CMSC 35900 (Spring 2009) Large Scale Learning **Lecture: 6**

Locally Weighted Regression

Instructors: Sham Kakade and Greg Shakhnarovich

1 NN in a subspace

A common pre-processing step is to project the data into a lower-dimensional subspace, before applying k-NN estimator. One example of this is the Eigenfaces algorithm for face recognition. PCA is applied on a database of face images (aligned, of fixed dimension) to get a principal subspace (of much lower dimensionality than the original, which is the number of pixels in the image). For some fixed m this means taking the m eigenvectors $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$ of $\mathbf{X}\mathbf{X}^T$ with the largest eigenvalues. Each face image is then represented by the vector of coefficients obtained by projecting it to the principal dimensions: $\mathbf{x}'_i = \mathbf{U}^T\mathbf{x}_i$. Given a test image, its coefficient vector $\mathbf{x}'_0 = \mathbf{U}^T\mathbf{x}_0$ is calculated, and classified using k-NN in this new m-dimensional representation.

2 Parametric vs nonparametric egression methods

We are now focusing on the regression problem. We will assume that the observations are generated by a process $y_i = f(\mathbf{x}_i) + \epsilon$, where the noise ϵ is independent of the data, and has zero mean and variance σ^2. Two "global" approaches to regression:

Parametric: assume parametric form $y = f(\mathbf{x};\, \theta)$, fit the parameters to the training set

$$\theta^* \;=\; \mathrm{argmin}_\theta \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i; \theta))$$

(for instance, using least squares procedure when L is squared loss) and then estimate

$$\hat{y}_0 \;=\; f(\mathbf{x}_0;\, \theta^*)$$

- Pros: Once trained, cheap to apply on any new data points. For many forms of f, a closed-form solution for θ^*
- Cons: A pretty strong assumption regarding the parametric form of f.

Non-parametric:

$$\hat{y}_0 \;=\; \sum_{i=1}^{n} y_i g(\mathbf{x}_i, \mathbf{x}_0).$$

A special case of this is a k-NN estimator, in which h is defined as

$$g(\mathbf{x}_i, \mathbf{x}_0) = \begin{cases} 1/k & \text{if } \mathbf{x}_i = \mathbf{x}_{(j)}(\mathbf{x}_0) \text{ for some } 1 \le j \le k, \\ 0 & \text{otherwise.} \end{cases}$$

More generally, g may itself be a parametric function. For instance, Gaussian kernel

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

requires settings for the kernel bandwidth σ^2; these settings affect the properties of the estimate (its smoothness in particular).

- Pros: very flexible, does not assume any particular form of f.
- Cons: very expensive: for every test set, need to go over all the training examples to compute the kernel (find the neighbors, in the case of k-NN). Still subject to parameter fitting, in case of parametric kernels.

3 Kernel regression

An estimator that emphasizes points closer to the query can be expressed as

$$\hat{y}_0^{NW} = \sum_{i=1}^{n} s_i(\mathbf{x}_0) y_i, \tag{1}$$

This form is called a linear smoother. One way to define s_i for 1D $\mathbf{x}$ is to set

$$s_i(\mathbf{x}_0) = \frac{K(\frac{\mathbf{x}_i - \mathbf{x}_0}{h})}{\sum_j K(\frac{\mathbf{x}_j - \mathbf{x}_0}{h})}$$

and K is a positive definite continuous kernel satisfying, for any u,

$$K(u) \ge 0 \tag{2}$$

$$\int K(u) du = 1, \tag{3}$$

$$\int u K(u) du = 0, \tag{4}$$

$$\int u^2 K(u) du > 0, \tag{5}$$

and $h > 0$ is a parameter effectively controlling the distance falloff. This estimator is called the Nadaraya-Watson (N-W) kernel estimator. Kernels that are particularly popular for this model are the Epanechnikov kernel,

$$K(u) = \begin{cases} 1 - (u)^2 & \text{if } |u| < 1, \\ 0 & \text{otherwise} \end{cases}$$

and the tri-cubic kernel

$$K(u) = \begin{cases} \left(1 - |u|^3\right)^3 & \text{if } |u| < 1, \\ 0 & \text{otherwise,} \end{cases}$$

and the "boxcar" kernel

$$K(u) \;=\; \begin{cases} 1 & \text{if } |u| < 1, \\ 0 & \text{otherwise} \end{cases}$$

which effectively defines a distance cutoff, determined by the value of h. All of these have finite support, and potentially allow for huge savings if we have a way to find the points with non-zero kernel values without an exhaustive scan of the training data. Theoretically, the Epanechnikov kernel has certain optimality properties; on the other hand, it suffers from discontinuity at the support boundaries, which may cause problems in practice as well as in theoretical analyses.

