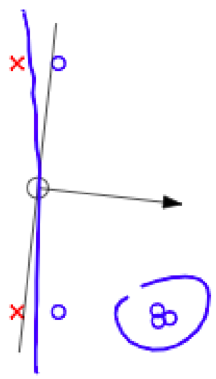
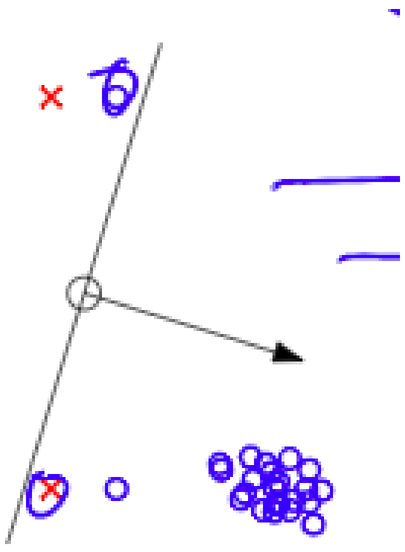
1 Lecture 6 – SVM

Remark 1.1.

$\ell = $ sq loss

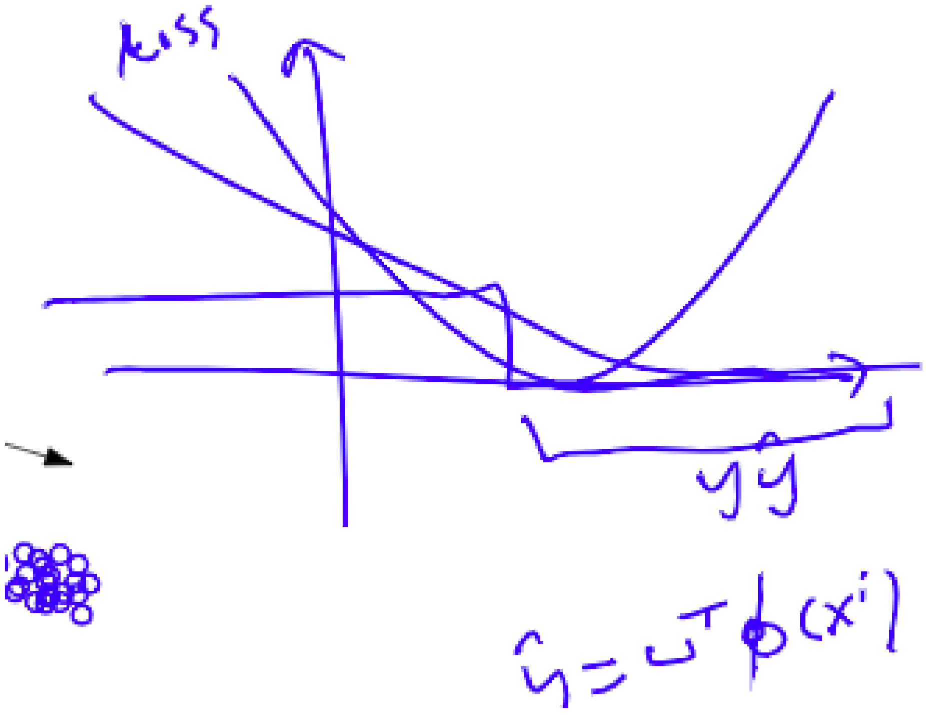


Because square loss seeks to minimize distance between the hyperplane determined by a weight vector and the location of the point, regardless if the point lies on the correct on the correct side of the hyperplane, the tilt above takes place. We see that, whereas the separating hyperplane was formerly straight and erect, the hyperplane's lower edge now tilts towards the circular cluster to minimize the loss contribution of the cluster's members.



Remark 1.2.

If we make a much larger cluster, then the tilting we observed in the previous diagram can become extreme. As above, intuition suggests that we devise a means for the separating hyperplane to deliberately mispredict on a few samples so that it can predict well on the majority of samples.



In the diagram above, $\hat{y} = w^T\phi(x^i)$ and the x-axis covers a range of values for $y\hat{y}$, and the y—axis represents loss. We observe that while logistic regression does eventually work well (meaning that its loss does go down as we become "more" and "more" correct, it still incurs error. For ideally, the loss function should come closs to following the zero one loss given by $1_{\hat{y}y<1}$. By this, we mean to suggest that even logistic regression can be fed a dataset that causes it to make huge blunders as a consequence (this is not verified but intuited by Aahan) of feeding a dataset whose $y\hat{y}$ values are necessarily within a "small" neighborhood of 1. How such a dataset would be constructed is not known.

Recall that the expression below is one term in the loss function for logistic regression, and we observe that as $y\hat{y}$ becomes larger and positive, the loss incurred drops; it should be noted that we still incur loss, however, even if $y\hat{y} = 1$.

$$\log(1 + \exp{(-y^i w^T\phi(x^i))})$$

Remark 1.3. To classify the data, we devise a *linear feasibility problem*, which is

$$\text{Find } w \in \mathbb{R}^2 \text{ such that } y^i w^T x^i > 0 \forall i \in \{1 \ldots n\}.$$

Here we assume that, as before, we are classifying points on the euclidean plane. Tautologically, this problem is solveable if and only if it is feasible.

Definition 1.4. The maximum margin principle chooses w (and thus chooses a hyperplane) such that the distance of a hyperplane to its closest surrounding point is minimal. Formally, the principle seeks to elicit w to solve

$$\max_{w\in\mathbb{R}^n} \min_{i\in[n]} \frac{y^i w^T x^i}{\|w_i\|_2}$$

Here, we restrict our consideration to those w that separate the data. Ie those w such that $y^i w^T x^i > 0 \forall i$.

Remark 1.5. Recall the following: Suppose that we wish to find the distance from x to the plane determined by w. Recall that w also determines the normal vector [1]. Thus, finding the distance from x to the plane determined by w is tantamount to running an instance of the normal vector w through the point x and then calculating the projection of x onto w. This is $\|x \cos\theta\|$ where θ is the angle formed between x and the instance of w running through x. This is also equivalent to $\left\|\frac{x \cdot w}{\|w\|}\frac{x}{\|x\|}\right\|$ by the dot product fomrmula, which reduces to $\frac{x \cdot w}{\|w\|}$. This is what informs the minimum above.

Proposition 1.6.

$$\max_{w \in \mathbb{R}^d} \min_{i \in [n]} \frac{y^i w^T x^i}{\|w_i\|_2} = \max_{w \in \mathbb{R}^d} \frac{1}{\|w\|} \text{ such that } \grave{} \leq y^i w^T x^i \forall i \in \{1 \dots n\}$$

Proof.

$$\begin{aligned}
&\max_{w \in \mathbb{R}^d} \min_{i \in [n]} \frac{y^i w^T x^i}{\|w\|_2} \\
&= \max_{w \in \mathbb{R}^d, r \geq 0} \frac{r}{\|w\|_2} \text{ such that } r \leq y^i w^T x^i \qquad \forall i \\
&= \max_{w \in \mathbb{R}^d, r \geq 0} \frac{r/r}{\frac{\|w\|_2}{r}} \text{ such that } r \leq y^i w^T x^i \qquad \forall i \\
&= \max_{w \in \mathbb{R}^d, r \geq 0} \frac{1}{\|w/r\|_2} \text{ such that } r \leq y^i w^T x^i \qquad \forall i \\
&= \max_{w \in \mathbb{R}^d, r \geq 0} \frac{1}{\|w/r\|_2} \text{ such that } 1 \leq y^i (w/r)^T x^i \qquad \forall i \\
&= \max_{w \in \mathbb{R}^d} \frac{1}{\|w\|_2} \text{ such that } 1 \leq y^i (w)^T x^i \qquad \forall i \\
&= \min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2 \text{ such that } 1 \leq y^i (w)^T x^i \qquad \forall i
\end{aligned}$$

Note that the previous problem is convex

□

Theorem 1.7. The dual for the separable case is

$$D(\alpha) := \inf_{w \in \mathbb{R}^d} L(w, \alpha) = \begin{cases} \sum_{i=1}^n \alpha_i - \frac{1}{2}\left\|\sum_{i=1}^n \alpha_i y^i x^i\right\|^2 & \alpha \geq 0 \\ -\infty \end{cases}$$

[1] I treat the potential problem that w may be augmented by a bias term later.

Proof.

$$= \min_{w \in \mathbb{R}^d} \frac{1}{2}\|w\|^2 \text{ such that } 1 \leq y^i(w)^T x^i \qquad \forall i$$

Compells a Lagrangian, which is:

$$= \frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \alpha_i (1 - y^i w^T x^i) \tag{1}$$

Where $\alpha_i \geq 0$.

