New Algorithms for Approximate Minimization of the Difference Between Submodular Functions, with Applications

Rishabh Iyer Jeff Bilmes

University of Washington, Seattle

Department of Electrical Engineering

December 13, 2013

Outline

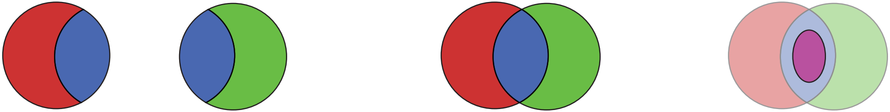
Outline

Submodular Functions

- A function $f : 2^V \to \mathbb{R}$ is submodular if for all $A, B \subseteq V$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$.
- Coverage of intersection of elements is less then common coverage.

$$f(A) + f(B) \quad \geq \quad f(A \cup B) \quad + \quad f(A \cap B)$$

$$= f(A_r) + 2f(C) + f(B_r) \qquad\qquad = f(A_r) + f(C) + f(B_r)$$



- Equivalently, diminishing returns: Let $A \subseteq B \subseteq V \setminus \{j\}$ then f is submodular iff

$$f(v|A) \triangleq f(A + v) - f(A) \geq f(B + v) - f(B) \triangleq f(v|B) \qquad (1)$$

- I.e., conditioning reduces valuation (like entropy).

Outline

Optimizing The Difference between Two Submodular Functions

- In this paper, we address the following problem. Given two submodular functions f and g, solve the optimization problem:

$$\min_{X \subseteq V} [f(X) - g(X)] \equiv \min_{X \subseteq V} [v(X)] \tag{2}$$

  with $v : 2^V \to \mathbb{R}$, $v = f - g$.
- A function r is said to be supermodular if $-r$ is submodular.

Applications

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $\min_A f(A) - \lambda c(A)$.

Applications

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $\min_A f(A) - \lambda c(A)$.
- Discriminatively structured graphical models, EAR measure $I(X_A; X_{V \setminus A}) - I(X_A; X_{V \setminus A} | C)$, and synergy in neuroscience.

Applications

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $\min_A f(A) - \lambda c(A)$.
- Discriminatively structured graphical models, EAR measure $I(X_A; X_{V \setminus A}) - I(X_A; X_{V \setminus A} | C)$, and synergy in neuroscience.
- Feature selection: a problem of maximizing $I(X_A; C) - \lambda c(A) = H(X_A) - [H(X_A|C) + \lambda c(A)]$, the difference between two submodular functions, where H is the entropy and c is a feature cost function.

Applications

- Sensor placement with submodular costs. I.e., let V be a set of possible sensor locations, $f(A) = I(X_A; X_{V \setminus A})$ measures the quality of a subset A of placed sensors, and $c(A)$ the submodular cost. We have $\min_A f(A) - \lambda c(A)$.
- Discriminatively structured graphical models, EAR measure $I(X_A; X_{V \setminus A}) - I(X_A; X_{V \setminus A} | C)$, and synergy in neuroscience.
- Feature selection: a problem of maximizing $I(X_A; C) - \lambda c(A) = H(X_A) - [H(X_A|C) + \lambda c(A)]$, the difference between two submodular functions, where H is the entropy and c is a feature cost function.
- Graphical Model Inference. Finding x that maximizes $p(x) \propto \exp(-v(x))$ where $x \in \{0, 1\}^n$ and v is a pseudo-Boolean function. When v is non-submodular, it can be represented as a difference between submodular functions.

Heuristics for General Set Function Optimization

Lemma (Narisimham & Bilmes, 2005)

Given any set function v, it can be expressed as $v(X) = f(X) - g(X), \forall X \subseteq V$ for some submodular functions f and g.

- We give a new proof that depends on computing $\alpha_v = \min_{X \subset Y \subseteq V \backslash j} v(j|X) - v(j|Y)$ which can be intractable for general v.
- However, we show that for those functions where α_v can be bounded efficiently, f and g can be computed efficiently.

Lemma

For a given set function v, if α_v or a lower bound can be found in polynomial time, a corresponding decomposition f and g can also be found in polynomial time.

