SVMs, Kernelized SVMs

Sriram Sankararaman

The instructor gratefully acknowledges Fei Sha, Ameet Talwalkar, Kai-Wei Chang, Eric Eaton, and Jessica Wu whose slides are heavily used, and the many others who made their course material freely available online.

Outline

1. Review of last lecture
2. SVM – Hinge loss
3. Kernelized SVMs
4. Evaluating ML algorithms

Support Vector Machine

- A linear classifier (hyperplane) that maximizes the margin.
- An alternative view of SVMs.

Outline

A general view of supervised learning

Definition Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \textsc{sign}(a(\boldsymbol{x}))$ with $a(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$

For classification: $0/1$ loss

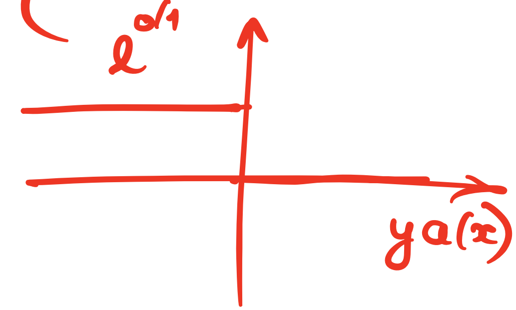
$$\ell^{0/1}(y, a(\boldsymbol{x})) = \begin{cases} 0 & \text{if } ya(\boldsymbol{x}) \geq 0 \\ 1 & \text{otherwise} \end{cases}$$

Minimize weighted sum of empirical risk and regularizer

$$
\begin{aligned}
&\arg\min_{\boldsymbol{w},b} R^{\text{EMP}}[h_{\boldsymbol{w},b}(x)] + \lambda R(\boldsymbol{w}, b) \\
&= \arg\min_{\boldsymbol{w},b} \frac{1}{N} \sum_n \ell(y_n, a(\boldsymbol{x}_n)) + \lambda R(\boldsymbol{w}, b)
\end{aligned}
\tag{1}
$$

Minimize weighted sum of empirical risk and regularizer

$$\ell^{0/1}(y, a(x)) = \begin{cases} 0 & ya(x) \geq 0 \\ 1 & \text{else} \end{cases}$$



$$
\begin{aligned}
&\arg\min_{\boldsymbol{w},b} R^{\text{EMP}}[h_{\boldsymbol{w},b}(x)] + \lambda R(\boldsymbol{w}, b) \\
&= \arg\min_{\boldsymbol{w},b} \frac{1}{N} \sum_n \ell(y_n, a(\boldsymbol{x}_n)) + \lambda R(\boldsymbol{w}, b) \qquad (1)
\end{aligned}
$$

- Problem with minimizing the $0/1$ loss ?

Hinge loss

Definition Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \text{SIGN}(a(\boldsymbol{x}))$ with $a(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$,

$$\ell^{\text{HINGE}}(y, a(\boldsymbol{x})) = \begin{cases} 0 & \text{if } ya(\boldsymbol{x}) \geq 1 \\ 1 - ya(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

Intuition

$ya(x) < 0$

$ya(x) \geq 0$

$ya(x) < 1$

Hinge loss

Definition Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \text{SIGN}(a(\boldsymbol{x}))$ with $a(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$,

$$\ell^{\text{HINGE}}(y, a(\boldsymbol{x})) = \begin{cases} 0 & \text{if } ya(\boldsymbol{x}) \geq 1 \\ \underbrace{1 - ya(\boldsymbol{x})} & \text{otherwise} \end{cases}$$

$ya(x) < 0$

Intuition

- No penalty if raw output, $a(\boldsymbol{x})$, has same sign and is far enough from decision boundary (i.e., if 'margin' is large enough)
- Otherwise pay a growing penalty, between 0 and 1 if signs match, and greater than one otherwise

Hinge loss

Definition Assume $y \in \{-1, 1\}$ and the decision rule is $h(\boldsymbol{x}) = \text{SIGN}(a(\boldsymbol{x}))$ with $a(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}) + b$,

$$\ell^{\text{HINGE}}(y, a(\boldsymbol{x})) = \begin{cases} 0 & \text{if } ya(\boldsymbol{x}) \geq 1 \\ 1 - ya(\boldsymbol{x}) & \text{otherwise} \end{cases}$$