Now differentiate this and set it to 0

$$0 = w^T - \sum_{i=1}^{n} \alpha_i y^i \left(x^i\right)^T$$
$$w = \sum_{i=1}^{n} \alpha_i y^i \left(x^i\right)$$

Now substitute this into 1

$$= \frac{1}{2}\left\|\sum_{i=1}^{n} \alpha_i y^i \left(x^i\right)\right\|^2 + \sum_{i=1}^{n} \alpha_i (1 - y^i (\sum_{i=1}^{n} \alpha_i y^i \left(x^i\right))^T x^i)$$
$$= \frac{1}{2}\left\|\sum_{i=1}^{n} \alpha_i y^i \left(x^i\right)\right\|^2 + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} y^i (\sum_{i=1}^{n} \alpha_i y^i \left(x^i\right))^T x^i$$

Notice that the right most expression is really $-\|w\|^2$.

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\left\|\sum_{i=1}^{n} \alpha_i y^i x^i\right\|^2$$

□

Remark 1.8. The foregoing all assumed that we were working in the separable case. If we cannot satisfy the feasibility condition:

$$y^i w^T x^i > 0 \qquad \forall i$$

we introduce a set of slack variables $\{\xi | \xi \geq 0\}$ and reframe the feasibility problem as an optimization problem

$$\min_{\{\xi\}} \sum_{i=1}^{n} \xi_i \qquad \text{such that } y^i w^T x^i > 0 - \xi_i \qquad \forall i$$

The idea is that $\{\xi_i\}$ represent the minimal amount of translations necessary for a feasible problem to be obtained.

Remark 1.9. Note that the previous reformulation is a slackening of the feasibility problem. To slacken the actual optimizaiton problem, we define the Soft Margin SVM, as follows:

Definition 1.10. The Soft Margin SVM (for the non-separable case) solves the following problem:

$$\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \leq y^i w^T x^i \qquad \forall i \in \{1\ldots n\}$$

This is obtained by slackening the former separable optimization problem

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}\|w\|^2 \text{ such that } 1 \leq y^i (w)^T x^i \qquad \forall i$$

Now add $\sum_{i=1}^{n}\xi_i$ to the LHS and subtract ξ_i from the RHS.

$$\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \leq y^i w^T x^i \qquad \forall i \in \{1\ldots n\}$$

Where $C > 0$ is effectively a regularizer

Remark 1.11. To develop intuition for what C does, let us divide C from

$$\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \leq y^i w^T x^i \qquad \forall i \in \{1\ldots n\} \quad (\gamma)$$

to obtain

$$\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} (1/C)\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \leq y^i w^T x^i \qquad \forall i \in \{1\ldots n\}$$

Let $(1/C) = \lambda$.

$$\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \leq y^i w^T x^i \qquad \forall i \in \{1\ldots n\}$$

Now we make the following observations:

- As $\lambda \to \infty$, w becomes more constrained so that more freedom is given to $\{\chi_i\}$. Hence, in the original formulation in (γ), we see that the smaller C is the more flexibility we give to $\{\chi_i\}$. What does it mean to be more flexible? Similarly, as $\lambda \to 0$, w becomes more and more flexible and we care more about minimizing the loss incurred by the $\{\chi_i\}$ so $\{\chi_i\}$ become less flexible.

Unresolved 1.12. What is this tradeoff? Can we motivate it more rigorously?

Theorem 1.13.

$$\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \le y^i w^T x^i \qquad \forall i \in \{1\dots n\}$$

is equivalent to

$$\min_{w\in\mathbb{R}^d} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n} l\text{-hinge}(y^i w^T x^i) \qquad \forall i \in \{1\dots n\}$$

where l-hinge applied to x is $\max\{1-x,0\}$.

Proof.

$$\begin{aligned}&\min_{w\in\mathbb{R}^d,\xi\in\mathbb{R}^n} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \le y^i w^T x^i \qquad \forall i \in \{1\dots n\}\\ =&\min_{w\in\mathbb{R}^d}\min_{\xi\in\mathbb{R}^n} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}\xi_i \quad \text{such that } 1-\xi \le y^i w^T x^i \qquad \forall i \in \{1\dots n\}\end{aligned}$$

Notice that once w is determined, there is an obvious choice, however, for the argument ξ such that the resultant value is minimized.

We obtained this by manipulating the constraint inequality

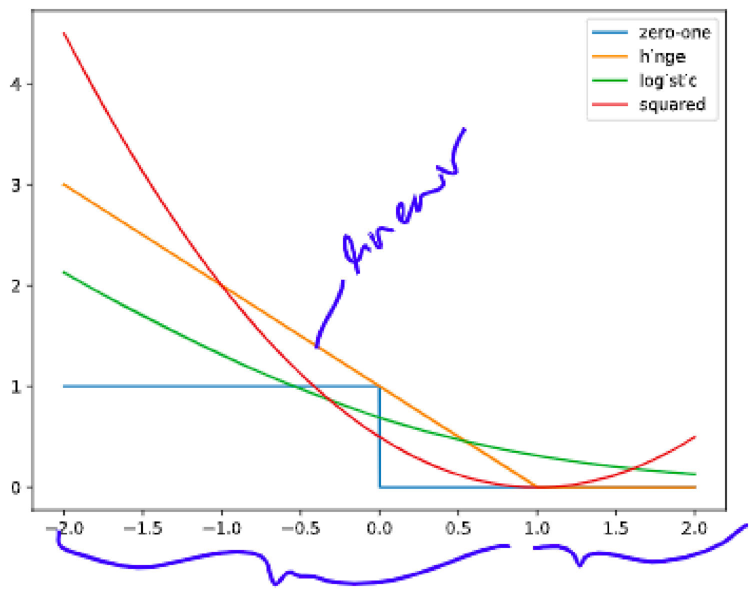
$$= \min_{w\in\mathbb{R}^d} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}(1-y^i w^T x^i)$$

Recall, however, that even if take ξ_i to be $1-y^i w^T x^i$ in order to maximally slacken the inequality $1-y^i w^T x^i \le \xi_i$, we must have $\xi_i \ge 0$, whence we replace $1-y^i w^T x^i$ with

$$= \min_{w\in\mathbb{R}^d} \lambda\frac{1}{2}\|w\|^2 + \sum_{i=1}^{n}(\max\{0, 1-y^i w^T x^i\})$$

□

Remark 1.14.



The graph above plots $y\hat{y} = y^i w^T x^i$ on the $x-$axis and the different loss functions on the y-axis: $(1 - y\hat{y})^2$, $1_{y\hat{y} \leq 1}$, $\sigma(y\hat{y})$ and then $\mathcal{L}(\hat{y}y)$. Hinge loss does not penalize correctness, and it penalizes misclassification linearly; that is $e(y_1\hat{y_1}) = 2e(y_2\hat{y_2}) \iff y_1\hat{y_1} = 2(y_2\hat{y_2})$, where e signifies error.

Remark 1.15.

Dual of slack formulation.

Primal:

$$P(\boldsymbol{w}, \boldsymbol{\xi}) := \begin{cases} \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n \boldsymbol{\xi}_i & 1 - \boldsymbol{\xi}_i \leq y^{(i)}\boldsymbol{w}^\top \boldsymbol{x}^{(i)} \quad \forall i \in \{1, \ldots, n\}, \\ \infty & \text{otherwise.} \end{cases}$$

Lagrangian (with $\boldsymbol{\alpha} \geq 0$)**:**

$$L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^n \boldsymbol{\xi}_i + \sum_{i=1}^n \boldsymbol{\alpha}_i(1 - \boldsymbol{\xi}_i - y^{(i)}\boldsymbol{w}^\top \boldsymbol{x}^{(i)}).$$

Dual (derived as $\inf_{\boldsymbol{w},\boldsymbol{\xi}} L(\boldsymbol{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})$)**:**

$$D(\boldsymbol{\alpha}) = \begin{cases} \sum_{i=1}^n \boldsymbol{\alpha}_i - \frac{1}{2}\left\|\sum_{i=1}^n \boldsymbol{\alpha}_i y^{(i)}\boldsymbol{x}^{(i)}\right\|^2 & 0 \leq \boldsymbol{\alpha}_i \leq C; \\ -\infty & \text{otherwise.} \end{cases}$$

If we try to solve the primal problem for the slack formulation, then when we optimize the Lagrangian by minimizing over w, ξ, we take the gradient with respect to w and then obtain $w = \sum_{i=1}^n \alpha_i y^i x^i$.