In a multivariate case, the kernel is parametrized by a symmetric positive-definite matrix $\mathbf{H}$, such that

$$K(\mathbf{x}_i\mathbf{x}_0;\mathbf{H}) \;=\; |\mathbf{H}|^{-1/2} K\left(\mathbf{H}^{-1/2}(\mathbf{x}_i - \mathbf{x}_0)\right)$$

Often the multivariate kernel is a product of 1D kernels,

$$K(\mathbf{x}_i\mathbf{x}_0;\mathbf{H}) \;=\; \prod_{q=1}^{d} K((x_{i,q} - x_{0,q})/h_q),$$

where $x_{i,q}$ is the q-th element of $\mathbf{x}_i$, and $\mathbf{H} = diag(h_1, \ldots, h_d)$. The conditions on a valid K become

$$\int \mathbf{u}\mathbf{u}^T K(\mathbf{u})d\mathbf{u} \;=\; v\mathbf{I}, \quad v \neq 0 \tag{6}$$

$$\int u_1^{l_1} \cdots u_d^{l_d} K(\mathbf{u})d\mathbf{u} \;=\; 0, \quad \text{for all odd } l_1, \ldots, l_d. \tag{7}$$

The specific choice of the kernel K turns out not to be very important. However, the value of the bandwidth h has an effect on the behavior of the estimate. It is possible to derive an optimal value of h that will minimize the risk of the estimator; however, this value depends on the unknown true function f (its derivatives, more precisely), as well as the input probability density p. We can try to estimate p, as well as the derivatives of f, directly; however, this of course leads to a new bandwidth selection problem.

A common way to estimate h in practice is via cross-validation: for each training example $\mathbf{x}_i$, remove it from the set, fit the function at $\mathbf{x}_i$ using the remaining points, compute the residual with respect to y_i, and average the squared residuals over the n samples. This sounds rather expensive, since it requires re-calculating the function parameters for each removed sample. Fortunately, for any linear smoother this can be done in closed form. We compute, for each i and j, the weights $s_i(\mathbf{x}_j)$. Then,

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}_{(-i)}(\mathbf{x}_i)\right)^2 \;=\; \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - s_i(\mathbf{x}_i)}\right)^2 \tag{8}$$

This allows an efficient estimate of a global bandwidth. The value of h carries a bias-variance tradeoff: larger h means smoother, less varying estimate, however it may be affected too much by distant points, increasing the bias. It has been proposed by many researchers that a better solution than to use a global h is to allow an adaptive h. A particularly popular method is to set h to be the distance to the k-th nearest neighbor.

4 Locally polynomial regression

The N-W estimator employs a truly non-parametric model. In fact, this is a locally constant estimator: it models the function at $\mathbf{x}_0$ (and in its infinitisemal vicinity) as a constant, defined by the weighted combination of the training labels. We now consider a parametric model in which we explicitly model the variation of the function at $\mathbf{x}_0$, but emphasize nearby points more heavily than those far away. During training, we can do that by weighting the training examples:

$$\theta^*(\mathbf{x}_0) \;=\; \sum_{i=1}^{n} L(y_i, f(\mathbf{x}_i;\theta))K(D(\mathbf{x}_i - \mathbf{x}_0; h)/h) \tag{9}$$

Note that here we use a more general notation for the kernel, referring to an arbitrary distance function D.

For squared loss, and linear model, we get an analytical solution as follows. First, we shift the $\mathbf{x}$s by subtracting the query $\mathbf{x}_0$ from each $\mathbf{x}_i$ and replacing $\mathbf{x}_0$ with $[0,\ldots,0,1]^T$ (the 1 is for the constant term). We set up an $n \times d+1$ matrix $\mathbf{X}$ by placing the shifted $\mathbf{x}_i^T$ in its rows. We denote

$$w_i \;\triangleq\; \sqrt{K(\mathbf{x}_i, \mathbf{x}_0)}$$

and

$$\mathbf{W} \;\triangleq\; \begin{bmatrix} w_1 & 0 & \ldots & 0 \\ 0 & w_2 & \ldots & 0 \\ & \ddots & & \\ 0 & 0 & \ldots & w_n \end{bmatrix}$$

We then reweight the original data, both $\mathbf{x}$ and y: $\mathbf{z}_i \;\triangleq\; w_i\mathbf{x}_i$, so that $\mathbf{Z} = \mathbf{W}\mathbf{X}$, and $v_i \;\triangleq\; w_i y_i$, so that $\mathbf{v} = \mathbf{W}\mathbf{y}$. Now, we have

$$\begin{aligned} & \theta(\mathbf{x}_0) \;=\; \sum_{i=1}^{n}\left(y_i - \mathbf{x}_i^T\theta\right)^2 K(\mathbf{x}_i,\mathbf{x}_0) \\ \Rightarrow\quad & (\mathbf{Z}^T\mathbf{Z})\theta \;=\; \mathbf{Z}^T\mathbf{v} \qquad (10) \\ \Rightarrow\quad & \theta^*(\mathbf{x}_0) \;=\; (\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v} \\ \Rightarrow\quad & \hat{y}_0 \;=\; \mathbf{x}_0^T(\mathbf{Z}^T\mathbf{Z})^{-1}\mathbf{Z}^T\mathbf{v} \end{aligned}$$

Note that we have to do this for each test example $\mathbf{x}_0$ producing a different parameter vector $\theta^*(\mathbf{x}_0)$, and so even if we use a locally linear model, the resulting function is no longer globally linear!