Outline

Convex/Concave and Semigradients

- A convex function ϕ has a subgradient at any in-domain point y, namely there exists h_y such that

$$\phi(x) - \phi(y) \geq \langle h_y, x - y \rangle, \forall x. \quad (3)$$

- A concave ψ has a supergradient at any in-domain point y, namely there exists g_y such that

$$\psi(x) - \psi(y) \leq \langle g_y, x - y \rangle, \forall x. \quad (4)$$

- If a function has both a sub- and super-gradient at a point, then the function must be affine.

Submodular Subgradients

- For submodular function f, the subdifferential can be defined as:

$$\partial f(X) = \{x \in \mathbb{R}^V : \forall Y \subseteq V, x(Y) - x(X) \leq f(Y) - f(X)\} \quad (5)$$

- Extreme points of the sub-differential are easily computable via the greedy algorithm:

Theorem (Fujishige 2005, Theorem 6.11)

A point y is an extreme point of $\partial f(Y)$, iff there exists a chain $\emptyset = S_0 \subset S_1 \subset \cdots \subset S_n$ with $Y = S_j$ for some j, such that $y(S_i \setminus S_{i-1}) = y(S_i) - y(S_{i-1}) = f(S_i) - f(S_{i-1})$.

The Submodular Subgradients (Fujishige 2005)

- Let σ be a permutation of V and define $S_i^\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$ as σ's chain containing Y, meaning $S_{|Y|}^\sigma = Y$ (we say that σ's chain contains Y).
- Then we can define a subgradient h_Y^f corresponding to f as:

$$h_{Y,\sigma}^f(\sigma(i)) = \begin{cases} f(S_1^\sigma) & \text{if } i = 1 \\ f(S_i^\sigma) - f(S_{i-1}^\sigma) & \text{otherwise} \end{cases}.$$

- We get a tight modular lower bound of f as follows:

$$h_{Y,\sigma}^f(X) \triangleq \sum_{x \in X} h_{Y,\sigma}^f(x) \leq f(X), \forall X \subseteq V.$$

Note, $h_{Y,\sigma}^f(Y) = f(Y)$.

Submodular/Supermodular Procedure

From Narisimham&Bilmes 2005.

Algorithm 1 The submodular-supermodular (SubSup) procedure

1: $X^0 = \emptyset$; $t \leftarrow 0$;
2: **while** not converged (i.e., $(X^{t+1} \neq X^t)$) **do**
3: Randomly choose a permutation σ^t whose chain contains the set X^t.
4: $X^{t+1} := \text{argmin}_X f(X) - h^g_{X^t,\sigma^t}(X)$
5: $t \leftarrow t + 1$
6: **end while**

Lemma

Algorithm 1 is guaranteed to decrease the objective function at every iteration. Further, the algorithm is guaranteed to converge to a local minima by checking at most $O(n)$ permutations at every iteration.

The Submodular Supergradients

- Can a submodular function also have a supergradient? We saw that in continuous case, simultaneous sub/super gradients meant linear.
- (Nemhauser, Wolsey, & Fisher 1978) established the following iff conditions for submodularity (if either hold, f is submodular):

$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j|X \setminus j) + \sum_{j \in Y \setminus X} f(j|X \cap Y),$$
$$f(Y) \leq f(X) - \sum_{j \in X \setminus Y} f(j|(X \cup Y) \setminus j) + \sum_{j \in Y \setminus X} f(j|X)$$

Note that $f(A|B) \triangleq f(A \cup B) - f(B)$ is the gain of adding A in the context of B.

Submodular and Supergradients

- Using submodularity further, these can be relaxed to produce two tight modular upper bounds (Jegelka & Bilmes, 2011):

$$f(Y) \leq m^f_{X,1}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|X \setminus j) + \sum_{j \in Y \setminus X} f(j|\emptyset),$$
$$f(Y) \leq m^f_{X,2}(Y) \triangleq f(X) - \sum_{j \in X \setminus Y} f(j|V \setminus j) + \sum_{j \in Y \setminus X} f(j|X).$$

Hence, this yields two tight (at set X) modular upper bounds $m^f_{X,1}, m^f_{X,2}$ for any submodular function f.