Unresolved 1.16. It is not known how they conclude that the optimal value for $\{\xi\}$ is somehow related to $\sup_{\xi \geq 0} \xi(C - \alpha_i)$.

Solution:

We seek to maximize the dual function, which is obtained as the infimum of the Langrangian. If we allowed $\alpha > C$, note that we could arbitrarily then minimize the dual, for then $\xi(C - \alpha_i) \to -\infty$. To foreclose this possibility, we therefore compel that $\alpha \leq C$.

It will turn out that $0 \leq \alpha_i \leq C$, so that the only values contributing to the optimal value of w, which was found to be $\sum_{i=1}^{n} \alpha_i y^i x^i$ are those i such that $\alpha_i > 0$. This is impactful as follows:

Suppose that we optimize the dual program

$$\max_{\alpha \in [0,C]^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left\| \sum_{i=1}^{n} \alpha_i y^i x^i \right\| \tag{β}$$

and obtain a corresponding optimal value for α. Now let us solve this problem again and, this time, discard those (x^i, y^i) such that $\alpha^i = 0$. Now try once more to solve (β) – and we will obtain a vector α_2 such that the α when restricted to its coordinates greater than 0 is precisely α_2. We call the x_i whose $\alpha^i > 0$ the support vectors.

Remark 1.17. Notice that we can rewrite (β) above to be

$$\max_{\alpha \in [0,C]^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y^i y^j (\phi(x_i)^T \phi(x_j))$$

When we define $k(x^i, x^j)$ to be $(\phi(x_i)^T \phi(x_j))$, we call the resultant k a kernel.

Proposition 1.18. If $\phi(x)$ is the feature transform function for a problem, then after finding the optimal $\{\alpha_i\}$ from using an SVM, the prediction function to determine which side of a hyperplane a point lies on is

$$f(x) = \sum_{i=1}^{n} \alpha^i y^i k(x^i, x)$$

Proof. Recall that the optimal value of w, found from the Lagrangian, was found to be

$$w = \sum_{i=1}^{n} \alpha_i y^i \phi(x^i)$$

Recall, further, that we decide the label of a point using $sgn(w^T x)$. Whence, substituting w as above, we now have:

$$\begin{aligned} &\left(\sum_{i=1}^{n} \alpha_i y^i \phi(x^i) \right)^T x \\ &= \left(\sum_{i=1}^{n} \alpha_i y^i k(x^i, x) \right) \end{aligned}$$

□

Proposition 1.19. Suppose that we want to perform gradient descent on hinge loss using only one training example. Then the update rule is

$$w' := (1 - \lambda) w + yx \cdot 1_{[yw^T x < 1]}$$

where we let $\alpha = 1$ be the step size.

Proof. Recall that hinge loss is

$$\mathcal{L}_{\text{hing}}(yw^Tx) + \lambda\|w\|^2/2 = \max\{1 - yw^Tx, 0\} + \lambda\|w\|^2/2$$

Whenever $yw^Tx < 1$, the loss function is therefore

$$1 - yw^Tx + \lambda\|w\|^2/2 \text{ and } \lambda\|w\|^2/2 \text{ otherwise.}$$

Taking the derivative of the first gives us (and transposing so that the gradient is a column vecto)

$$-yx + \lambda w \text{ and the other derivative is } \lambda w$$

Exclusively working with the first derivative, compute

$$w := w - \alpha(-yx + \lambda w) = w(1 - \lambda) + \alpha yx1[yw^Tx < 1]$$

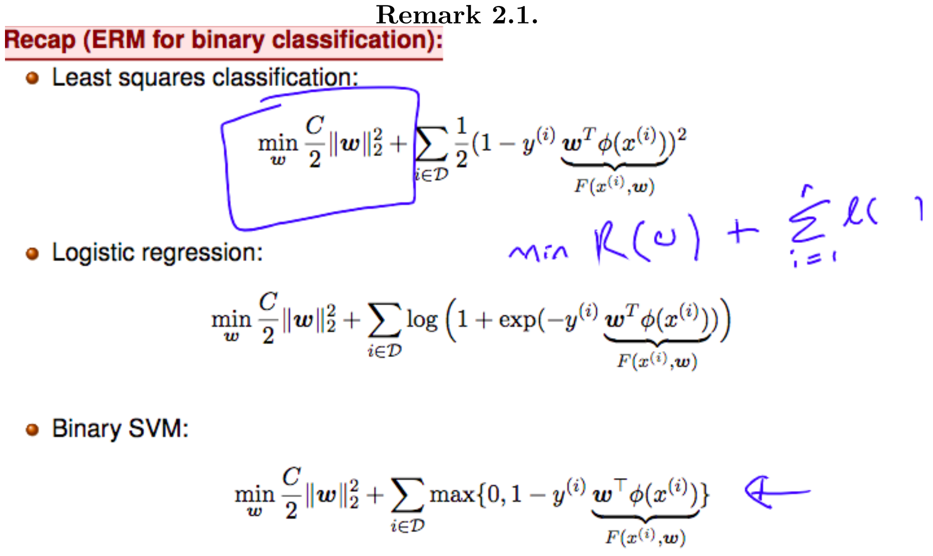
where the final term is added, to account for the fact that we looked at a piecewise component of hinge loss to obtain the loss function which we subsequently differentiated.

Let $\alpha = 1$ to obtain the form in the proposition. ☐

Remark 1.20. This update rule has a geometric interpretation: If $yw^Tx < 1$, meaning that our prediction is off, then update the hyperplane normal vector by tilting it in the direction of yx. If $0 < \lambda < 1$, then we also keep the value of $(1 - \lambda)w$ small.

2 Lecture 7

Remark 2.1.



Recap (ERM for binary classification):

- Least squares classification:

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

$$\min R(\omega) + \sum_{i=1}^{n} \ell(\quad)$$

- Logistic regression:

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

- Binary SVM:

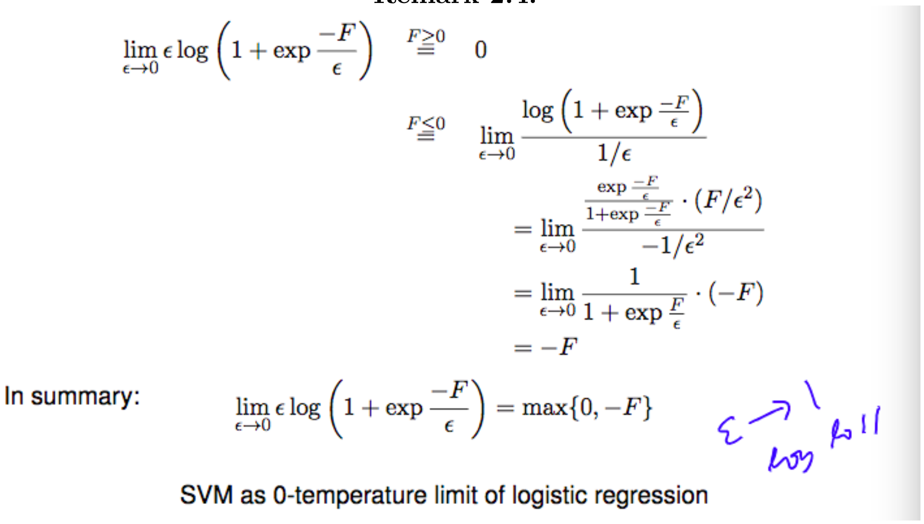
$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\mathcal{D}} \max\{0, 1 - y^{(i)} \underbrace{\boldsymbol{w}^\top\phi(x^{(i)})}_{F(x^{(i)},\boldsymbol{w})}\} \quad \longleftarrow$$

Note that all loss functions studied to date are of the form Regularization + ERM.

Definition 2.2. When a loss function can be interpolated into another loss function, we mean that there exists a function $F(x, \epsilon)$ that can be manipulated to be either l_1 or l_2, where l_1 and l_2 are the loss functions. This often involves some limiting procedure involving ϵ.

Theorem 2.3. We can interpolate log loss into SVM loss.

