ECE 544NA: Pattern Recognition

Lecture 9: Multiclass Classification and Kernel Methods (Sep 27 and Oct 2)

Lecturer: Alexander Schwing Scribe: Mohammad Almasri

1 Objectives

So far in this class, we discussed different techniques to perform binary classification with two classes only. In this lecture, we extend the classification topic and discuss multi-class classification with more than two classes. More information can be found in [1], chapter 14.

2 Recap from previous lectures

For binary classification, we discussed three frameworks: linear regression, logistic regression, and support vector machines (SVM). Linear regression is the simplest and its cost/loss function to minimize for w is formulated as shown in figure 1. We refer to $w^T\phi(x^{(i)})$ as function F which depends on w and x. Up to this lecture, we assume that function F is linear but this constraint will be relaxed in future lectures. Figures 2 and 3 show the cost/loss functions for logistic regression (log loss) and SVM (hinge loss).

$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\mathcal{D}} \frac{1}{2}(1 - y^{(i)} \underbrace{\boldsymbol{w}^T\phi(x^{(i)})}_{F(x^{(i)},\boldsymbol{w})})^2$$

Figure 1: Binary linear regression cost/loss function with regularization C.

$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\mathcal{D}} \log\left(1 + \exp(-y^{(i)} \underbrace{\boldsymbol{w}^T\phi(x^{(i)})}_{F(x^{(i)},\boldsymbol{w})})\right)$$

Figure 2: Binary logistic regression cost/loss function with regularization C.

$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\mathcal{D}} \max\{0, \underbrace{1}_{\text{taskloss}} - y^{(i)} \underbrace{\boldsymbol{w}^\top\phi(x^{(i)})}_{F(x^{(i)},\boldsymbol{w})}\}$$

Figure 3: Binary SVM cost/loss function with regularization C.

Figure 4 shows the three loss functions plotted. The x-axis represents $y_i F_i(x, w)$. The y-axis plots the loss for a single data point. Ideally, the perfect loss is the 0/1 loss function, however, it has a derivative of zero everywhere which makes hard to optimize. The quadratic loss of linear regression highly penalizes for the points that are easy to predict. Log and hinge loss do very well for the easy-to-predict points.

Logistic and SVM loss functions can be generalized into a single framework. To combine these two functions, we use the following limit: $\lim_{\epsilon\to} \epsilon \log(1 + \exp\frac{-F}{\epsilon})$. When F is larger than zero, then the

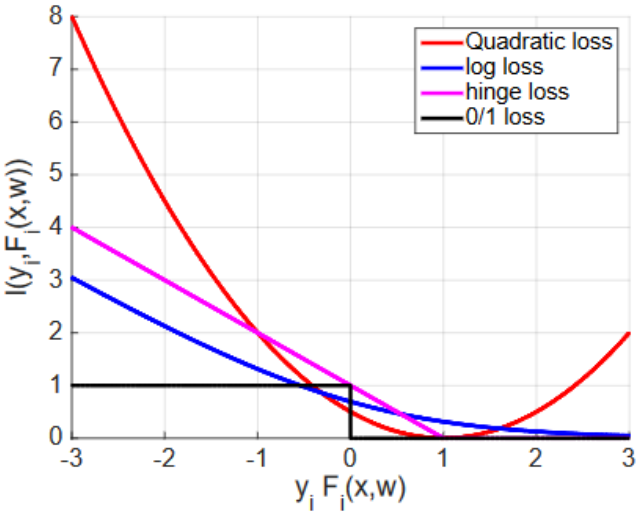

Figure 4: Loss functions plotted for binary frameworks.

