10-601B Recitation 11

Calvin McCarter

November 12, 2015

1 The Gaussian Kernel

The RBF (Gaussian) kernel is one of most popular general-purpose kernels. Here we show one reason for its popularity: it corresponds to an infinite-dimensional feature mapping. The Gaussian kernel is defined as follows:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{-\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2).$$

The Gaussian kernel encourages similarity between inputs based on their Euclidean distance. Because we're not dealing with probabilities which must sum to 1, we can omit the normalization constant found in the Gaussian distribution. But the shape is the same: we strongly encourage similarity among nearby inputs, while as inputs get further away, we encourage similarity by an amount that decreases exponentially to 0.

For simplicity, we prove that the feature mapping is infinite-dimensional for one-dimensional inputs where $2\sigma^2 = 1$:

$$\begin{aligned}
k(x, x') &= \exp\left(-(x - x')^2\right) \\
&= \exp(-x^2)\exp(-x'^2)\exp(2xx') \\
&= \exp(-x^2)\exp(-x'^2)\sum_{k=0}^{\infty}\frac{2^k x^k x'^k}{k!} && \text{using } \exp(z) = \sum_{k=0}^{\infty}\frac{z^k}{k!} \\
&= \sum_{k=0}^{\infty}[\phi(x)]_k[\phi(x')]_k, \qquad [\phi(x)]_k = \exp(-x^2)\sqrt{\frac{2^k}{k!}}x^k
\end{aligned}$$

Thus, $k(x, x')$ is an inner product in an infinite-dimensional space.

What is so great about this? This means that if all samples in our training set are unique, we can achieve 0 training error with a small enough bandwidth σ. In other words, underfitting will only come from poor features (leading to duplicate samples) or from poor choices for σ and C, rather than from the model itself. Meanwhile, computing the kernel $k(\mathbf{x}, \mathbf{x}')$ for $\mathbf{x} \in \mathbb{R}^d$ is just $O(d)$. Computationally, this is much better than adding non-linear features or using an explicitly non-linear classifier, which will typically be harder to train.

2 Convexity

For some simple machine learning models, like ridge regression, our estimated parameters given training data can be written in closed form. For many ML models, however, parameter estimation given a dataset requires solving an optimization problem. Optimization is an important topic when applying ML in the real world:

- Our choice of model often depends on our computational resources compared to the requirements of different models.
- Once we've chosen a model, analyzing our optimization problem can help us choose a fast, more accurate method.
- Optimization can give us insight into the properties of different ML methods.

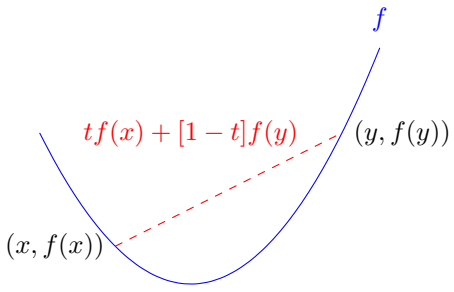
2.1 Convex optimization basics

Convex optimization problems are a special kind of optimization problems that are easier to solve and analyze. Convex optimization problems have the following form:

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ such that } x \in C,$$

where f is a convex function and the constraint region C is a convex set. A convex function has the following property:

$$f(t\mathbf{x} + [1-t]\mathbf{y}) \leq tf(\mathbf{x}) + [1-t]f(\mathbf{y}) \qquad 0 \leq t \leq 1.$$



Similarly, a convex set C is defined as follows:

$$\forall x, y \in C \Rightarrow tx + [1-t]y \in C, \qquad 0 \leq t \leq 1.$$

Another important term in convex optimization is "convex hull". For a set A (possibly non-convex), its convex hull is the smallest convex set that contains A.

Solving convex problems is much easier because once we find a local minimum, we're done: it's guaranteed to be a global minimum. Fortunately, some of the most common loss functions in ML (e.g. logistic loss, squared-error loss) are convex. Also, maximizing a concave function f can always be written as minimizing a convex function $-f$, so all of the nice properties of convex optimization apply to those problems as well.