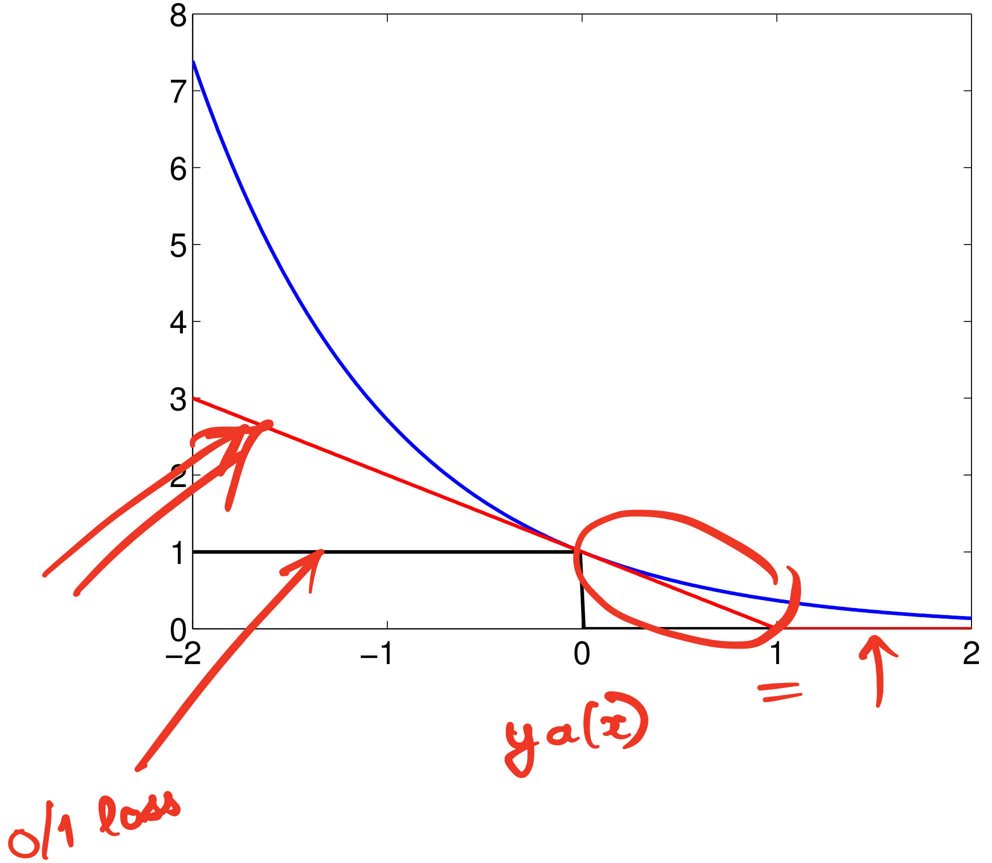
Intuition

- No penalty if raw output, $a(\boldsymbol{x})$, has same sign and is far enough from decision boundary (i.e., if 'margin' is large enough)
- Otherwise pay a growing penalty, between 0 and 1 if signs match, and greater than one otherwise