limit goes to zero. When F is less than zero, we get 0 by inf. Solving this indefinite limit using L'Hopital's rule gives -F as final result. By combining the two results, $\lim_{\epsilon \to} \epsilon \log(1 + \exp \frac{-F}{\epsilon}) = max(0, -F)$. Using this max term, we formulate a general binary classification framework as shown in figure 5. Using $\epsilon = 1$, we return back to logistic regression (assuming that L=0). Using $\epsilon = 0$, we use the result of the previous limit to return back to SVM (assuming L=1). Figure 6 shows how the value of ϵ controls the log loss (general binary loss function) curve. Using small values of ϵ approximates the hinge loss smoothly without having non-differentiability as in hinge loss.

$$\min_{\boldsymbol{w}} \frac{C}{2} \|\boldsymbol{w}\|_2^2 + \sum_{i \in \mathcal{D}} \epsilon \log \left(1 + \exp \left(\frac{L - y^{(i)} \boldsymbol{w}^T \phi(x^{(i)})}{\epsilon} \right) \right)$$

Figure 5: General binary classification loss function. Logistic regression loss is used when $\epsilon = 1$ and SVM loss is used when $\epsilon = 0$.

3 Multi-class Classification

The binary classification is useful when the final answer belongs to one of two classes only. For example, object detection in images answers the question whether or not an object is in the image. Multi-class classification is the problem of classifying inputs into one of three or more classes. There are different strategies to perform multi-class classification using K classes. In this lecture, we discuss multi-class classification using transformation to binary, multi-class logistic regression, and multi-class SVM. Then, we construct a general framework that subsumes the logistic regression and SVM. Lastly, we talk about kernels as a method to deal with non-linear features.

3.1 Transformation to binary

The first approach to classify between K classes is to reduce the problem to multiple binary problems. This approach can be categorized into a) one vs. all or 1 vs. rest and b) one vs. one.

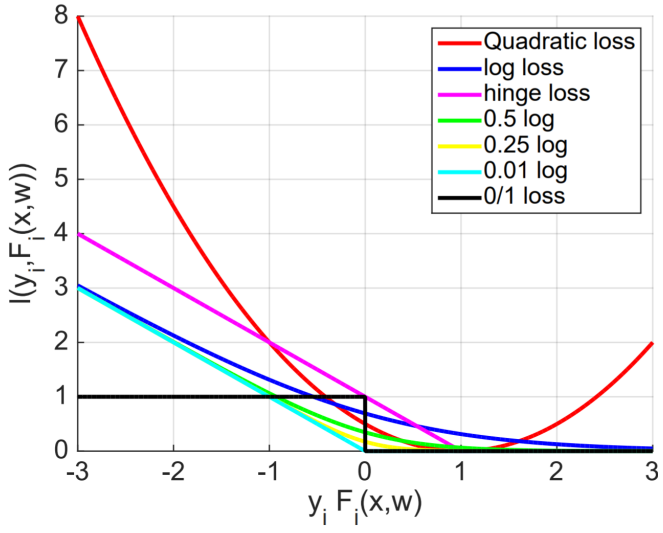

Figure 6: Loss functions plotted using the general classification framework. Different values of ϵ is used.

3.1.1 One vs. all

This technique uses K-1 binary classifiers. For the kth classifier, decide whether or not a point is in class k (give 1 if it belongs to class k and give -1 if it does not belong to class k). Figure 7 shows an example on one vs. all classification. Classifier C1 decides whether or not a point belongs to region/class R1. Classifier C2 decides whether or not a point belongs to region/class R2. If a point is in class R3, both C1 and C2 give a negative answer. However, if a point is in the green region, then the approach fails to predict because there is no way to tell whether this point belongs to R1 or R2. Therefore, one to all classification suffers from either more than good answer or no good answer prediction.

