10-725: Optimization **Fall 2012**

Lecture 4: September 6

Lecturer: Geoff Gordon/Ryan Tibshirani *Scribes: Haijie Gu, William Bishop*

Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

4.1 Example: Structured Classifier for Handwriting Recognition

Problem: Given an input image of a handwriting word, for example: "c o n v e x", predict the most likely string for the image, "convex" in this case.

Model: We use a linear model for each single character and neighbouring characters (bigram). Table 4.1 lists the variables and parameters used in the model.

Model Variables	Description
x_i	pixels (raw feature vector) of character i
$\phi(x_i)$	feature vector of x_i
y_i	label of character i in a given alphabet Σ (e.g. $a, \ldots, z$)
$\psi_{ik}(x_i, y_i)$	$\phi(x_i)\delta(y_i = k)$, connects an observation and its label
$\chi_{ikl}(y_i, y_{i+1})$	$\delta(y_i = k)\delta(y_{i+1} = l)$, connects consecutive labels
Model Parameters	
w_{ik}	weights for ψ_{ik}
v_{ikl}	weights for χ_{ikl}

Table 4.1: Variables and parameters in the model.

Score function: for each input image $\mathbf{x}$ and string label $\mathbf{y}$, the model computes a score as follows:

$$L(\mathbf{x}, \mathbf{y}; w, v) = \sum_{i=1}^{|\mathbf{x}|} \sum_{k=1}^{|\Sigma|} \psi_{ik}^T(x_i, y_i) w_{ik} + \sum_{i=1}^{|\mathbf{x}|-1} \sum_{k,l=1}^{|\Sigma|} \chi_{ikl}(y_i, y_{i+1}) v_{ijk} \tag{4.1}$$

Classifier: the classifier chooses the string label which outputs the highest score for the input.

$$\hat{y} = \arg\max_y L(\mathbf{x}, \mathbf{y}; w, v) \tag{4.2}$$

Note that without the second part in Eq. 4.1, it would be a simple classifier based on each individual character.

Learning: Given a set of n training examples: $\{\mathbf{x_t}, \mathbf{y_t}\}_{t=1}^n$, learn the weights v, w in the model.
Intuitively, we want our learning algorithm to optimize a function such that the corrected labels in the training set gets the highest score:

$$L(\mathbf{x_t}, \mathbf{y_t}; w, v) > L(\mathbf{x_t}, \mathbf{y}'; w, v) \qquad \forall y' \neq y_t \tag{4.3}$$

However, the $<$ in the above equation defines a "open" relationship which is too loose, because it allows LHS and RHS to get arbitrarily close, which could lead to trivial solutions where all weights are set to zero. Furthermore, we would like to penalize labels that are more wrong than others. For example, "convex" is a more reasonable answer than "xevnoc", thus deserving higher score. We introduce the distance function $\pi(\mathbf{y}, \mathbf{y}') = dist(\mathbf{y}, \mathbf{y}')$, where $dist$ is the edit distance between string $\mathbf{y}$ and $\mathbf{y}'$.

$$L(\mathbf{x_t}, \mathbf{y_t}; w, v) \geq \max_{y'} L(\mathbf{x_t}, \mathbf{y}'; w, v) + \pi(\mathbf{y_t}, \mathbf{y}') \tag{4.4}$$

Finally, the optimization problem of learning the weights v, w is defined by summing over all training example the difference of the true label score and the maximum over all label score plus the label distance:

$$\min_{v,w} \sum_{t=1}^{n} (\max_{y'} L(\mathbf{x_t}, \mathbf{y}'; w, v) + \pi(\mathbf{y_t}, \mathbf{y}')) - (L(\mathbf{x_t}, \mathbf{y_t}; w, v)) + \lambda(\|v\|^2 + \|w\|^2) \tag{4.5}$$

Remarks:

- The function in 4.5 is convex in v, w because the score function 4.1 is liner.
- $\max_{y'} L(\mathbf{x_t}, \mathbf{y}'; w, v)$ can be computed efficiently using dynamic programming. Let $A(i, j)$ be the max score of a label whose ith character is j. Then $A(i+1, k) = \max_j A(i, j) + \psi^T_{i+1,k}(x_{i+1})w_{ik} + \chi_{i+1,j,k} v_{i+1,j,k}$
- The regularization for v, w is necessary to prevent v, w from getting arbitrary large. For example, in the "optimal" case where there exists v, w such that all the conditions defined by the Inequalities 4.4 are satisfied, scaling v, w by an arbitrarily large constant still results in a perfect solution.