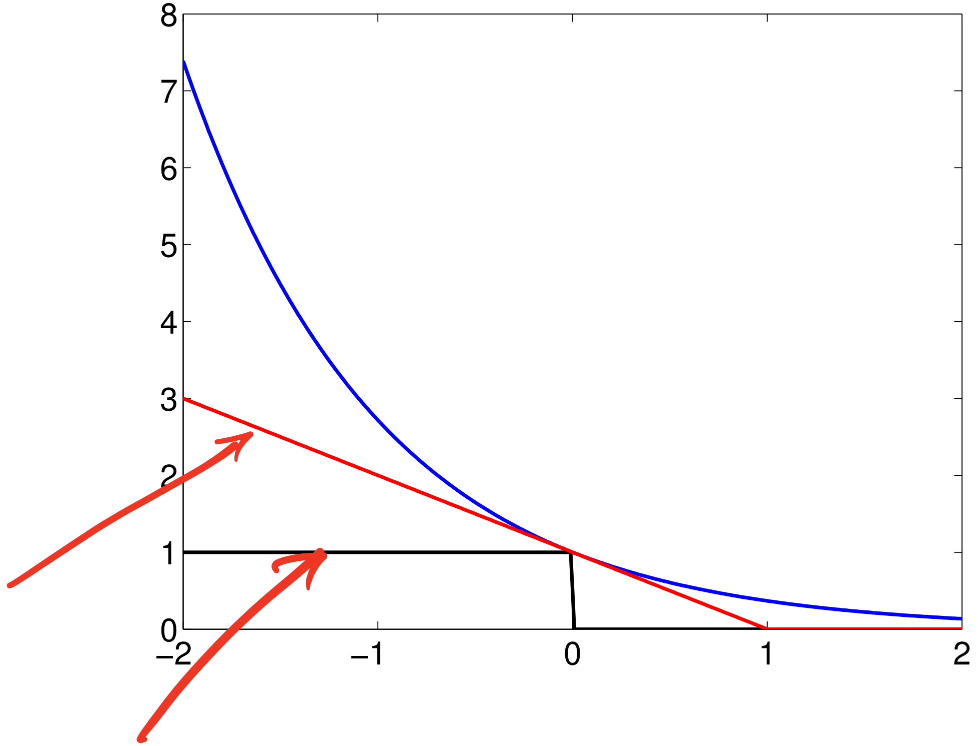
Convenient shorthand

$$\ell^{\text{HINGE}}(y, a(\boldsymbol{x})) = \max(0, 1 - ya(\boldsymbol{x})) = (1 - ya(\boldsymbol{x}))_+$$

Visualization and Properties



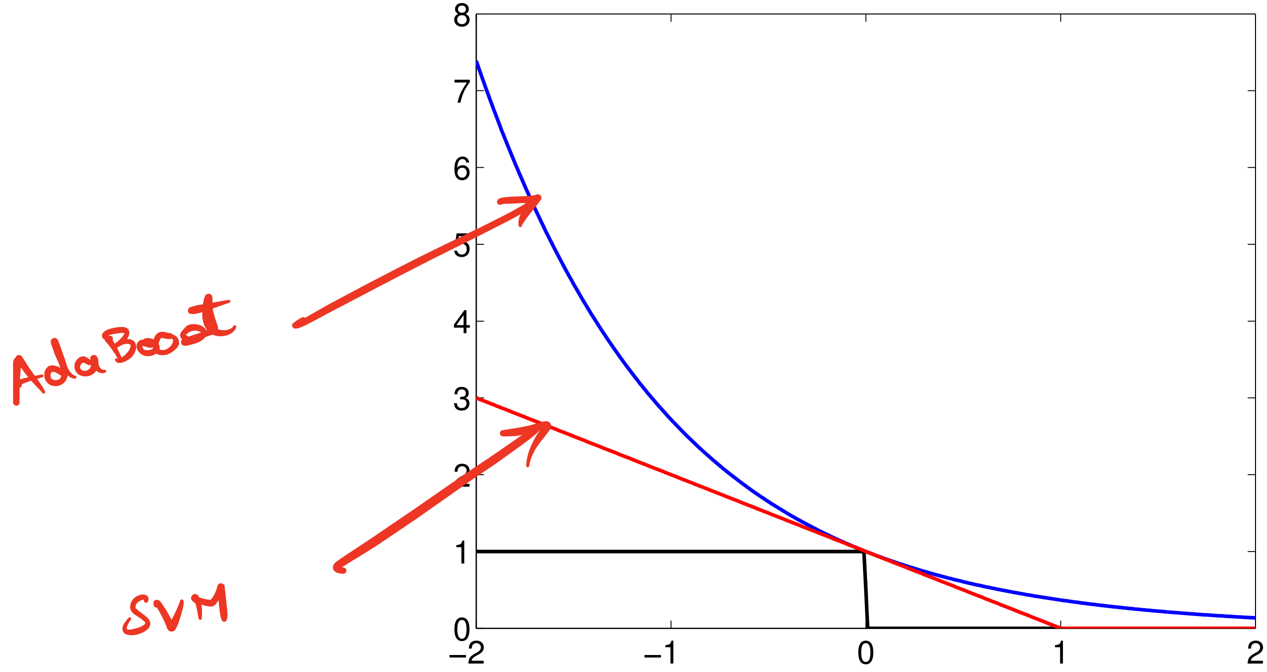
Visualization and Properties



- Upper-bound for $0/1$ loss function (black line)
- We use hinge loss as a *surrogate* to $0/1$ loss – Why?