3.1.2 One vs. One

This classifier uses $\binom{K}{2}$ = K(K-1)/2 binary classifiers; a classifier for each possible pair of classes. For example, if we have 4 classes cat, dog, rabbit, mouse, we will need 6 classifiers to perform one vs. one classification cat vs. dog, cat vs. rabbit, ..., rabbit vs. mouse. Each binary classifier is trained to distinguish between its assigned two classes. At prediction, a voting strategy is applied; all K(K-1)/2 classifiers are applied to new data points and the class that achieves highest score (highest number of +1) wins and gets predicted. Figure 8 shows an example of three classes problem. Using one vs. one classifier, we need 3 binary classifiers: C1 vs. C2, C1 vs. C2, and C2 vs. C3. For R1, R2, and R3 the voting system works well. However, there is an ambiguity in the green region due to voting ties that occurs because the two way preferences are not necessarily transitive. The C1 vs. C2 classifier prefers C2 over C1. C2 vs. C3 classifier prefers C3 over C2. If the binary preferences were transitive, we expect that the final classifier to prefer C3 over C1, but it doesn't. That classifier prefers C1 over C3 and this is why we get a tie in the scoring mechanism.

4 Multi-class classification using probability model

For the binary logistic regression, we used the binomial distribution to build our model. The binomial distribution models the outcomes of independent events that have one of two values

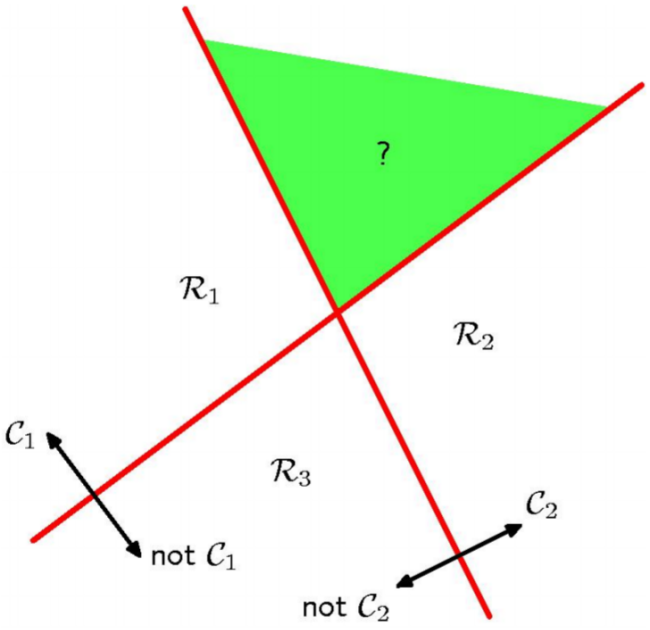

Figure 7: One-vs-all multi-class classification.

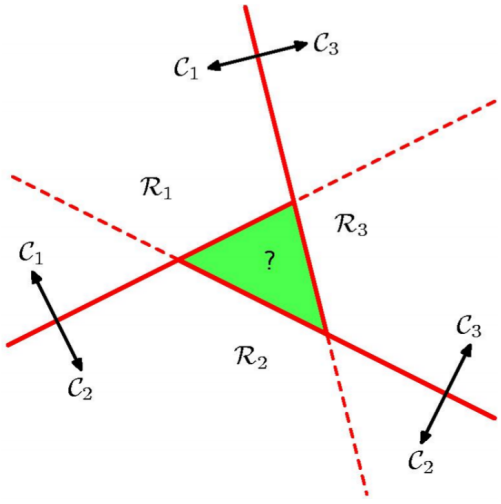

Figure 8: One-vs-one multi-class classification.

only (suitable for binary classification). The multinomial distribution models the outcomes of independent events that have one of three or more values. Thus, we use the multinomial distribution for the multi-class classification over k $\in$ {0, 1, ..., K-1}. With this model, $p(y = k|x)$ represents the score (logit) of class k given an input x. Since this is a probability model, the sum of all the scores for a given input x should equals to one ($\sum_{k=0}^{k=K-1} p(y = k|x) = 1$). We can satisfy this constraint by introducing a weight vector for each class and formulate the scoring function as a parameterized multinomial distribution (shown in figure 9). To get the score of $p(y = k|x)$, we exponentiate $w_{(k)}^T \phi(x^i)$ and divide it by the sum of all the scores. To find the largest score, we need to maximize the log likelihood of scores assuming our samples are independent and identically

distributed (i.i.d) as shown in figure 10. Figure 10 also shows how we transform the maximization problem into a minimization problem using minus-log. The minimization problem can be solved using, for example, gradient descent.