Remark 2.4.



$$\lim_{\epsilon\to 0} \epsilon \log\left(1+\exp\frac{-F}{\epsilon}\right) \overset{F\geq 0}{=} 0$$

$$\overset{F\leq 0}{=} \lim_{\epsilon\to 0} \frac{\log\left(1+\exp\frac{-F}{\epsilon}\right)}{1/\epsilon}$$

$$= \lim_{\epsilon\to 0} \frac{\frac{\exp\frac{-F}{\epsilon}}{1+\exp\frac{-F}{\epsilon}} \cdot (F/\epsilon^2)}{-1/\epsilon^2}$$

$$= \lim_{\epsilon\to 0} \frac{1}{1+\exp\frac{F}{\epsilon}} \cdot (-F)$$

$$= -F$$

In summary: $$\lim_{\epsilon\to 0} \epsilon \log\left(1+\exp\frac{-F}{\epsilon}\right) = \max\{0, -F\}$$

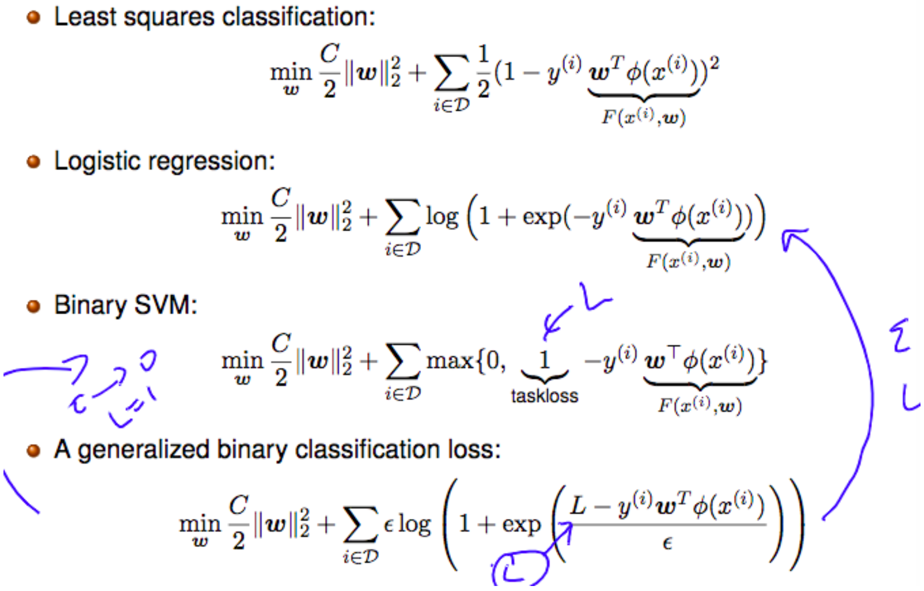
$\epsilon \to 1$

log loss

SVM as 0-temperature limit of logistic regression

Note that we apply L'Hoptial's rule when our limit expression is of the form 0/0.

Remark 2.5.



- Least squares classification:

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

- Logistic regression:

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

- Binary SVM:

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

$\swarrow L$

$\epsilon \to 0$

ϵ

L

- A generalized binary classification loss:

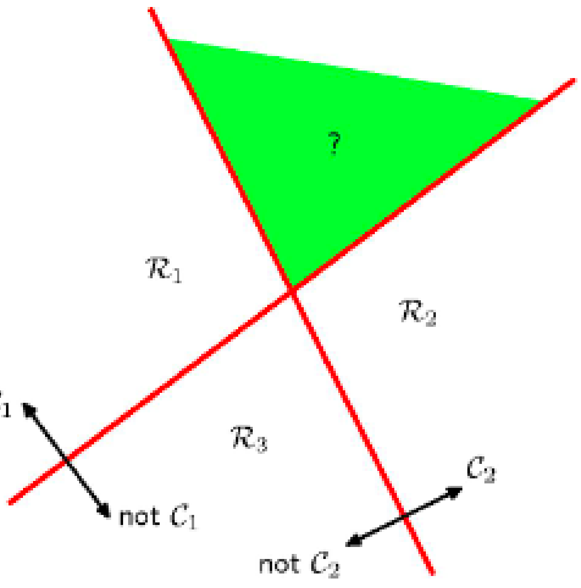
$$\min_{\boldsymbol{w}} \frac{C}{2}\|\boldsymbol{w}\|_2^2 + \sum_{i\in\mathcal{D}} \epsilon \log\left(1 + \exp\left(\frac{L - y^{(i)}\boldsymbol{w}^T\phi(x^{(i)})}{\epsilon}\right)\right)$$

(L)

By choosing different values for L and ϵ in the generalized loss function, we can get different loss functions.

Definition 2.6. To classify between K classes, a 1 versus all classifier constructs $K-1$ classifiers. Each classifier, say C^i, labels if an input belongs to class i or else belongs to some other class.

Remark 2.7.

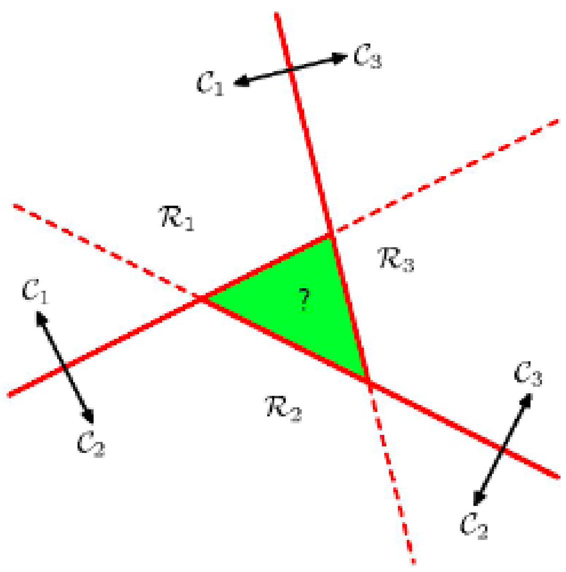
- The 1 versus rest classifier is problematic insofar as:
  - The data used to train a particular classifier, say C^i, is likely imbalanced:
    - If all classes had m samples each, then we would constructing a classifier to correctly categorize m samples as belonging to one class and $m(d-1)$ samples as belonging to another class.
  - We may produce more than one good answer or no good answer:
    - 

      · Here, observe that both C_1 and C_2 are alleged labels for the shaded region.
- It is good insofar as it requires $O(k)$ classifiers.

Definition 2.8. To classify K classes, a 1 versus 1 clasifier creates binary classifiers for each pair of classes. Once all $k(k-1)/2$ classifiers have been constructed, given a sample x, we then give a label for x using each of the classifiers. Whichever class achieves a plurality over x is set to be the class label of x.

Remark 2.9.

- This is problematic insofar as:
  - Each classifiers may have very little data to work with
  - There are $O(K^2)$ classifiers.
  - Two-way preferences may not be transitive:
    -



· Choose a point in the green region. Observe that it would be given the class label C_2 over C_1 and C_3 over C_2; that does not imply, however, that it would be given the label C_3 over C_1; indeed, C_1 is preferred over C_3 here.

- This is beneficial because:
  - The data required to build each classifier is more likely to be balanced than the data required to build each 1 versus all classifier.
  - We can train the classifiers in parallel.

Definition 2.10. The multinomial classifier for classification over K classes constructs $K-1$ weight vectors $w_1 \dots w_{K-1}$. [2] After these vectors are constructed, given a sample x, we predict its label to be the class i such that

$$\mathbb{P}(Y = y^i|x) = \frac{\exp(w_i^T \phi(x))}{\sum_{j=1}^K \exp(w_j^T \phi(x))}$$

is maximal over all other labels. To train the $\{w_i\}$, we do the following: set $w = [w_1 \dots w_K]$ and then minimize the loss function:

$$\arg\max_w \prod_{(x^i,y^i)\in D} \mathbb{P}(Y = y^i|x^i) = \arg\min_w - \sum_{(x^i,y^i)} \log(\mathbb{P}(Y = y^i|x^i)) = \mathcal{L}$$

Performing Gradient Descent, we then minimize over

$$\begin{bmatrix} w_1 \\ \vdots \\ w_K \end{bmatrix} - \alpha_i \begin{bmatrix} \nabla_{w_1}\mathcal{L} \\ \vdots \\ \nabla_{w_K}\mathcal{L} \end{bmatrix}$$