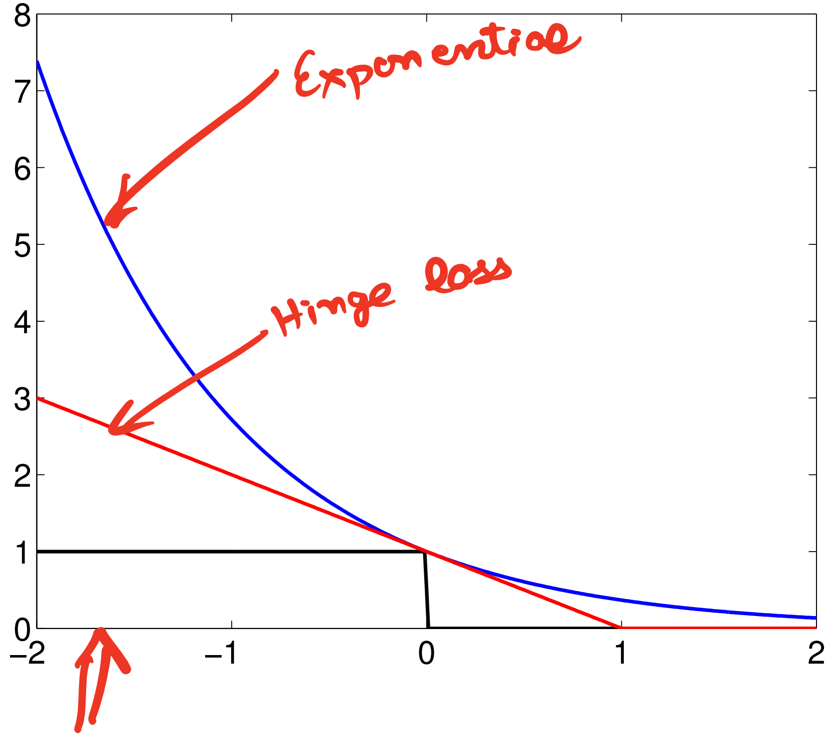
Visualization and Properties

Logistic loss → Logistic regression



- Upper-bound for $0/1$ loss function (black line)
- We use hinge loss as a *surrogate* to $0/1$ loss – Why?
- Hinge loss is convex, and thus easier to work with.

Visualization and Properties



- Other surrogate losses can be used, e.g., exponential loss for Adaboost (in blue), logistic loss (not shown) for logistic regression
- Hinge loss less sensitive to outliers than exponential (or logistic) loss

Primal formulation of support vector machines (SVM)

Minimizing the total hinge loss on all the training data with l_2 regularization

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Analogous to l_2 regularized least squares, as we balance between two terms (the loss and the regularizer).

Hinge loss

(x_n, y_n)

$\max(0, 1 - y\,a(x))$

$a(x) = [w^T \phi(x) + b]$

Primal formulation of support vector machines (SVM)

Minimizing the total hinge loss on all the training data with l_2 regularization

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Analogous to l_2 regularized least squares, as we balance between two terms (the loss and the regularizer).

Previously, we used geometric arguments to derive:

Margin

Slack variables

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n \quad \text{and} \quad \xi_n \geq 0, \quad n \in \{1, \ldots, N\}$$

Do these yield the same solution?