$$p(y = k|x^{(i)}) = \frac{\exp \boldsymbol{w}_{(k)}^{\top} \phi(x^{(i)})}{\sum_{j \in \{0,1,\ldots,K-1\}} \exp \boldsymbol{w}_{(j)}^{\top} \phi(x^{(i)})}$$

Figure 9: Parameterized multinomial distribution to model a multi-class classification problem.

$$\arg\max_{\boldsymbol{w}} \prod_{(x^{(i)},y^{(i)}) \in \mathcal{D}} p(y = y^{(i)}|x^{(i)}) = \arg\min_{\boldsymbol{w}} \sum_{(x^{(i)},y^{(i)}) \in \mathcal{D}} -\log p(y = y^{(i)}|x^{(i)})$$

Figure 10: Finding the maximum score by maximizing the log likelihood of samples assuming that samples are i.i.d. This maximization can be transformed to a minimization using minus-log function.

To understand what weights really mean, figure 11 shows an example of 2-dimensional example (actually 3-dimensional but $\phi 3$ always equals to one). The first and second dimensions of our feature vector is the x-axis and y-axis, respectively. Then, we have three classes: red, green, and blue. Thus, we have three weight vectors each is 3-dimensional (since ϕ is also 3-dimensional). Gradient descent is used to optimize and generate three colored lines ($w_{(k)}^{T}\phi(x^{i}) = 0$). For each data point, we compute the score or the distance from each line (by multiplying ϕ of each point by its corresponding w vector). The larger the distance from a line is the higher the score for the class represented by that line. For the red line, for example, we would easily guess that the red points have positive high score and are classified to red whereas the green and blue points have negative score. For the blue line, red points have negative scores, blue points have small positive score, and green have large positive score. Lastly, $\phi 3$ is set one to create a bias from origin.

If we have two classes, we learn two weight vectors and get two decision boundaries that may or may not coincide. However, the decision boundary is the same as the binary logistic regression.

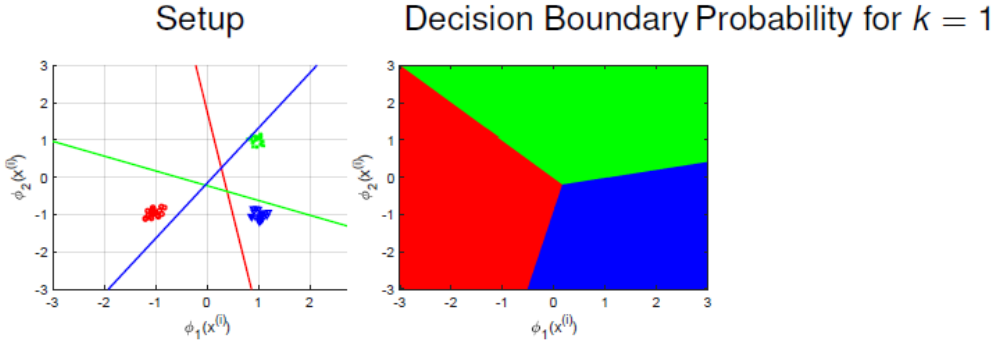

Figure 11: Example on multi-class classification.