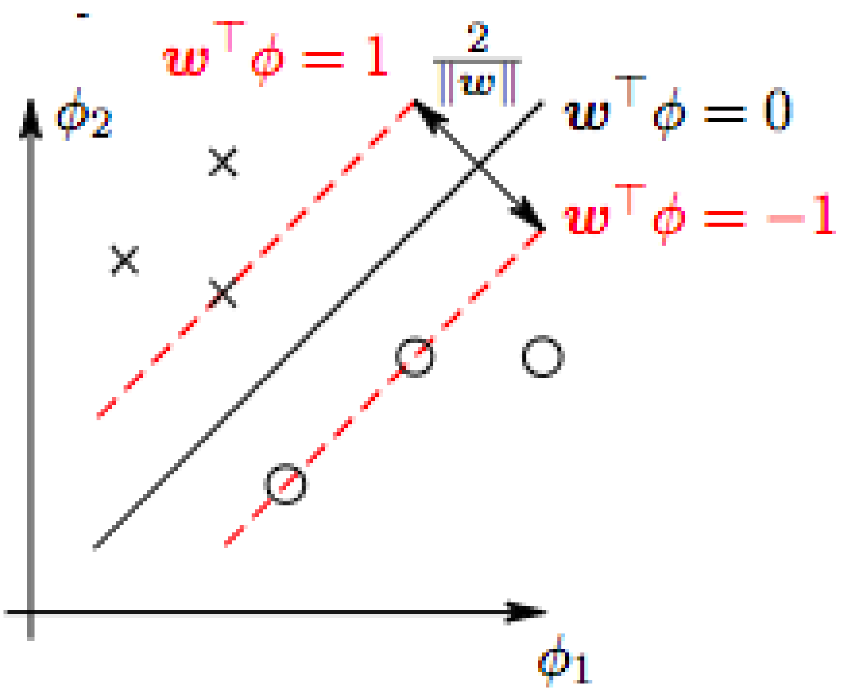
[2]A technicality is that we need the probability function to include a summation over all K weight vectors in its denominator; however, when we train our model, we only need to perform updates for the first $K-1$ weight vectors, and by obtaining probabilities over all the toer

Example 2.11. Suppose that we perform this procedure for two classes. One can show, in that case, that we end up finding that $w_1 = -w_2$.

Example 2.12. We provide some intuition on how maximizing the "margin" is related to the optimization problem:

$$\min_w \frac{1}{2}\|w\|^2 \text{ such that } y^i w^T \phi(x^i) \geq 1 \forall (x^i, y^i) \in \mathcal{D}$$

Suppose that we have data configured as follows:



Suppose that $w^T u_1 = 1$ (this is labeled as $w^T \phi = 1$ in the diagram) and that $w^T u_2 = 2$. Then the following computations can be done and we observe that the distance from u_1 to u_2 is $\frac{2}{\|w\|}$.

$$u_1 + cw = u_2 \qquad w^\top u_1 = 1 \qquad w^\top u_2 = -1$$

$$w^\top (u_1 + cw) = -1$$

$$c = \frac{-2}{\|w\|_2^2}$$

$$\|u_1 - u_2\|_2 = \| -cw\|_2 = \frac{2}{\|w\|_2^2}\|w\| = \frac{2}{\|w\|}$$

Minimizing $\frac{2}{\|w\|}$ is tantamount to minimizing the norm (and hence the square of the norm).

Definition 2.13. Suppose that we have labels $y \in \{0 \dots K-1\}$ and corresponding class vectors w_y. We want to so finely choose our class vectors that we have:

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

Definition 2.14. If we make this definition where $\delta(y^i = Z)$ is really $1_{y^i = Z}$.

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

then observe that

$$w^T\left(\psi(x^i, y^i) - \psi(x^i, \hat{y})\right) \geq 1 - \xi^i \qquad \forall i, \hat{y}$$

is equivalent to

$$w_{y^i}^T \phi(x^i) - w_{\hat{y}}^T \phi(x^i) \geq 1 - \xi^i \qquad \forall i, \hat{y}$$

which informs

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

Using the same technique that we used earlier to define hinge loss, observe that we now need to minimize ξ^i while ensuring that

$$\xi^i \geq 1 - w^T\left(\psi(x^i, y^i) - \psi(x^i, \hat{y})\right) \forall i, \hat{y}$$

If we fix a particular i, so that we want to determine the minimum value of ξ^i, then we declare ξ^i to be at least

$$\max_{\hat{y}}(1 - w^T\left(\psi(x^i, y^i) - \psi(x^i, \hat{y})\right) \forall i, \hat{y})$$

and 0 if this value is negative. Whence we obtain the objective function

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

Notice that $1 - w^T(\psi(x^i, y^i) - \psi(x^i, \hat{y})$ is precisely 1 when $\hat{y} = y^i$ so that the expression $\max_{\hat{y}} 1 - w^T(\psi(x^i, y^i) - \psi(x^i, \hat{y})) \geq 0$.

$$\max_{\hat{y}} 1 - w^T(\psi(x^i, y^i) - \psi(x^i, \hat{y}))$$

Since the outer maximum will unambigously choose the second parameter, the objective becomes

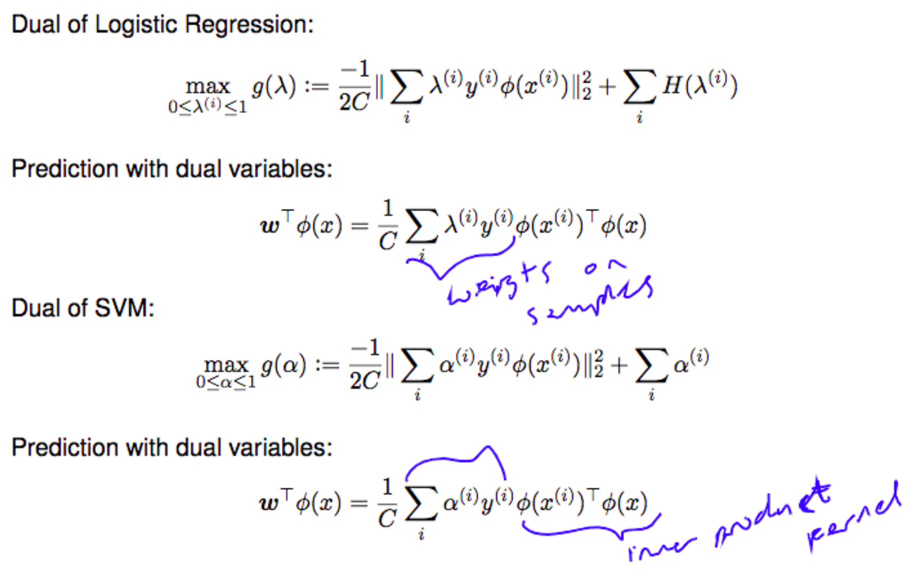
$$\min_w \frac{C}{2}\|w\|^2 + \sum_{i \in D} \max_{\hat{y}}(1 - w^T(\psi(x^i, y^i) - \psi(x^i, \hat{y})))$$

Only the first term after the summation depends on $\hat{y}$, so we have

$$\min_w \frac{C}{2}\|w\|^2 + \sum_{i \in D} \max_{\hat{y}}(1 + w^T(\psi(x^i, \hat{y}))) - w^T(\psi(x^i, y^i))$$

This is the Multiclass SVM objective function.

Remark 2.15. We can also perform interpolation of the SVM to obtain the multiclass logistic regression equation and vice versa.



Dual of Logistic Regression:

$$\max_{0 \leq \lambda^{(i)} \leq 1} g(\lambda) := \frac{-1}{2C}\|\sum_i \lambda^{(i)} y^{(i)} \phi(x^{(i)})\|_2^2 + \sum_i H(\lambda^{(i)})$$

Prediction with dual variables:

$$\boldsymbol{w}^\top \phi(x) = \frac{1}{C}\underbrace{\sum_i \lambda^{(i)} y^{(i)}}_{\text{weights on samples}} \phi(x^{(i)})^\top \phi(x)$$

Dual of SVM:

$$\max_{0 \leq \alpha \leq 1} g(\alpha) := \frac{-1}{2C}\|\sum_i \alpha^{(i)} y^{(i)} \phi(x^{(i)})\|_2^2 + \sum_i \alpha^{(i)}$$

Prediction with dual variables:

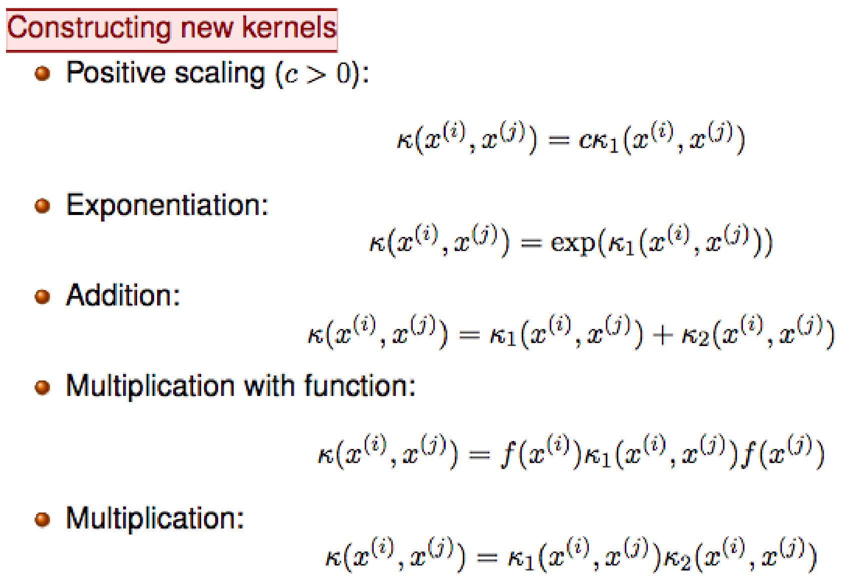
$$\boldsymbol{w}^\top \phi(x) = \frac{1}{C}\overbrace{\sum_i \alpha^{(i)} y^{(i)}}\underbrace{\phi(x^{(i)})^\top \phi(x)}_{\text{inner product kernel}}$$

Notice that both the SVM predictiona and logistic regression (after training) have are similar in form; note especially the inner product $(\phi(x^i))^T \phi(x)$. We can replace the expression $(\phi(x^i))^T \phi(x)$ by $k(x^i, x)$ where k is some function in which the action of ϕ is implicit. This is practically significant: in a training algorithm, instead of constructing $phi(x)$ and then performing the dot product, just replace the call to $\phi(x)^T \phi(x)$ with one to $k(x^T, x)$. There are many kernels we can choose from, however! Can we choose any function? – no! We can choose only those kernels that can be induced as the inner product of two feature vectors obtained via some feature transformation.

There happen to be two tests for this, the theory behind which is presently UNRESOLVED. The first is to form the Gram matrix where $K_{i,j} = k(x^i, x^j)$

and, supposedly, K is positive definite if and only if k is a valid kernel. The second is to decompose the kernel into and from known kernels.

These are valid means of constructing new kernels (UNRESOLVED):



Constructing new kernels

- Positive scaling ($c > 0$):

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

- Exponentiation:

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

- Addition:

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

- Multiplication with function:

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

- Multiplication:

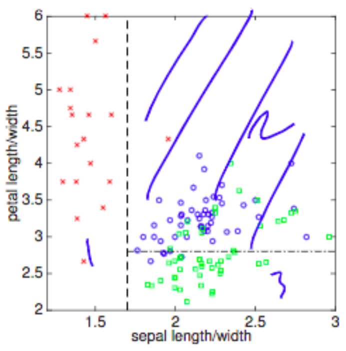
$$\kappa(x^{(i)}, x^{(j)}) = \kappa_1(x^{(i)}, x^{(j)})\kappa_2(x^{(i)}, x^{(j)})$$

3 Lecture 8 – Deep Learning I

4 Lecture 9 – Deep Learning II

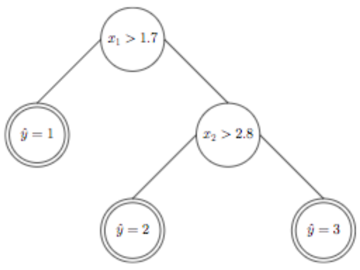
5 Lecture 10 – Trees, Bagging and Boosting

Definition 5.1. A decision tree is a classifier that walks an input down a tree. At each node of the tree, a function is applied to the input: if the output is "positive" in one sense, we go to the right and to the left otherwise.



Classifying irises by sepal and petal measurements

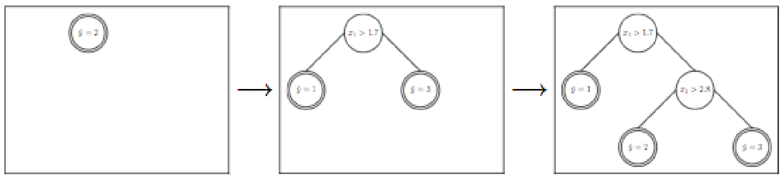
- $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{1, 2, 3\}$
- $x_1 =$ ratio of sepal length to width
- $x_2 =$ ratio of petal length to width



Example 5.2. In the example above, we see that an input initially goes right if its value is greater than 1.7, and it goes left otherwise. Eventually, when we reach a leaf, we have reached the label that we assign to the input. As a consequence of this binary decision nature, decision boundaries often end up like looking like the boundary structure shown to the left.

Basic decision tree learning algorithm.

Leaves are labeled with **majority label** of data reaching them.



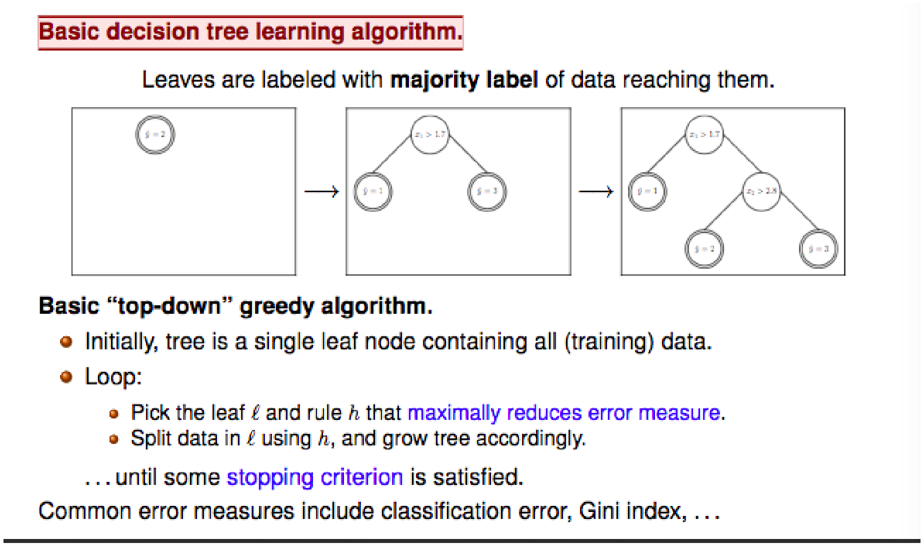
Basic "top-down" greedy algorithm.

- Initially, tree is a single leaf node containing all (training) data.
- Loop:
  - Pick the leaf ℓ and rule h that maximally reduces error measure.
  - Split data in ℓ using h, and grow tree accordingly.

. . . until some stopping criterion is satisfied.

Common error measures include classification error, Gini index, . . .

Remark 5.3. There are two other stopping criterions that are popularly used. The first is to stop when the tree reaches a pre-specified size (it is, for some reason, remarked in the lecture slide that this involves the choice of some tuning hyperparameters). The second is to stop when every leaf is pure (recall that if we begin with a leaf, then the leaf is labeled by all labels). This can result in overfitting, however, in that we also capture noise, as shown below:



Basic decision tree learning algorithm.

Leaves are labeled with **majority label** of data reaching them.

Basic "top-down" greedy algorithm.

- Initially, tree is a single leaf node containing all (training) data.
- Loop:
  - Pick the leaf ℓ and rule h that maximally reduces error measure.
  - Split data in ℓ using h, and grow tree accordingly.

. . . until some stopping criterion is satisfied.

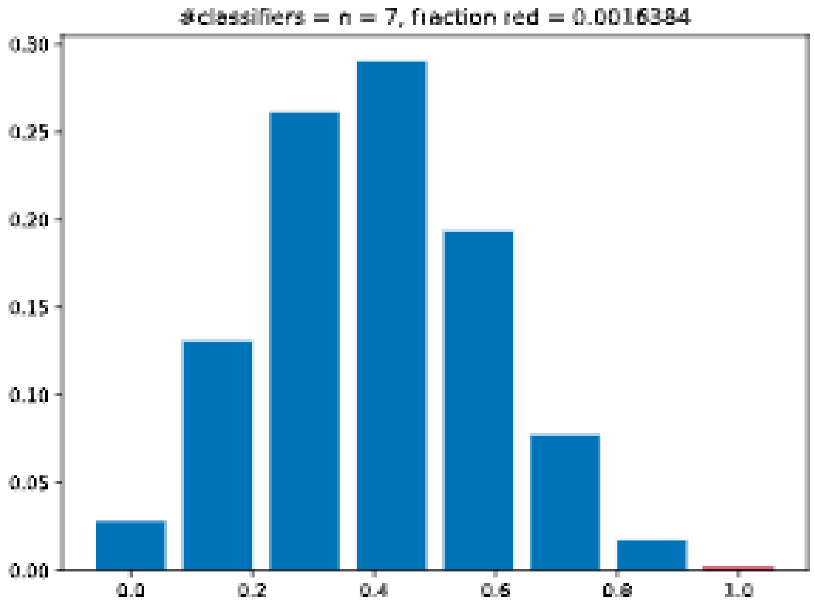
Common error measures include classification error, Gini index, . . .