4.2 Strict, Strong Convexity, and More

Strictly convex:

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y) \qquad x, y \in dom(f), t \in (0, 1), x \neq y \tag{4.6}$$

Example: $f(x) = ln(1 + e^x)$ is strictly convex.

k-Strongly convex

$$f(y) \geq f(x) + (y - x)f'(x) + \frac{k(y-x)^2}{2} \tag{4.7}$$

Example: $f(x) = |x| + x^2$ is 2-strongly convex.

Lipschitz $Lipschitz(L)$ w.r.t. norm $\|x\|$ is a class of function defined as:

$$\{f : |f(x) - f(y)| \leq L\|x - y\|\} \tag{4.8}$$

Example: $x^2/3$ is not $Lipschitz$, $|x|$ is in $Lipschitz(1)$, and $sgn(x)$ is not $Lipschitz$.

Remarks:

- Strict convex is a general condition and k-strongly convex is a more specific and stronger condition which states that the function is lower bounded everywhere by a quadratic function at that point. Less rigorously, the faster a function grows (in either direction), the stronger convexity it has.

- Lipschitz condition, on the other hand controls the smoothness of a function. A function which satisfies Lipschitz condition cannot grow too fast, precisely, faster than linear. It is easy to check that a function cannot be both Lipschitz and strongly convex.
- Lipschitz condition is also commonly applied to the gradient of a function to upper bound the growth of the function.

Extended Reals We define extended reals for the convenience of convex analysis so that we don't have to worry about the checking the domain.
Suppose $domf \subset R^n$, we define

$$g(x) = \begin{cases} f(x) & \text{if } x \in dom(f) \\ \infty & \text{otherwise} \end{cases}$$

It is easy to check that f is convex $\Leftrightarrow$ g is convex using the definition of convex function.

4.3 Convergence Rates of Gradient Descent

We can use now state the convergence rate of gradient descent under different conditions. These learning rates, as well as the step-sizes that should be used for each condition, are listed in the table below. In all cases we assume the function, $f(x)$, we are optimizing is convex and the gradient exists.

Assumptions on $f(x)$	Iterations to Reach Accuracy $\epsilon(f(x_0) - f(x^*))$	Step Size To Select
f is Lipschitz	$O(1/\epsilon^2)$	$t_k \approx 1/\sqrt{k}$
$\bigtriangledown f$ is Lipschitz	$O(1/\epsilon)$	t_k is constant
f is strongly convex	$O(\ln(1/\epsilon))$	t_k is constant

Table 4.2: Convergence rates of gradient descent and learning rates to chose under different assumptions on the function being optimized, $f(x)$. In the notation in the heading for the second column, x_0 is the point from which gradient descent starts, and x^* is the optimal solution.

Note that the performance of gradient descent will also depend on the conditioning of $f(x)$, where conditioning can be understand as characterizing how elliptical the Hessian is.

4.4 Extensions to Gradient Descent

There are multiple methods that can be considered as extensions of gradient descent. Among these are, subgradient descent, generalizations based on the prox operator, accelerated methods (FISTA, mirror descent and conjugate gradient), Nesterov's smoothing method, line search, and stochastic gradient descent.

Stochastic Gradient Descent Stochastic GD is useful when the function you are trying to minimize is the expected value of a function of a random variable. Basically, you just look at one sample at every iteration and calculate your gradient based on that sample, take a step and hope that on average you are going in the right direction. Note that because you only look at a single example every iteration, the per iteration computational cost is very low.

The behavior of stochastic GD is quite different than normal gradient descent. If the function you are optimizing is Lipschitz, stochastic GD still converges at a rate of $O(1/\epsilon^2)$, but the constant hidden in this notation is worse. However, the cost for each iteration is low, so stochastic gradient descent can be a big win

if your function is Lipschitz. If the function is strongly convex, we get a convergence rate of $O(\ln(1/\epsilon)/\epsilon)$, which is not as good as normal gradient descent, but again, this will be balanced by the low computational cost per iteration.

4.5 Convex Functions and Global Minima

Theorem 4.1 *Let C be the feasible set for an optimization problem. Suppose there exists an $x^* \in C$ such that $f(x^*) \leq y \ \forall y$ s.t. $||x^* - y|| \leq R$ for some value of $R > 0$ and f is a convex function. Then x^* is a global minimum of f.*

Proof:

The proof is by contradiction. Assume that there exists a $z \in C$ such that $f(z) < f(x^*)$ and $||x^* - z|| > R$.