In practice, especially in high dimensions, the regression matrix $\mathbf{Z}^T\mathbf{Z}$ can be singular. The usual solution is to use ridge regression:

$$\theta^* \;=\; (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{v}$$

An alternative to this is to perform a weighted dimensionality reduction, by applying SVD on $\mathbf{Z}^T\mathbf{Z}$.

In terms of (1), we can define

$$[s_1(\mathbf{x}_0),\ldots,s_n(\mathbf{x}_0)]^T \;=\; \mathbf{x}_0^T\left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}\right)^{-1}\mathbf{Z}^T\mathbf{W}.$$

If we want to use a locally polynomial model of order higher than 1, we will need to replace the design matrix $\mathbf{X}$ with the degree-extended matrix, which includes in each column, in addition to $\mathbf{x}$, features obtained by higher-order polynomial terms: $x_1^2, x_1x_2, \ldots$

5 Bias and variance

In case when the kernel and the distance function have the form that is applicable to both N-W and the local linear estimator, we can characterize the asymptotic behavior of these estimators as follows. Details can be found in [2].

Theorem 5.1. *(Fan, 1992) Let $y_i = f(\mathbf{x}_i) + \epsilon_i$, for $i = 1, \ldots, n$, with $\mathbf{x}_i \sim p(\mathbf{x})$ i.i.d. over a bounded support, with additive noise $\epsilon_i \sim p(\epsilon)$ i.i.d., with zero mean and variance σ^2. Furthermore, for $\mathbf{x}_0$, assume $p(\mathbf{x}_0) > 0$, p is continuously differentiable at $\mathbf{x}_0$, and all second-order derivatives of f are continuous in $\mathbf{x}_0$. Consider a sequence of bandwidth matrices $\mathbf{H}$ such that, as $n \to \infty$, $|\mathbf{H}|/n \to 0$, as well as each entry of $\mathbf{H}$. We will also assume that the condition number (ratio of the largest to the smallest eigenvalue) of $\mathbf{H}$ is bounded from below by some positive constant, for all n. Then, both the N-W and the local linear estimator have variance in $\mathbf{x}_0$ (conditional on $\mathbf{x}_1, \ldots, \mathbf{x}_n$)*

$$\frac{1}{n|\mathbf{H}|^{1/2}} \frac{\sigma^2}{p(\mathbf{x}_0)} \int K(\mathbf{u})^2 d\mathbf{u} \, (1 + o_P(1)) \tag{11}$$

The bias of N-W kernel estimator is

$$\frac{1}{2} v \, \mathrm{tr}\left(\mathbf{H}\mathcal{H}_f(\mathbf{x}_0)\right) + v \frac{\nabla_f(\mathbf{x}_0)^T \mathbf{H}\mathbf{H}^T \nabla_p(\mathbf{x}_0)}{p(\mathbf{x}_0)} + o_P(\mathrm{tr}(\mathbf{H})), \tag{12}$$

where v is defined as in (6), $\mathcal{H}_f(\mathbf{x})$ stands for the Hessian of f in $\mathbf{x}$, and $\nabla_f(\mathbf{x})$ for the gradient. The bias of the local linear estimator is

$$\frac{1}{2} v \, \mathrm{tr}\left(\mathbf{H}\mathcal{H}_f(\mathbf{x}_0)\right) + o_P\left(\mathrm{tr}(\mathbf{H})\right). \tag{13}$$

This shows that the N-W estimator suffers from "design bias": it depends on the density of $\mathbf{x}$, while the locally linear estimator does not. Furthermore, it can be shown that the the bias of the N-W is higher at the boundaries. These results hold in general, for local polynomial models of order t. Interestingly, using odd t vs. the next even p reduces bias without increasing variance. Note however that these results are asymptotic!

For finite-sample case we can write the estimate in $\mathbf{x}_0$ as

$$\hat{y}_0 = \mathbf{s}(\mathbf{x}_0)^T \mathbf{y}.$$

The conditional mean of the locally linear estimator is given exactly by

$$E_{\mathbf{x}_1,\ldots,\mathbf{x}_n}[\hat{y}_0] = E_{\mathbf{x}_1,\ldots,\mathbf{x}_n}\left[\mathbf{s}(\mathbf{x}_0)^T \mathbf{y}\right] = \sum_{i=1}^{n} s_i(\mathbf{x}_0) f(\mathbf{x}_i),$$

and the variance is

$$\mathrm{var}\, \hat{y}_0 = \sigma^2 \sum_{i=1}^{n} s_i^2(\mathbf{x}_0) = \sigma^2 \|\mathbf{s}(\mathbf{x}_0)\|^2$$

Note that this estimate does not depend on the labels y, but only on the $\mathbf{x}$s!

References

[1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[2] D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *Annals of Statistics*, 22(3):1346–1370, 1994.

[3] L. Wasserman. *All of Nonparametric Statistics*, chapter 5. Springer, 2006.