Remark 5.4. We say that a tree overfits if training error goes to zero as we increase the number of tree nodes but the true error, while initially decreasing, eventually increases. To underfit, by contrast, is to have both poor training error and poor true error. For this reason, there is some regularization done on trees; one idea is to remove tree leaves, a strategy called pruning.

Unresolved 5.5. It is NP hard to determine the smallest decision tree consistent with data. I don't know what this means.

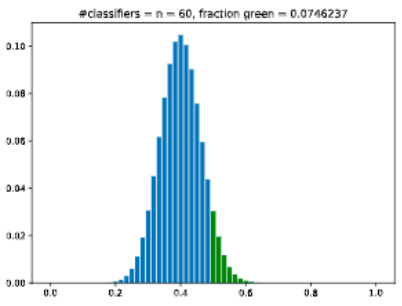
Definition 5.6. An ensemble classifier builds multiple classifiers and then classifies an example using the majority label of the classifiers that it uses.

Remark 5.7. If we assume that each classifier is independent and misclassifies with probability less than 0.5 (in the example that follows, the probability of failure is 0.4), then the probability that all classifications are wrong is very small. This follows a binomial distribution: $B(n, 0.4)$, the results of which are shown below. Note that the x axis is a bit nonsensical – they should really be values of n from 1 to 7.



Remark 5.8. Of course, if we employ an ensemble classification, then we will use the majority label over the ensemble's models, so the relevant question is: what is the probabiliy that more than half the classifiers mispredict? Again, however, we see from the diagram below that this phenomenon is less and less likely the greater n is.

Majority vote.



Green region is error of majority vote!
Suppose $y_i \in \{-1, +1\}$.

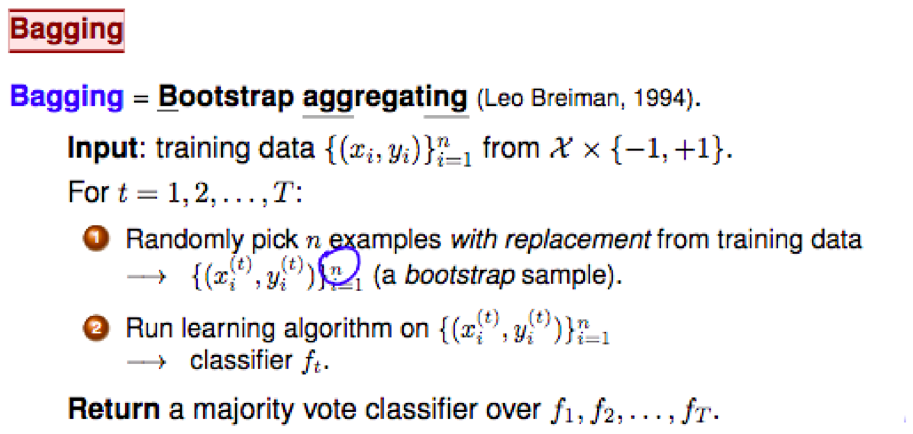
$$\mathsf{MAJ}(y_1, \ldots, y_n) := \begin{cases} +1 & \text{when } \sum_i y_i \geq 0, \\ -1 & \text{when } \sum_i y_i < 0. \end{cases}$$

Error rate of majority classifier (with individual error probability p):

$$\Pr[\mathsf{Binom}(n,p) \geq n/2] = \sum_{i=n/2}^{n} \binom{n}{i} p^i (1-p)^{n-i} \leq \exp\left(-n(1/2-p)^2\right).$$

5.1 Bagging and Random Forests

Remark 5.9. We explored a deterministic rule for constructing a tree. Therefore, the idea of constructing multiple trees as classifiers is nonsensical. There is a remedy called bagging, however. Critical to bagging is the idea that we can construct random datasets over which to make our model using sampling with replacement:



Bagging

Bagging = **Bootstrap aggregating** (Leo Breiman, 1994).

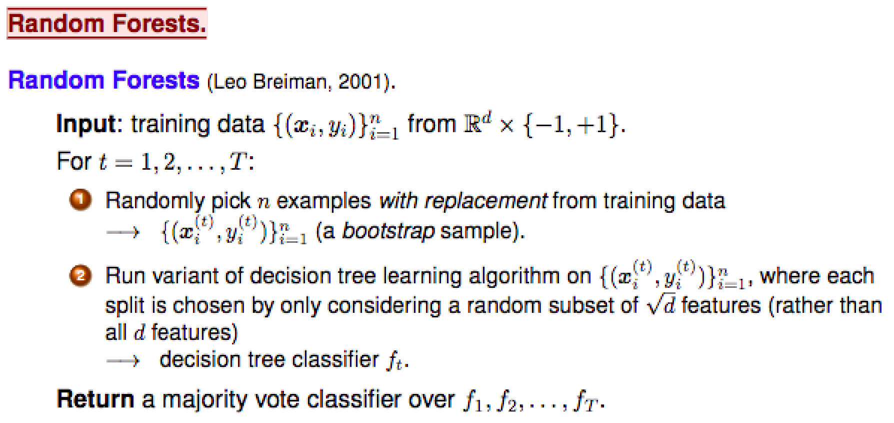
Input: training data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{X} \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick n examples *with replacement* from training data $\longrightarrow$ $\{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ (a *bootstrap* sample).
2. Run learning algorithm on $\{(x_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ $\longrightarrow$ classifier f_t.

Return a majority vote classifier over $f_1, f_2, \ldots, f_T$.

Remark 5.10. A random forest is a particular adapation of bagging to decision trees (note that bagging, above, was not particular to decision trees) in which we construct trees not only over a random sample of data but over a random subset of $\sqrt{d}$ features:



Random Forests.

Random Forests (Leo Breiman, 2001).

Input: training data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ from $\mathbb{R}^d \times \{-1, +1\}$.

For $t = 1, 2, \ldots, T$:

1. Randomly pick n examples *with replacement* from training data $\longrightarrow$ $\{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^n$ (a *bootstrap* sample).
2. Run variant of decision tree learning algorithm on $\{(\boldsymbol{x}_i^{(t)}, y_i^{(t)})\}_{i=1}^n$, where each split is chosen by only considering a random subset of $\sqrt{d}$ features (rather than all d features) $\longrightarrow$ decision tree classifier f_t.

Return a majority vote classifier over $f_1, f_2, \ldots, f_T$.

5.2 Boosting

Definition 5.11. Assume that we have a weak learning oracle (WLO) that, given a reweighted dataset (a dataset $\mathcal{D}$ with a probability assigned to each point), gives us a weak classifier with error $1/2 - \gamma$ for some $\gamma > 0$. Then do the following:

$\Sigma\, w_i$ [illegible]

```
input  Training data $\{(x_i, y_i)\}_{i=1}^n$ from $\mathcal{X} \times \{-1, +1\}$.
1: initialize $D_1(i) := 1/n$ for each $i = 1, 2, \dots, n$ (a probability distribution).
2: for $t = 1, 2, \dots, T$ do
3:     Give $D_t$-weighted examples to WLO; get back classifier $f_t : \mathcal{X} \to \{-1, +1\}$.
4:     Update weights:
           $z_t := \sum_{i=1}^n D_t(i) \cdot y_i f_t(x_i) \in [-1, +1]$
           $\alpha_t := \frac{1}{2} \ln \frac{1+z_t}{1-z_t} \in \mathbb{R}$   (weight of $f_t$)
           $D_{t+1}(i) := D_t(i) \exp(-\alpha_t \cdot y_i f_t(x_i)) / Z_t$   for each $i = 1, 2, \dots, n$,
       where $Z_t > 0$ is normalizer that makes $D_{t+1}$ a probability distribution.
5: end for
6: return  Final classifier $\hat{f}(x) := \mathrm{sign}\left(\sum_{t=1}^T \alpha_t \cdot f_t(x)\right)$.
           (Let $\mathrm{sign}(z) := 1$ if $z > 0$ and $\mathrm{sign}(z) := -1$ if $z \le 0$.)
```

weight

Remark 5.12. We give some intuition for why AdaBoost is favorable:
We can motivate this as if it were a problem:

Problem 1. Suppose that $(X, Y) \sim D_t$ (this means that (X, Y) obey the distribution D_t). Suppose that we have a weak classifier f_t such that

$$\mathbb{P}(f_t(X) = Y) = \frac{1}{2} + \gamma_t$$

Then show that

$$\sum_{i=1}^n D_t(i)\,(y_i f_t(x_i)) \ge 2\gamma_t$$

Hint: the claim

$$\mathbb{P}(f_t(X) = Y) = \frac{1}{2} + \gamma_t$$

is "approximately" equivalent to the claim that

$$\sum_{i=1}^n \delta(y^i = f(x_i)) D_t(i) \ge \frac{1}{2} + \gamma_t$$

Proof. Observe that

$$\begin{aligned}
&\sum_{i=1}^n \delta(y^i = f(x_i)) D_t(i) \ge \frac{1}{2} + \gamma_t \\
\implies &\sum_{i=1}^n \delta(y^i \ne f(x_i)) D_t(i) \le \frac{1}{2} - \gamma_t \\
\implies &\sum_{i \text{ such that } y_i f(x_i) = -1} (-1) D_t(i) \ge \gamma_t - \frac{1}{2}
\end{aligned}$$

Similarly:

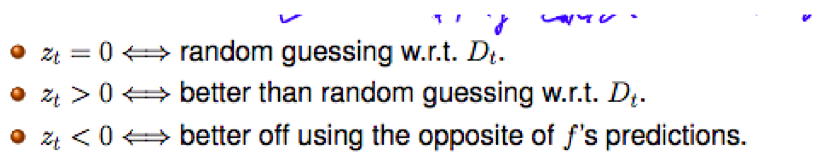
$$\sum_{i \text{ such that } y_i f(x_i)=1} (1)D_t(i) \geq \gamma_t + \frac{1}{2}$$

Combining the foregoing too, we get

$$\sum_{i \text{ such that } y_i f(x_i)=1} (1)D_t(i) + \sum_{i \text{ such that } y_i f(x_i)=-1} (-1)D_t(i) = \sum_{i=1}^{n} D_t(i)(y_i f(x_i)) \geq \gamma_t + \frac{1}{2} + \gamma_t - \frac{1}{2} = 2\gamma_t$$

$\square$

<++>



- $z_t = 0 \Longleftrightarrow$ random guessing w.r.t. D_t.
- $z_t > 0 \Longleftrightarrow$ better than random guessing w.r.t. D_t.
- $z_t < 0 \Longleftrightarrow$ better off using the opposite of f's predictions.

Recall that

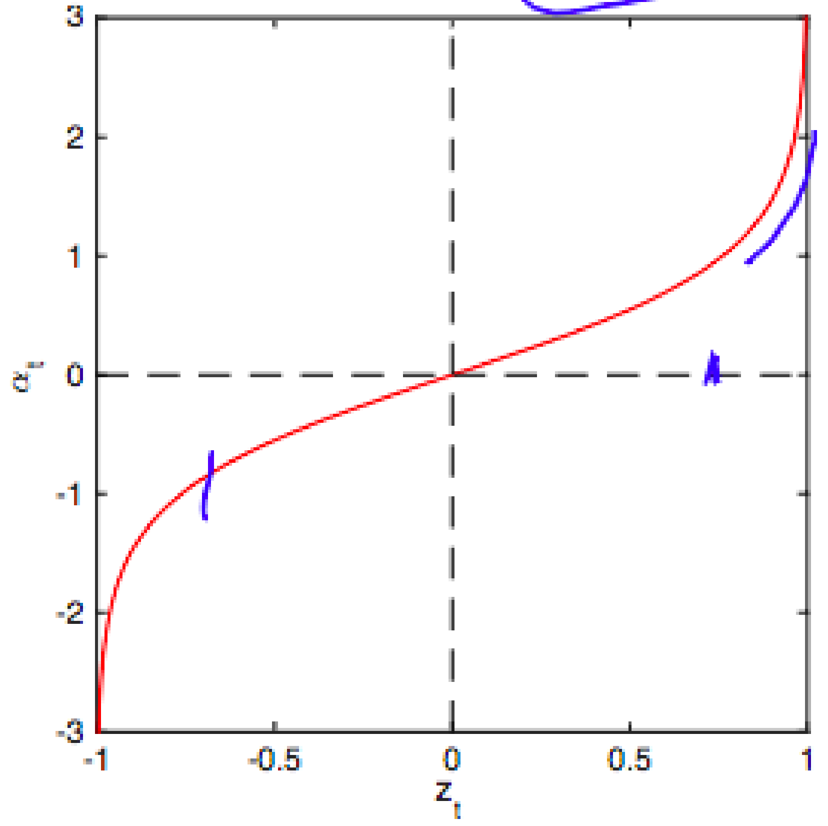
$$z_t = \sum_{i=1}^{n} D_t(i)(f_t(x_i)y_i)$$

We are weighing correctness $(+1)$ and incorrectness (-1) by the probabilities that either correctness or incorrectness take place with.

So that a positive z_t signals prediction better than random guessing in the world defined by D_t, $z_t = 0$ signals random guessing and negative z_t signals prediction worse than random guessing.

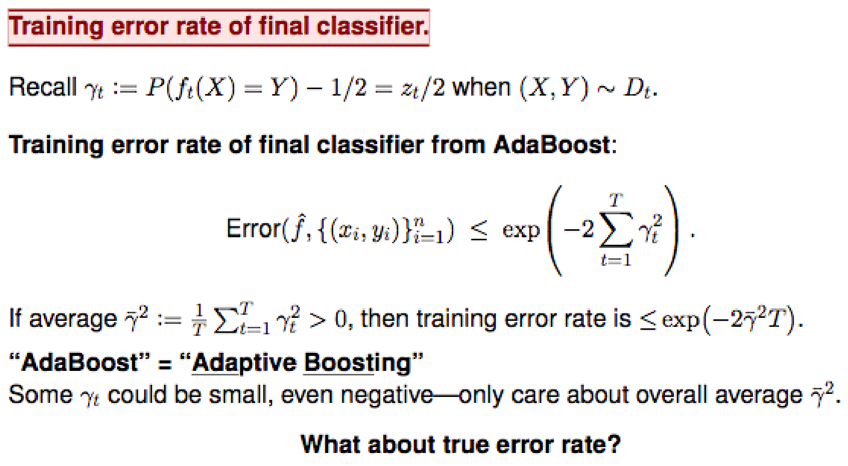
Once we calculate z_t, if z_t is positive (meaning, again, that we are doing a better job than normal), we effectively set α_t to be positive and negative otherwise (ie if z_t is negative). α_t will then divert training attention accordingly in the next round of AdaBoost.

Classifier weights $\alpha_t = \frac{1}{2}\ln\frac{1+z_t}{1-z_t}$



Now recall that $D_{t+1}(i) = D_t(i)\exp\left(-\alpha_i \cdot y_i f_t(x_i)\right)/Z_t$, sot hat we give a higher weight to data points that were misclassified than to those that were well classified.

Remark 5.13.



Training error rate of final classifier.

Recall $\gamma_t := P(f_t(X) = Y) - 1/2 = z_t/2$ when $(X,Y) \sim D_t$.

Training error rate of final classifier from AdaBoost:

$$\text{Error}(\hat{f}, \{(x_i, y_i)\}_{i=1}^n) \le \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right).$$

If average $\bar{\gamma}^2 := \frac{1}{T}\sum_{t=1}^{T}\gamma_t^2 > 0$, then training error rate is $\le \exp(-2\bar{\gamma}^2 T)$.

"AdaBoost" = "Adaptive Boosting"

Some γ_t could be small, even negative—only care about overall average $\bar{\gamma}^2$.

What about true error rate?

The first line is really a manipulation of the previously proven fact that if $z_t = \sum_{i=1}^n D_t(i)[y_i f_t(x_i)]$ and if

$$\frac{1}{2} + \gamma_t = \mathbb{P}(f_t(X) = Y) \approx \sum_{i=1}^{n} D_t(i)\delta[y_i = f(x_i)]$$

then we can say $z_t = 2\gamma_t$. Clearly then, it is true, as the line asserts, that

$$\mathbb{P}(f_t(X) = Y) = \gamma_t + \frac{1}{2} = \frac{2\gamma_t + 1}{2} = \frac{z_t + 1}{2}$$

The other lines are black boxes that we must take for granted. Importantly, the error of the classifier that adaboost would give us at iteration T is bounded exponentially by the averages of γ_t^2 for $t \in \{1 \dots T\}$. This measn that even if some γ_t are negative, that is okay: we care that the average be positive.

When we finish with Ada Boost, we get this final predictor above. UNRESOLVED: why do larger SVM margins imply better resistance to overfitting.