sigma^2} g(\lambda) \right) \]

This reduces to a modified SVM with known \( \lambda \)

\[ \min_{\lambda} \frac{1}{2} \lambda^T \Sigma^{-1} \lambda + \frac{1}{2} \max_{c=1} \max_{y, b} \phi'(d_i, y_i, b) \]

\[ - \frac{1}{2} \sum_{i \in C} \lambda \cdot \phi'(d_i, y_i, h^*) \]

Note: Similarity to EM.

Step 1 involves estimating the hidden state \( h^* \).

Step 2 estimates \( \lambda \).

Note: Like EM, there is no guarantee that this will converge to the global optimum.