Next, define a point $y = (1-\alpha)x^* + \alpha z$ for $\alpha \in (0,1)$. We pick $\alpha = \frac{R}{||x^* - z||}$, so that:

$$\begin{aligned} ||x^* - y|| &= \|x^* - (1-\alpha)x^* + \alpha z^*\| \\ &= \|\alpha(x^* - z)\| \\ &= \alpha \, \|(x^* - z)\| \end{aligned}$$

Substituting $\alpha = \frac{R}{||x^* - z||}$, we find:

$$\begin{aligned} ||x^* - y|| &= \frac{R \, \|(x^* - z)\|}{||x^* - z||} \\ &= R \end{aligned}$$

Next we derive:

$$\begin{aligned} f(y) &= f((1-\alpha)x^* + \alpha z) \\ &\leq (1-\alpha)f(x^*) + \alpha f(z) \\ &< (1-\alpha)f(x^*) + \alpha f(x^*) \\ &= f(x^*) \end{aligned}$$

We have achieved a contradiction by selecting an α such that $f(y) < f(x^*)$ and $||x^* - y|| \leq R$, and the proof is complete. ■

4.6 Outline for the Rest of the Class

Figure 4.6 shows an outline for the rest of the class. Most easy stat/ML problems are convex. There are a few that are not, yet we still know how to solve them exactly. A few of these will be presented in class. However, the first two thirds of the course will cover convex problems. For the convex problems, we will cover first and second order methods as well as coordinate methods and dual methods. We will investigate

fast and useful algorithms for each class of problems and then have a discussion about how to chose which algorithm to use for a particular problem.

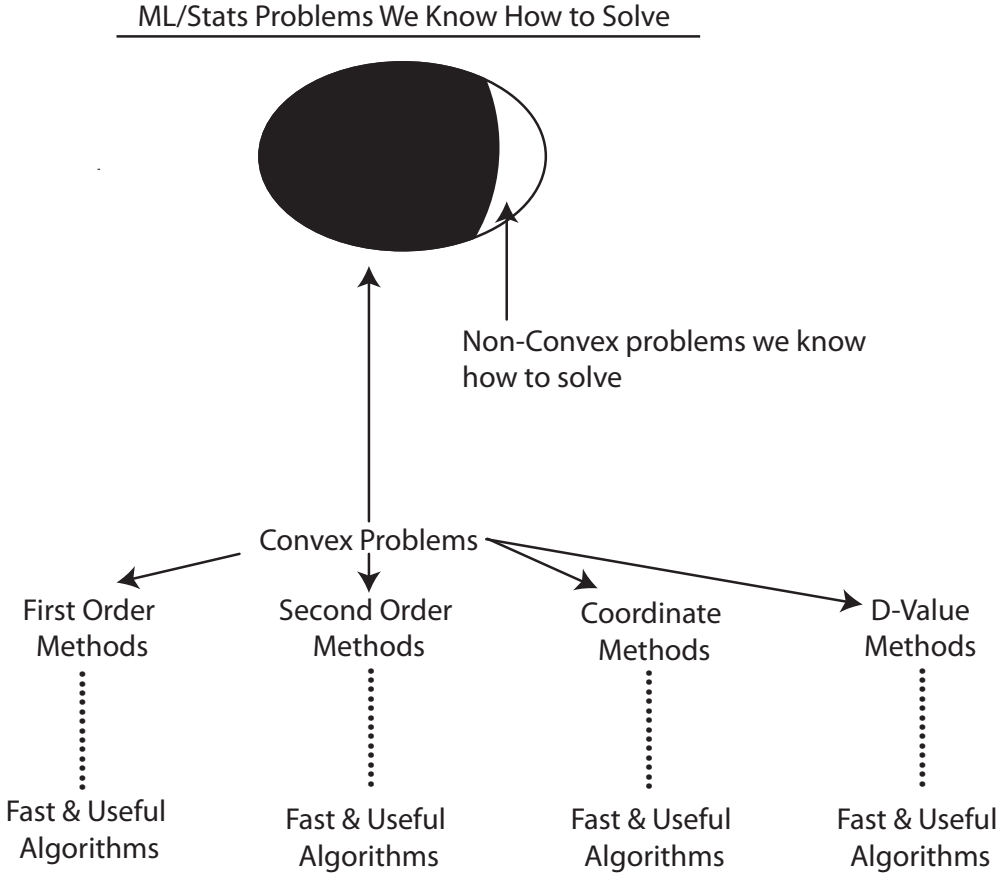

Figure 4.1: Outline for the rest of the class.