Supermodular-Submodular (SupSub) Procedure

Algorithm 2 The supermodular-submodular (SupSub) procedure

1: $X^0 = \emptyset$; $t \leftarrow 0$;
2: **while** not converged (i.e., $(X^{t+1} \neq X^t)$) **do**
3: $\quad X^{t+1} := \text{argmin}_X \, m^f_{X^t}(X) - g(X)$
4: $\quad t \leftarrow t + 1$
5: **end while**

Theorem

The supermodular-submodular procedure (Algorithm 2) monotonically reduces the objective value at every iteration. Moreover, assuming a submodular maximization procedure in line 3 that reaches a local maxima of $m^f_{X^t}(X) - g(X)$, then if Algorithm 2 does not improve under both modular upper bounds then it reaches a local optima of v.

Supermodular-Submodular (SupSub) Procedure

- Each iteration requires submodular maximization, while this is NP-complete, it is easy to well approximate.
- Very recently, a fast randomized linear-time 1/2-approximation algorithm for submodular max was developed (FOCS 2012, "A Tight Linear Time (1/2)-Approximation for Unconstrained Submodular Maximization", Buchbinder, Feldman, Naor and Schwartz).
- The algorithm is extremely simple, and is essentially a randomized bi-directional greedy algorithm (very few iterations needed in practice).

Modular-Modular Procedure

Algorithm 3 Modular-Modular (ModMod) procedure

1: $X^0 = \emptyset$; $t \leftarrow 0$;
2: **while** not converged (i.e., $(X^{t+1} \neq X^t)$) **do**
3: Choose a permutation σ^t whose chain contains the set X^t.
4: $X^{t+1} := \text{argmin}_X \, m^f_{X^t}(X) - h^g_{X^t,\sigma^t}(X)$
5: $t \leftarrow t + 1$
6: **end while**

Theorem

Algorithm 3 monotonically decreases the function value at every iteration. If the function value does not increase on checking $O(n)$ different permutations with different elements at adjacent positions and with both modular upper bounds, then we have reached a local minima of v.

Modular-Modular Procedure

- Each iteration is very fast since only modular min.
- If $v >= 0$, then also applies to combinatorial constraints (trees, paths, matchings, cuts, etc.) since each iteration becomes standard combinatorial algorithm.
- In SubSup and ModMod the choice of the permutations is important since there are a combinatorial number of them (an problem left open from 2005).
- In the paper, we provide some heuristics which work well in practice.

Outline

1. Background
2. Optimizing $v(X) = f(X) - g(X)$ with f, g submodular
3. Procedures for minimizing $v(X)$
   - The Submodular Supermodular Procedure
   - The Supermodular Submodular Procedure
   - The Modular Modular Procedure
4. **Some Additional Theoretical Results**
5. Experiments

A P $\neq$ NP Hardness Result

- Given submodular functions f and g, the problem $\min_X [f(X) - g(X)]$ is inapproximable.

Theorem

Unless P = NP, there cannot exist any polynomial time approximation algorithm for $\min_X v(X)$ where $v(X) = [f(X) - g(X)]$ is a positive set function and f and g are given submodular functions. In particular, let n be the size of the problem instance, and $\alpha(n) > 0$ be any positive polynomial time computable function of n. If there exists a polynomial-time algorithm which is guaranteed to find a set $X' : f(X') - g(X') < \alpha(n) OPT$, where $OPT = \min_X v(X)$, then P = NP.

Information Theoretic Hardness

- We also have an information theoretic hardness result (i.e., one that is independent of the P $=$ NP question).

Theorem

For any $0 \leq \epsilon < 1$, there cannot exist any deterministic (or possibly randomized) algorithm for $\min_X [f(X) - g(X)]$ (where f and g are given submodular functions), that always finds a solution which is at most $\frac{1}{\epsilon}$ times the optimal, in fewer than $e^{\epsilon^2 n/8}$ queries.

