Grafting-Light: Fast, Incremental Feature Selection and Structure Learning of Markov Random Fields
(Zhu, Lao and Xing, 2010)

Pete Stein
COMP 790-90 | 10/28/10
Grafting: Sneak Peek

- What is “grafting?”
  - Gradient Feature Testing
- What are we doing?
  - Using “gradient descent” and other methods to approximate optimal features from a dataset.

(...not this kind of grafting.)
Motivation: Feature Selection

- Past approaches to features/feature-selection
  - Mysterious
  - Ad-hoc
  - Detached from the learning process
- Choosing the best features versus learning: two sides of the same coin?
Motivation: Feature Selection

- Challenge: a robust framework for intelligent, efficient feature selection
  - Support large number of features
  - Computational efficiency
Grafting-Light Application

- Seeks features for modeling with Markov random fields
  - Entails large number of high-dimensional inputs and features
  - Selecting optimal features in NP-hard

- Options? Previous works:
  - “Batch” methods that optimize all candidate features
  - Incremental methods (e.g., “Grafting”)

Andrey Markov, 1856-1922
Generally, how do we select features?

Want features that minimize risk:

\[
R(\theta) = \int L(f_\theta(x), y) p(x, y) \, dx \, dy
\]

\(L()\) = penalty for deviation of \(f()\) from \(y\)

\(P()\) = JPDF (joint probability density function)

Example \(L()\):

\[
L = \frac{1}{2} \left| y - \text{sign}(f(x)) \right|
\]
Empirical Risk

• In practice, we don’t know $p$. Instead:
  • Minimize a combination of empirical risk (for each data point)
  • Plus a regularization term

$$C(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(f_{\theta}(x_i), y_i) + \Omega(\theta)$$
Loss Functions

- Desirable to choose classifiers that separate data as much as possible.
- I.e., maximize margin $\rho = y \cdot f(x)$
- So, better loss functions $L()$ are possible, e.g., Binomial Negative Log Likelihood:

$$L_{bnll} = \ln(1 + e^{-\rho})$$
Loss Functions

![Example Loss Functions](image)

- Error Rate
- SVM
- Perceptron
- Binomial Negative Log Likelihood
Regularization

- Why not just minimize empirical risk and leave it at that?
  - Optimization problem may be unbounded w.r.t. parameters
  - Overfitting: solution may fit one set of data very well, and then do poorly on the next set.
What does this regularization term look like?

\[ \Omega_q(w) = \lambda \sum_{i=1}^{p} \alpha_i |w_i|^q \]

- \( q \) = regularization norm
- \( w \) = weighting vector
- \( \lambda \) = regularization parameters
- \( \alpha \) = inclusion/exclusion vector – e.g., \{0, 1\}
Which Regularization?

- Depends on “pragmatic” motivations (performance, scalability)
- Of particular interest: $q \in \{0, 1, 2\}$

<table>
<thead>
<tr>
<th>Criterion</th>
<th>$\Omega_0$</th>
<th>$\Omega_1$</th>
<th>$\Omega_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Models pragmatic motivations for feature selection?</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Models performance motivations for feature selection?</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Leads to sparse solutions?</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Performance when most features are relevant?</td>
<td>OK</td>
<td>Good</td>
<td>Excellent</td>
</tr>
<tr>
<td>Performance when most features are irrelevant?</td>
<td>Poor</td>
<td>Good</td>
<td>OK</td>
</tr>
<tr>
<td>Convex regularizer? (doesn’t add extra local optima)</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Numerical friendliness</td>
<td>Poor</td>
<td>Good</td>
<td>Excellent</td>
</tr>
</tbody>
</table>
Unified Regularization

Combining the three criterions:

\[ C(w) = \frac{1}{m} \sum_{i=1}^{m} L(f(x), y_i) + \lambda_2 \sum_{i=1}^{\mu} \alpha_{2,i} |w_i|^2 + \lambda_1 \sum_{i=1}^{\mu} \alpha_{1,i} |w_i| + \lambda_0 \sum_{i=1}^{n} \alpha_{0,i} \delta_i \]

Now, how do we minimize \( C(w) \)? Grafting.
Grafting

• General strategy:
  • Use gradient descent with model parameters $w$

• But, obstacles exist:
  • Slow (quadratic w.r.t. data dimensionality)
  • Numerical issues for some regularizers
Gradient Descent

• First, what is gradient descent?
  • First-order optimization algorithm
  • Finds local minima (local maxima = gradient ascent)
  • Take steps proportional to the gradient
• Relatively slow in some situations
Grafting: Stagewise Optimization

- **Basic steps:**
  - Split parameter weights \(w\) into free \(F\) and fixed \(Z\) sets
  - During each stage, transfer one weight (with largest contribution to \(C(w)\)) from \(Z\) to \(F\).
  - Use gradient descent to minimize \(C(w)\) w.r.t. \(F\).

- Gradient descent is faster with the stage-wise approach, but still slow.
Grafting-Light

- Main difference: at each grafting step, use fast gradient-based heuristic to choose features that will reduce $C(w)$ the most.
- Isn’t this the same as regular grafting?
Grafting-Light

- Grafting: optimize over all free parameters at each stage.
- Grafting-Light: only one gradient descent optimization

Figure 1: 2D illustration of the algorithms: (Left) Grafting-Light; (Right) Grafting.
Grafting/Grafting-Light Comparison

- Step-by-step comparison of Grafting and Grafting-Light:

  **Grafting-Light:**
  1. One step of gradient descent
  2. Select new features (e.g., v)
  3. 2D gradient descent

  **Grafting:**
  1. Optimize from \( \mu \) to local optimum \( \mu^* \)
  2. Select new features (e.g., v)
  3. Optimize over \( \mu \) and v

---

Figure 1: 2D illustration of the algorithms: (Left) Grafting-Light; (Right) Grafting.
Results / Comparison

Full-Opt.-L1  Grafting  Grafting-Light

3/22/2011
The End

Thank you; questions?