Recovering our previous SVM formulation

Minimizing the total hinge loss on all the training data

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Define $C = 1/\lambda$:

$$\min_{\boldsymbol{w},b} C\sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

Recovering our previous SVM formulation

Minimizing the total hinge loss on all the training data

$$\min_{\boldsymbol{w},b} \sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$$

Define $C = 1/\lambda$:

$$\min_{\boldsymbol{w},b} C\sum_n \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

Define $\xi_n = \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b])$

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) = \xi_n, \quad n \in \{1, \ldots, N\}$$

Recovering our previous SVM formulation

$(w^*, b^*), \{\xi_n^*\}$

Minimizing hinge loss

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2$$

$$\text{s.t.} \quad \max(0, 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b]) = \xi_n, \quad n \in \{1, \ldots, N\}$$

is equivalent to

$\xi_n^* = \max(0, 1 - y_n(w^{*T}\phi(x_n) + b^*))$

SVM

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \left[C\sum_n \xi_n + \frac{1}{2}\|\boldsymbol{w}\|_2^2 \right]$$

$$\text{s.t.} \quad 1 - y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \leq \xi_n, \quad n \in \{1, \ldots, N\}$$

$$0 \leq \xi_n, \quad n \in \{1, \ldots, N\}$$

$\Rightarrow \xi_n \geq \max(0, 1 - y_n(w^T\phi(x_n) + b))$

$\Rightarrow \xi_n^* \geq \max(0, 1 - y_n(w^{*T}\phi(x_n) + b^*))$

At optimal solution constraints are active so we have equality! Why?

$$\xi_n^* \geq \max\left(0,\ 1 - y_n\left(w^{*T}\phi(x_n) + b^*\right)\right)$$

$$\tilde{\xi}_n$$

$$C \sum_n \xi_n + \frac{1}{2}\|w\|_2^2$$

At optimal solution constraints are active so we have equality! Why?

- If $\xi_n^* > \max(0, 1 - y_n f(\boldsymbol{x}_n))$, we could choose $\bar{\xi}_n < \xi_n^*$ and still satisfy the constraint while reducing our objective function!
- Since $c \geq \max(a, b) \iff c \geq a,\, c \geq b$, we recover previous formulation

SVM: hyperplane with largest margin

= minimizes Hinge loss

+ ℓ_2- regularization

Outline

Goal: Kernelize SVMs

$\phi(x_n)$

- Rewrite the SVM optimization problem so that it no longer depends on $\phi(\boldsymbol{x}_n)$ but instead depends only on inner products $\phi(\boldsymbol{x}_n)^{\mathrm{T}}\phi(\boldsymbol{x}_m)$.
- If we can do this, we can then replace $\phi(\boldsymbol{x}_n)^{\mathrm{T}}\phi(\boldsymbol{x}_m)$ with a kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$.

Primal formulation of support vector machines (SVM)

$$\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_n \xi_n$$

$$\text{s.t.} \quad y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1 - \xi_n \quad \text{and} \quad \xi_n \geq 0, \quad n \in \{1, \ldots, N\}$$

This can be converted to its *dual form*.

Dual formulation of SVM

Let $\boldsymbol{w}_*$ be the minimizer of the SVM problem for training data: $\{(\boldsymbol{x}_n, y_n)\}$.

Then $\boldsymbol{w}_* = \sum_n \alpha_n y_n \phi(\boldsymbol{x}_n)$ for a new set of dual variables: $\alpha_n \geq 0$.

Primal

Dual

Dual formulation of SVM

$y_m \in \{+1, -1\}$

Dual is also a convex program

$\alpha = (\alpha_1, \ldots, \alpha_N)$

$$\max_{\boldsymbol{\alpha}} \left(\sum_n \alpha_n - \frac{1}{2} \sum_{m,n} y_m y_n \alpha_m \alpha_n \boldsymbol{\phi}(\boldsymbol{x}_m)^{\mathrm{T}} \boldsymbol{\phi}(\boldsymbol{x}_n) \right)$$

$$\text{s.t.} \quad 0 \leq \alpha_n \leq C, \quad n \in 1, \ldots, N$$

$$\sum_n \alpha_n y_n = 0$$

There are N dual variable α_n, one for each constraint in the primal formulation

Kernel SVM

We replace the inner products $\phi(\boldsymbol{x}_m)^{\mathrm{T}}\phi(\boldsymbol{x}_n)$ with a kernel function

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_n \alpha_n - \frac{1}{2}\sum_{m,n} y_m y_n \alpha_m \alpha_n k(\boldsymbol{x}_m, \boldsymbol{x}_n) \\ \text{s.t.} \quad & 0 \leq \alpha_n \leq C, \quad n \in 1, \ldots, N \\ & \sum_n \alpha_n y_n = 0 \end{aligned}$$