Poly-time lower bounds on the optima

- On the more positive side, we do have lower bounds:

Theorem

Given submodular functions f and g, define $f'(X) \triangleq f(X) - \sum_{j \in X} f(j|V \backslash j)$, $g'(X) \triangleq g(X) - \sum_{j \in X} g(j|V \backslash j)$ *and* $k(X) = \sum_{j \in X} v(j|V \backslash j)$. *Then we have the following bounds:*

$$\min_X v(X) \geq \min_X f'(X) + k(X) - g'(V)$$

$$\min_X v(X) \geq f'(\emptyset) - g'(V) + \sum_{j \in V} \min(k(j), 0)$$

Computational bounds

This can be used to prove:

Theorem

The ϵ-approximate versions of algorithms 1, 2 and 3 have a worst case complexity of $O(\frac{\log(|M|/|m|)}{\epsilon} T)$, where $M = f'(\emptyset) + \sum_{j \in V} \min(v(j|V \backslash j), 0) - g'(V)$, $m = \min_k v(k)$ and $O(T)$ is the complexity of every iteration of the algorithm (which corresponds to respectively the submodular minimization, maximization, or modular minimization in algorithms 1, 2 and 3)..

Theoretical results - summary

- The problem $\min_X f(X) - g(X)$ for given submodular functions f and g is in general inapproximable.
- An information theoretic lower bound guarantees no sub-exponential time algorithm for exact minimization.
- Can provide poly-time lower bounds on the optimum, which can yield worst case additive approximation guarantees.
- Complexity results that our algorithms are polynomial time.
- And, again, the aforementioned local optima results of our new algorithms.

Outline

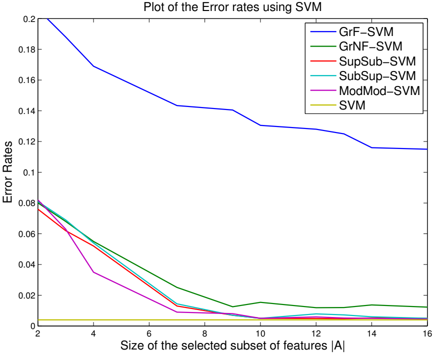
Experiments

- We consider features selection with objective $f(A) = I(X_A; C) = H(X_A) - H(X_A|C)$ (a difference between submodular functions) and **not** under the naïve Bayes model.
- We also consider two cost models, λ a tradeoff coefficient. Either
  1. modular cost model $c(A) = \lambda|A|$
  2. or submodular cost model using $c(A) = \lambda \sum_i \sqrt{m(A \cap S_i)}$ for a random partition of V and random weights m.
- We test two classifiers, a linear kernel SVM and a naïve Bayes (NB) classifier
- Data sets:
  1. Mushroom data (Iba, Wogulis, Langley, 1988), 8124 examples with 112 features.
  2. Adult data (Kohavi, 1996), 32,561 examples with 123 features.

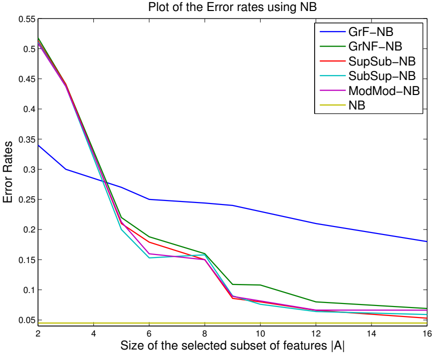
Feature Selection Algorithms Evaluated

- Feature selection algorithms evaluated.
  1. Greedy with factored MI (GrF) - simple greedy selection using conditional mutual information (CMI) with a NB assumption.
  2. Greedy with non-factored MI (GrNF) - greedy selection using CMI without assumptions.
  3. Submodular-Supermodular procedure (SubSup).
  4. Supermodular-Submodular procedure (SupSub).
  5. Modular-Modular procedure (ModMod)
- In SubSup and ModMod, we used the aforementioned smart permutation heuristic.
- ModMod and SubSup use exact minimization at each iteration, while SupSub use approximate minimization (via the new FOCS 2012 submodular maximization algoritm).