3 Lagrangians and Duality

The standard form for constrained optimization problems is the following:

$$\begin{aligned} & \min_{x \in \mathbb{R}^n} f(x) \\ & \text{such that} \\ h_i(x) \leq 0 \qquad & i \in [1, m] \\ \ell_j(x) = 0 \qquad & j \in [1, r]. \end{aligned}$$

The Lagrangian is defined as a function of the "primal variables" x and the dual variables u and v:

$$L(x, u, v) = f(x) + \sum_{i=1}^{m} u_i h_i(x) + \sum_{j=1}^{r} v_j \ell_j(x).$$

We require $u \geq 0$, ie $u_i \geq 0 \forall i$. We then define the dual function to be:

$$g(u, v) := \min_x L(x, u, v),$$

where we've been able to drop the constraints on x from the original problem. The dual problem can be written in terms of the dual function:

$$\max_{u \geq 0, v} g(u, v).$$

Why would we care about solving the dual problem? It turns out that the dual function is a lower bound on solution to the primal problem:

$$\min_{x \in C} f(x) \geq g(u, v) \text{ if } u \geq 0,$$

so maximizing the dual gives another approach to solving the primal. It has some other nice properties:

- The dual function is concave, and since we're maximizing it, we can rewrite it as minimizing a convex function. Thus, the dual problem can be written as a convex problem, even if the primal can't!
- For lots of problems, the solutions to the dual and primal are exactly equal, $f^* = g^*$. ("strong duality")
- For some settings the dual may be nicer to solve.

3.1 Example: subset selection and Lasso

Convexity and Lagrangians give us the intuition for the Lasso problem. For feature selection, we might ideally like to solve the subset selection problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda \sum_i I\{\beta_i \neq 0\}$$

This is actually the Lagrangian form of the following constrained optimization problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 \text{ such that } \sum_i I\{\beta_i \neq 0\} \leq C.$$

The function to minimize is convex, but the set given by the constraint is non-convex. So instead we substitute in the convex hull of the constraint:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 \text{ such that } \sum_i I\{\beta_i \neq 0\} \leq C,$$

which gives us a convex problem. Writing this in Lagrangian form gives us our familiar Lasso problem:

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1.$$

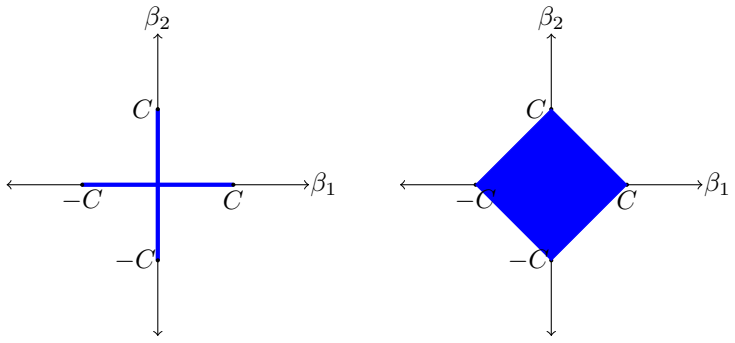


Figure 1: Illustration for two-dimensional problem: (left) Constraint region for $\sum I\{\beta_i \neq 0\} < C$. (right) Constraint region for $\sum |\beta_i| < C$.

4 SVMs: Dual and Duality

4.1 Deriving the dual

Our SVM primal problem is:

$$\begin{aligned} \min_{b,\mathbf{w},\xi} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n} \boldsymbol{\xi}_i \quad & \text{such that} \\ \boldsymbol{\xi}_i \geq 0, \quad & i \in [0,n] \\ y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \boldsymbol{\xi}_i, \quad & i \in [0,n]. \end{aligned} \tag{1}$$

Our constraints written in standard form are:

$$\begin{aligned} -\boldsymbol{\xi}_i &\leq 0 \\ 1 - \boldsymbol{\xi}_i - y_i(\mathbf{w}^T\mathbf{x}_i + b) &\leq 0. \end{aligned}$$