5 Multi-class SVM

For the binary SVM, we maximized the margin between the two lines that separate two classes. We also introduced a slack variable when the points are not linearly separable. Then, we formulated the

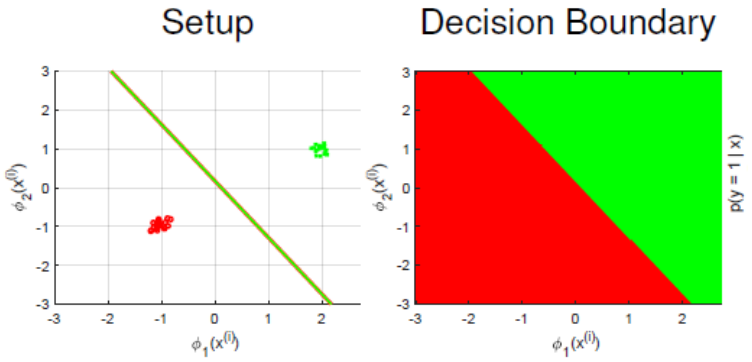

Figure 12: Example for using the multi-class model to do binary classification. The two weight vector lines coincide creating two decision regions.

problem as a minimization problem with some constraints. For the multi-class SVM, we introduce K weight vectors and we want the score for the ground truth class to be larger than the score of any other class possibly by some margin. Figure 13 formulate this objective. Using this constraint (objective), we formulate the loss function of the multi-class SVM as shown in figure 14. In this equation, we set the margin to one and we introduce a slack variable to guarantee feasibility of the model. Also, the w without indexing refers to all the w vectors concatenated into one large vector.

$$\boldsymbol{w}_{y^{(i)}}^T \phi(x^{(i)}) \geq \boldsymbol{w}_{\hat{y}}^T \phi(x^{(i)}) \qquad \forall i \in \mathcal{D}, \hat{y} \in \{0, 1, \ldots, K-1\}$$

Figure 13: The objective of muti-class SVM. We use larger or equal operator to include all classes.

$$\min_{\boldsymbol{w}, \xi^{(i)} \geq 0} \frac{C}{2} \|\boldsymbol{w}\|_2^2 + \sum_{i \in \mathcal{D}} \xi^{(i)} \quad \text{s.t.} \quad \boldsymbol{w}_{y^{(i)}}^T \phi(x^{(i)}) - \boldsymbol{w}_{\hat{y}}^T \phi(x^{(i)}) \geq 1 - \xi^{(i)} \quad \forall i, \hat{y}$$

Figure 14: The loss/cost function of muti-class SVM.

To minimize the loss function of the multi-class SVM (figure 14), we first simplify the notation (avoid all the subscripts) by introducing the vector ψ that stacks up all the ϕ vectors as shown in figure15. Each entry of ψ is the original ϕ multiplied by the indicator function $\delta(y^{(i)} = k)$ that have the value of zero everywhere except for that block that corresponds to the ground truth that equals to k. For example, if the ground truth equals to 2, then the vector of $y^{(i)} = 2$ keeps its ϕ values and all other vectors ($y^{(i)} = 0$, $y^{(i)} = 1$, $y^{(i)} = 3$, ..., $y^{(i)} =$ k-1) are set to zero. Note that ψ now depends also on y to find the value of the ground truth.

$$\boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_0 \\ \vdots \\ \boldsymbol{w}_{K-1} \end{bmatrix} \qquad \psi(x^{(i)}, y^{(i)}) = \begin{bmatrix} \phi(x^{(i)}) \delta(y^{(i)} = 0) \\ \vdots \\ \phi(x^{(i)}) \delta(y^{(i)} = K-1) \end{bmatrix}$$

Figure 15: Combine all the the weight vectors into single large w. Combine all the feature vectors into a single vector ψ.

Using the new notation, we use an equivalent loss function shown in figure 16

$$\min_{\boldsymbol{w},\xi^{(i)}\geq 0} \frac{C}{2}\|\boldsymbol{w}\|_2^2+\sum_{i\in\mathcal{D}}\xi^{(i)} \quad \text{s.t.} \quad \boldsymbol{w}^T\left(\psi(x^{(i)},y^{(i)})-\psi(x^{(i)},\hat{y})\right)\geq 1-\xi^{(i)} \quad \forall i,\hat{y}$$

Figure 16: Equivalent loss function using the new notation of concatenated all the w_i vectors in a single large w and the ψ vector that has all the ϕ vectors (the values of this vector is controlled using indicator function).