α_*

We can define a kernel function to work with nonlinear features and learn a nonlinear decision surface

Recovering solution to the primal formulation

$w^T \phi(x_n) + b$ $\quad$ $k(x, x_n)$

Weights w (primal variables) relation to α (dual variables)

$$w = \sum_n y_n \alpha_n \phi(x_n) \leftarrow$$ Linear combination of the input features

Prediction on a test point x

$$h(x) = \text{SIGN}(w^T \phi(x) + b) = \text{SIGN}(\sum_n y_n \alpha_n k(x_n, x) + b)$$

At test time it suffices to know the kernel function!

$$w = \sum_n y_n \alpha_n \phi(x_n)$$

$$w^T \phi(x) = \left(\sum_n y_n \alpha_n \phi(x_n)\right)^T \phi(x) = \sum_n y_n \alpha_n \phi(x_n)^T \phi(x)$$

Recovering solution to the primal formulation

We already identified the primal variable $\boldsymbol{w}$ as

$$\boldsymbol{w} = \sum_n \alpha_n y_n \boldsymbol{\phi}(\boldsymbol{x}_n)$$

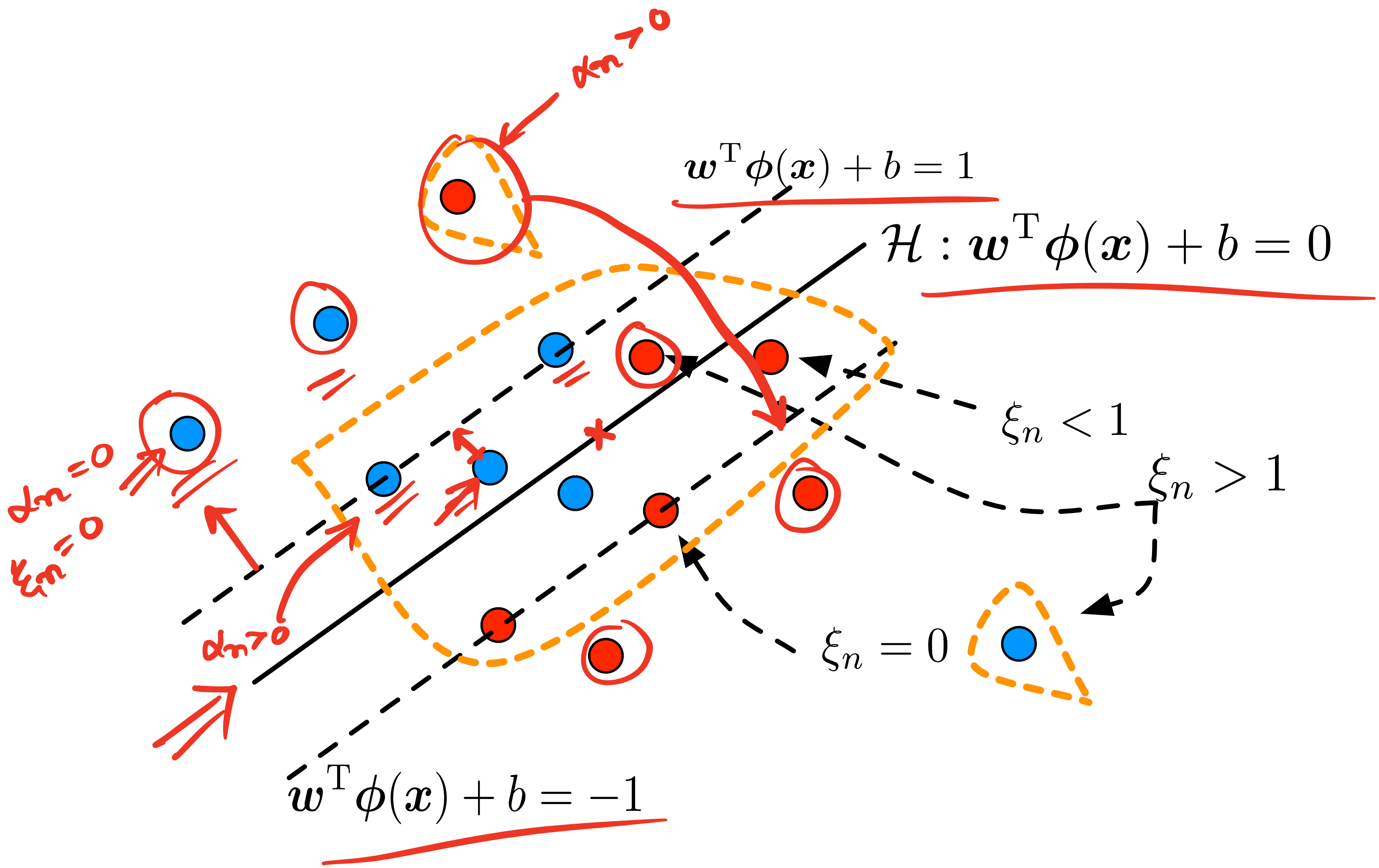
$0 \leq \alpha_n \leq C$

- From the dual, we know that $0 \leq \alpha_n \leq C$.
- If $\alpha_n = 0$, then the example n does not affect the hyperplane/decision boundary computed by SVM. $y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \geq 1$.
- Only those examples with $\alpha_n > 0$ affect the hyerplane/decision boundary. These examples are termed support vectors.
- When will $\alpha_n > 0$?

$\xi_n \geq 0$

  - $\alpha_n = C \Rightarrow y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] \leq 1$.
  - $0 < \alpha_n < C \Rightarrow y_n[\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_n) + b] = 1$.

Visualization of how training data points are categorized



Support vectors are highlighted by the dotted orange lines

Outline

Binary classification

Goal is to achieve high accuracy

Accuracy: defined to be the mean of the $0/1$ loss $l(y, \hat{y})$.