So our Lagrangian is:

$$\begin{aligned} L(b,\mathbf{w},\boldsymbol{\xi},\mathbf{v},\boldsymbol{\alpha}) =& \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\boldsymbol{\xi}_i + \sum_{i=1}^{n}\mathbf{v}_i(-\boldsymbol{\xi}_i) + \sum_{i=1}^{n}\boldsymbol{\alpha}_i\left(1 - \boldsymbol{\xi}_i - y_i(\mathbf{w}^T\mathbf{x}_i + b)\right) \\ =& \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\mathbf{1}^T\boldsymbol{\xi} - \mathbf{v}^T\boldsymbol{\xi} + \boldsymbol{\alpha}^T\mathbf{1} - \boldsymbol{\alpha}^T\boldsymbol{\xi} - \mathbf{w}^T[\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i] - b\boldsymbol{\alpha}^T\mathbf{y} \\ =& \boldsymbol{\alpha}^T\mathbf{1} + \frac{1}{2}\mathbf{w}^T\mathbf{w} + [C\mathbf{1} - \mathbf{v} - \boldsymbol{\alpha}]^T\boldsymbol{\xi} - \mathbf{w}^T[\sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i] - b\boldsymbol{\alpha}^T\mathbf{y} \end{aligned} \tag{2}$$

The SVM dual problem is defined as

$$\begin{aligned} g(\mathbf{v},\boldsymbol{\alpha}) &= \min_{b,\mathbf{w},\xi} L(b,\mathbf{w},\xi,\mathbf{v},\alpha) \\ \text{such that } \mathbf{v} \geq 0, \boldsymbol{\alpha} \geq 0 \quad & \text{ie. } \forall \mathbf{v}_i \geq 0, \boldsymbol{\alpha}_i \geq 0 \end{aligned} \tag{3}$$

To get rid of the primal variables, we'll need to solve $\min_{b,\mathbf{w},\boldsymbol{\xi}} L(b,\mathbf{w},\boldsymbol{\xi},\mathbf{v},\boldsymbol{\alpha})$ analytically, by setting gradients to $\mathbf{0}$:

$$\nabla_{\mathbf{w}}[L(b,\mathbf{w},\boldsymbol{\xi},\mathbf{v},\boldsymbol{\alpha})] = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n}\alpha_i y_i \mathbf{x}_i \tag{4}$$

$$\nabla_b[L(b,\mathbf{w},\boldsymbol{\xi},\mathbf{v},\boldsymbol{\alpha})] = -\sum_{i=1}^{n}\boldsymbol{\alpha}_i y_i = 0 \quad \Rightarrow \quad \boldsymbol{\alpha}^T\mathbf{y} = 0 \tag{5}$$

$$\nabla_{\boldsymbol{\xi}}[L(b,\mathbf{w},\boldsymbol{\xi},\mathbf{v},\boldsymbol{\alpha})] = C\mathbf{1} - \mathbf{v} - \boldsymbol{\alpha} = 0 \quad \Rightarrow \quad \boldsymbol{\alpha} = C\mathbf{1} - \mathbf{v} \tag{6}$$

While we didn't get formulas for b and $\boldsymbol{\xi}$ as we were expecting, we'll try plugging in the formula we got for $\mathbf{w}$. But first, let's define $\tilde{\mathbf{X}} := \text{diag}(y)\mathbf{X}$, so we can

write $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i = \widetilde{\mathbf{X}}^T \boldsymbol{\alpha}$. Now we plug into Equation 2:

$$\begin{aligned}
&\boldsymbol{\alpha}^T \mathbf{1} + \frac{1}{2}[\widetilde{\mathbf{X}}^T \boldsymbol{\alpha}]^T [\widetilde{\mathbf{X}}^T \boldsymbol{\alpha}] + [\mathbf{0}]^T \boldsymbol{\xi} - [\widetilde{\mathbf{X}}^T \boldsymbol{\alpha}]^T [\widetilde{\mathbf{X}}^T \boldsymbol{\alpha}] - b * 0 \\
=&\boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^T \boldsymbol{\alpha}
\end{aligned}$$

Thus, our dual is:

$$\begin{aligned}
\max_{\boldsymbol{\alpha}, \mathbf{v}} \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^T \boldsymbol{\alpha} \\
\text{such that } \boldsymbol{\alpha} =& C\mathbf{1} - \mathbf{v} \\
\mathbf{v} \geq& 0 \\
\boldsymbol{\alpha} \geq& 0
\end{aligned}$$