Mushroom Data - modular cost features



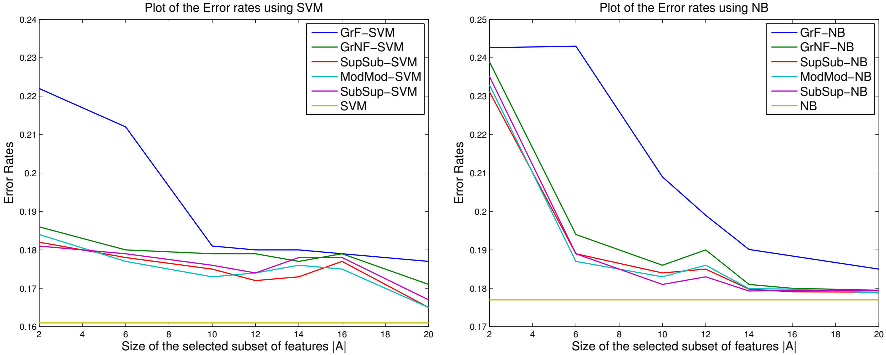
(a) SVM



(b) NB

Figure : Plot showing the accuracy rates vs. the number of features on the Mushroom data set.

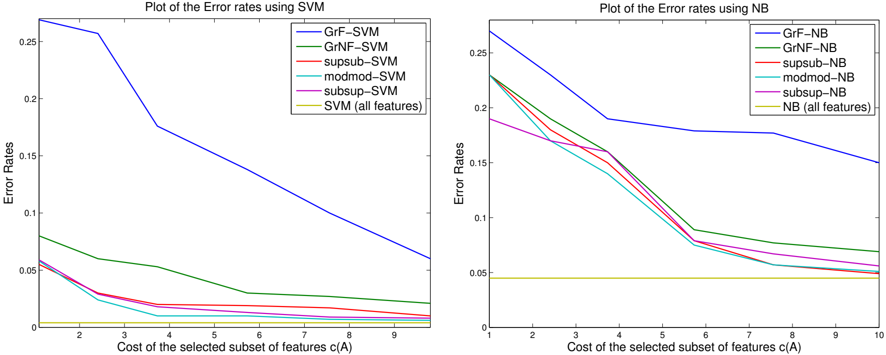
Adult Data - modular cost features



(a) SVM (b) NB

Figure : Plot showing the accuracy rates vs. the number of features on the Adult data set.

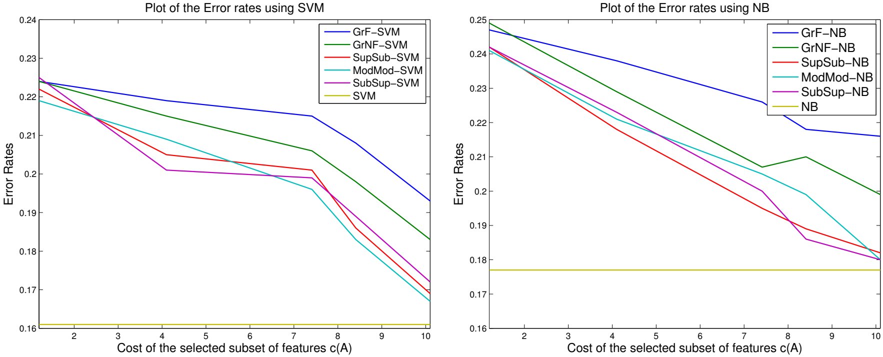
Mushroom Data - submodular cost features



(a) SVM (b) NB

Figure : Plot showing the accuracy rates vs. the cost of features for the Mushroom data set

Adult Data - submodular cost features



(a) SVM (b) NB

Figure : Plot showing the accuracy rates vs. the cost of features for the Adult data set

Experiments - Results Summarized

- Permutation heuristic is important for the performance of ModMod and SubSup.
- ModMod and SubSup do not show significant difference, but ModMod is must faster and scales very well.
- SubSup does not show appreciable benefit even though it uses exact submodular minimization at each iteration (and is slower).
- GrF and GrNF in general does not perform as well (with GrF worse than GrNF).
- More benefit to the $v = f - g$ approach under the submodular cost model than under the modular cost model.

Summary

- Applications of minimizing the difference between two submodular functions.
- New algorithms for minimizing the difference between two submodular functions.
- New theoretical hardness results and complexity bounds.
- Empirical experimental validation.