This new loss function can be solved using gradient descent without dealing with constraints (we used Lagrange to deal with constraints terms). We know from the binary SVM that the slack variable can have one of two values: a) zero, in case we classify correctly and b) $1-y^{(i)}w^T\phi(x^{(i)}$, in case we classify incorrectly (recall that this value represents the distance from the line that separates the two classes). Then we combine these two values using the max operator.

Using the same approach, we can rewrite our loss function as shown in figure 17. Note that we have $\hat{y}$ which we maximize it and the $y^{(i)}$ which is the ground truth y that the indicator functions use. We can also remove the outer max operator, since the inner max operator term is always larger than or equal to zero. Then we take out the terms that do not depend on $\hat{y}$ out of the max operator. The rearranged loss function in shown in figure 18.

$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2+\sum_{i\in\mathcal{D}}\max\{0,\max_{\hat{y}}(1-\boldsymbol{w}^T(\psi(x^{(i)},y^{(i)})-\psi(x^{(i)},\hat{y})))\}$$

Figure 17: The equivalent multi-class SVM loss function without the slack variable.

$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2+\sum_{i\in\mathcal{D}}\underbrace{\max_{\hat{y}}(1+\boldsymbol{w}^T\psi(x^{(i)},\hat{y}))}_{\text{Loss-augmented inference}}-\boldsymbol{w}^T\psi(x^{(i)},y^{(i)})$$

Figure 18: The equivalent multi-class SVM loss function rearranged.

Multi-class logistic regression and SVM can be generalized using the same limit equation as shown in the recap section ($\lim_{\epsilon\to}\epsilon\log(1+\exp\frac{-F}{\epsilon})$). Therefore, the final equation is shown in figure 19. This equation represents a generalized framework for multi-class classification that can go back to either the logistic regression or SVM based of the values of ϵ and L. Also, we can revert it to binary classification by replacing the ψ with ϕ and making y two-way.

$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2+\sum_{i\in\mathcal{D}}\epsilon\ln\sum_{\hat{y}}\exp\frac{L(y^{(i)},\hat{y})+\boldsymbol{w}^T\psi(x^{(i)},\hat{y}))}{\epsilon}-\boldsymbol{w}^T\psi(x^{(i)},y^{(i)})$$

Figure 19: The generalization loss function for multi/binary logistic regression and SVM.

6 Kernels

There is a possible limitation in our previous general framework; which is the inner product of the linear features (or linear transformations of features). In another word, where are just learning lines in the feature space that determine the scores. One approach to solve the linearity problem is the usage of kernels.

In order to see how why and how we can use kernels, we need to briefly review the dual of logistic regression and SVM functions. Figure 20 show the dual functions and how to predict using the dual variables for logistic regression and SVM. We notice that there are many inner products $\phi^T(x^{(i)})\phi(x^{(j)})$ between different samples that appear in the two prediction functions. Thus, the kernel trick states that we replace the inner product of ϕs with kernel $\kappa(x^{(i)}, x^{(j)})$. The recite of the kernel should be equal to some $\phi^T(x^{(i)})\phi(x^{(j)})$. The advantage of using kernel is that we do not need to construct the feature vector $\phi(x)$ (neither in the prediction nor in the optimization) but we need to ensure that κ is a valid kernel such that it corresponds to an inner product in some feature space. So, we cannot replace with any function. We need to ensure that a feature space would exist such that the result of κ is identical to transforming the x to that feature space and then computing the inner product $\phi^T(x^{(i)})\phi(x^{(j)})$.