$$\ell(y, h(\boldsymbol{x})) = \mathbf{1}\{y \neq h(\boldsymbol{x})\}$$

$$R^{\text{EMP}}[h(\boldsymbol{x})] = \frac{1}{N} \sum_{n} \ell(h(\boldsymbol{x}_n), y_n)$$

Assumes $y \in \{0, 1\}$ and h outputs values in $\{0, 1\}$.

Binary classification

Goal is to achieve high accuracy

Accuracy: defined to be the mean of the $0/1$ loss $l(y, \hat{y})$.

$$\ell(y, h(\boldsymbol{x})) = \mathbf{1}\{y \neq h(\boldsymbol{x})\}$$
$$R^{\mathrm{EMP}}[h(\boldsymbol{x})] = \frac{1}{N}\sum_n \ell(h(\boldsymbol{x}_n), y_n)$$

Assumes $y \in \{0, 1\}$ and h outputs values in $\{0, 1\}$.

May not be what we want

Is this email spam?

Does the patient have cancer?

Confusion matrix

Given a dataset of P positive instances and N negative instances,

		Predicted class 1	Predicted class 0	
Actual class	1	TP	FN	P
	0	FP	TN	N

P = Positive, N = Negative
TN = True negative, FP = False positive
TP = True positive, FN = False negative

Spam classification

- Positive : spam, Negative: non-spam
- TP: Spam email classified as spam
- FP: Non-spam email classified as spam

Confusion matrix

Given a dataset of P positive instances and N negative instances,

		Predicted class 1	Predicted class 0	
Actual class	1	TP	FN	P
	0	FP	TN	N

P = Positive, N = Negative
TN = True negative, FP = False positive
TP = True positive, FN = False negative

- Accuracy $= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{P}+\mathrm{N}}$
- Error = 1 - Accuracy

Confusion matrix

Given a dataset of P positive instances and N negative instances,

		Predicted class		
		1	0	
Actual	1	TP	FN	P
class	0	FP	TN	N

P = Positive, N = Negative
TN = True negative, FP = False positive
TP = True positive, FN = False negative

- True Positive Rate (TPR) = Recall = Sensitivity = $\frac{\mathrm{TP}}{\mathrm{P}}$
- Probability of classifying a spam email as spam.