Finally, we typically simplify the constraints to remove $\mathbf{v}$.

$$\begin{aligned}
\max_{\boldsymbol{\alpha}, \mathbf{v}} \boldsymbol{\alpha}^T \mathbf{1} - \frac{1}{2} \boldsymbol{\alpha}^T \widetilde{\mathbf{X}} \widetilde{\mathbf{X}}^T \boldsymbol{\alpha} \\
\text{such that } 0 \leq \boldsymbol{\alpha}_i \leq C \ \forall i \in [1, n]
\end{aligned}$$

4.2 Discarding support vectors

So far, we've seen three different kinds of constraints which are satisfied by the solution to an optimization problem. First, there are the primal constraints, as in Equation 1. Second, there are the dual constraints, as given in Equation 3. Third, there are the "stationarity" constraints $\mathbf{w} = \widetilde{\mathbf{X}}^T \boldsymbol{\alpha}, \boldsymbol{\alpha}^T \mathbf{y} = 0$, and $\boldsymbol{\alpha} = C\mathbf{1} - \mathbf{v}$ that arise when we minimize the Lagrangian in terms of the primal variables. It turns out that there's another class of conditions that hold at the solution to the SVM optimization problem. These are called the "complementarity" (or "complementary slackness") conditions, as they relate the dual solution to the primal solution:

$$u_i h_i(x) = 0 \quad \forall i \in [1, m]$$

These 4 sets of conditions are called the Karush-Kuhn-Tucker (KKT) conditions. These 4 classes of conditions are guaranteed to hold if $f^* = g^*$ (strong duality) holds, which turns out to be the case for the SVM problem.

For the SVM problem, this implies the following:

$$\begin{aligned}
\mathbf{v}_i \boldsymbol{\xi}_i =& 0 \quad i \in [1, n] \quad \Rightarrow (C - \boldsymbol{\alpha}_i) \boldsymbol{\xi}_i = 0 \\
\boldsymbol{\alpha}_i (1 - \boldsymbol{\xi}_i - y_i(\mathbf{w}^T \mathbf{x}_i + b)) =& 0 \quad i \in [1, n]
\end{aligned}$$

We can use these conditions to consider different possible outcomes for a given sample j.

Suppose $\alpha_j = 0$. Then $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i = \sum_{i \neq j} \alpha_i y_i \mathbf{x}_i$. This means that at test time we can ignore the jth sample. Also, looking at the dual objective, we

see that its value will not change if we simply remove the jth coordinate of $\boldsymbol{\alpha}$ from the problem. So if we retrained our model without the jth sample, our solution would be the same.

But how do we know that we often have exactly $\alpha_j = 0$? Suppose $\alpha_j \neq 0$. By the second complementarity condition, we have

$$1 - \xi_j - y_j(\mathbf{w}^T\mathbf{x}_j + b) = 0.$$

Now either $\xi_j = 0$ or $\xi_j \neq 0$. If $\xi_j = 0$, we have $1 - y_j(\mathbf{w}^T\mathbf{x}_j + b) = 0$. Geometrically, this means that our sample lies exactly on the edge of the margin. If instead $\xi_j \neq 0$, by the first complementarity condition, we have $C - \alpha_j = 0 \Rightarrow \alpha_j = C$. Geometrically, this means that our sample lies on the wrong side of the margin. Because these are the only two possible cases, we can conclude that if our sample lies on the good side of the margin, $\alpha_j = 0$.

5 SVMs and LOOCV error

Suppose we train a linear SVM with no slack variables on a linearly separable training set with N samples and D dimensions. Suppose after training we have K support vectors. What is a upper bound on the leave-one-out cross-validation error?

Answer: $\frac{K}{N}$. On the full training set, we are guaranteed to have 0 training error since it was linearly separable. There are $N-K$ non-SVs, and each of these times we leave-out the training sample and re-train on the remaining samples. Because these are not support vectors, the weight vector will be identical to the original weight vector. Because the original weight vector had 0 training error, each of these $N - K$ times, the left-out samples will be classified correctly. On each of the K SVs, we may or may not classify the left-out sample correctly. So the total number of errors on left-out samples is at most K, and the LOOCV error is at most $\frac{K}{N}$.