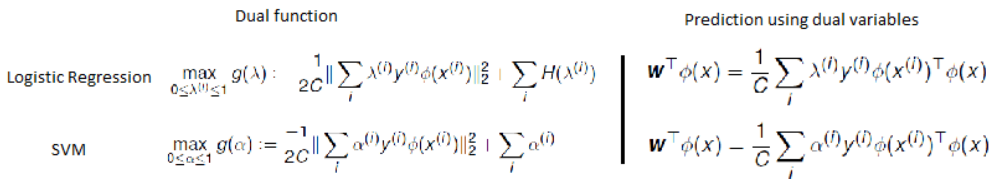

Figure 20: The dual functions and how to predict using the dual variables for logistic regression and SVM.

Figure 21 shows an example that explains what it means that there is an inner product and the inner product has to exist. κ takes to inputs x and z. The operation is the square of the inner product between x and z (the result is single value). So the question is: is this a valid kernel? Can we find a feature space in which we can represent this as an inner product. Assuming that the two vectors are 2 dimensional (x:{x1, x2}, z:{z1, z2}). Then, we do the inner product and apply the square operator. We can rearrange the equation into two vectors each vector depends only on a single vector. Thus, each vector is a transformation function ($\phi(x)$ and $\phi(z)$) of single vectors. So, why kernels are useful? There are feature spaces where we can compute κ easily, but ϕ would be infinite dimension which means that we cannot compute it completely even though we know it exists. Therefore, kernels allow performing transformations that cannot be applied in the primal domain.

$$\begin{aligned}
\kappa(x, z) &= (x^\top z)^2 \\
&= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\
&= \begin{bmatrix} x_1^2 & \sqrt{2}x_1 x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} z_1^2 & \sqrt{2}z_1 z_2 & z_2^2 \end{bmatrix}^\top \\
&= \phi(x)^\top \phi(z)
\end{aligned}$$

Figure 21: Kernel example.

There two ways to test the validity of kernels without explicitly constructing feature vectors. The first is to construct Gram matrix $(K)_{i,j} = \kappa(x^i, y^i)$ and check if it is positive definite $\forall x^i, x^j$. The second is to decompose/compose kernel into/from known kernels. Figure 22 shows three example kernels that are valid. There are several techniques to construct a new kernel form existing ones, figure 23 shows five techniques. With such techniques, we can construct large number of kernels.

7 Next lecture

In the next lecture, we will talk about deep nets which is another way of getting non-linearity to our frameworks.

1. Linear Kernel:

$$\kappa(x^{(i)}, x^{(j)}) = {x^{(i)}}^\top x^{(j)}$$

2. Squared exponential (Gaissian) kernel:

$$\kappa(x^{(i)}, x^{(j)}) = \exp\left(-\frac{1}{2}(x^{(i)} - x^{(j)})^\top \Sigma^{-1}(x^{(i)} - x^{(j)})\right)$$

3. Sigmoid kernel:

$$\kappa(x^{(i)}, x^{(j)}) = \tanh(a \cdot {x^{(i)}}^\top x^{(j)} + b)$$

Figure 22: Example kernels that are valid.

1. Positive Scaling

$$\kappa(x^{(i)}, x^{(j)}) = c\kappa_1(x^{(i)}, x^{(j)})$$

2. Exponentiation

$$\kappa(x^{(i)}, x^{(j)}) = \exp(\kappa_1(x^{(i)}, x^{(j)}))$$

3. Addition

$$\kappa(x^{(i)}, x^{(j)}) = \kappa_1(x^{(i)}, x^{(j)}) + \kappa_2(x^{(i)}, x^{(j)})$$

4. Multiplication with function

$$\kappa(x^{(i)}, x^{(j)}) = f(x^{(i)})\kappa_1(x^{(i)}, x^{(j)})f(x^{(j)})$$

5. Multiplication

$$\kappa(x^{(i)}, x^{(j)}) = \kappa_1(x^{(i)}, x^{(j)})\kappa_2(x^{(i)}, x^{(j)})$$

Figure 23: Techniques to construct a new kernel form existing ones. c >0.

References

[1] C. Robert. Machine learning, a probabilistic perspective, 2014.