Confusion matrix

Given a dataset of P positive instances and N negative instances,

		Predicted class		
		1	0	
Actual	1	TP	FN	P
class	0	FP	TN	N

P = Positive, N = Negative
TN = True negative, FP = False positive
TP = True positive, FN = False negative

- Specificity $= \frac{\mathrm{TN}}{\mathrm{N}}$
- Probability of classifying a non-spam email as non-spam.
- False Positive Rate (FPR) = 1 - Specificity

Confusion matrix

Given a dataset of P positive instances and N negative instances,

		Predicted class		
		1	0	
Actual	1	TP	FN	P
class	0	FP	TN	N

P = Positive, N = Negative
TN = True negative, FP = False positive
TP = True positive, FN = False negative

- Precision = Positive Predictive Value (PPV) = $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$
- Probability that an email classified as spam is spam.
- Different from specificity!

Visualizing performance

Many different performance measures

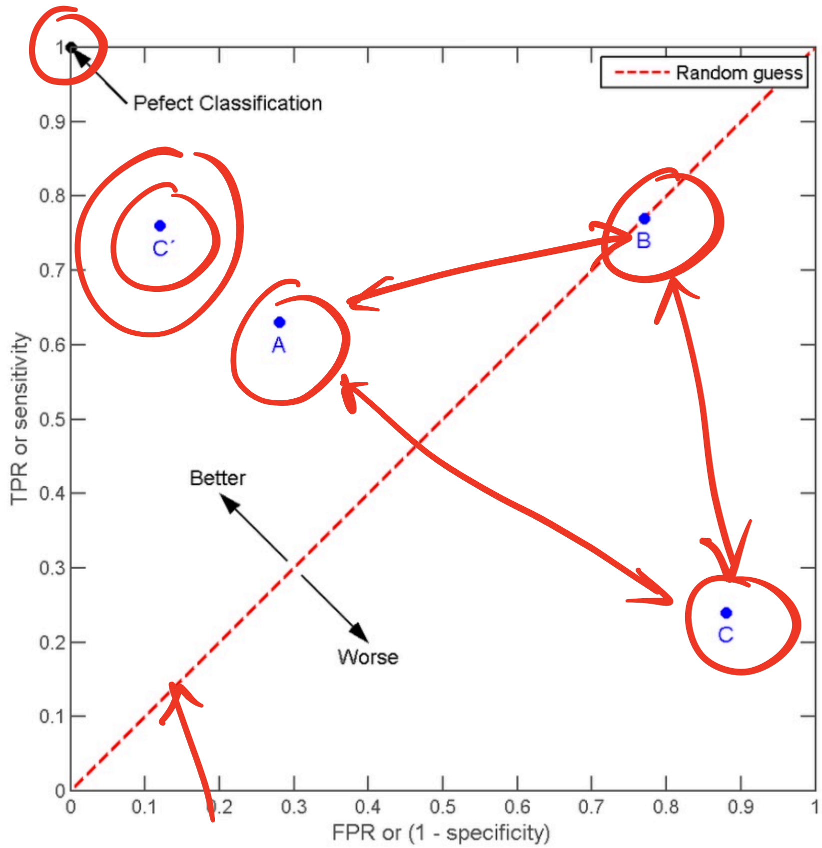
A			B			C			C′		
TP=63	FN=37	100	TP=77	FN=23	100	TP=24	FN=76	100	TP=76	FN=24	100
FP=28	TN=72	100	FP=77	TN=23	100	FP=88	TN=12	100	FP=12	TN=88	100
91	109	200	154	46	200	112	88	200	88	112	200

[1]

[1] https://en.wikipedia.org/wiki/Receiver_operating_characteristic

ROC curve

TPR (sensitivity) vs FPR (1-specificity)



2

- Best possible method is in the upper left corner.
- Random guess would fall on the diagonal.

https://en.wikipedia.org/wiki/Receiver_operating_characteristic

Sometimes we care about the confidence of predictions

- For linear models (*e.g.*, perceptron, logistic regression), the activation $a(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ gives us a ranking of the predictions.
- In the case of the perceptron, given $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ such that $a(\boldsymbol{x}_1) > a(\boldsymbol{x}_2) > 0$.
  - Both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are classified as positive.
  - Perceptron is more confident in the prediction for $\boldsymbol{x}_1$ than for $\boldsymbol{x}_2$.

ROC curve allows us to visualize the ranking of predictions

Building a ROC curve

- Sort the instances according to the confidence of instance being in class 1 (positive class).
  - How confident is the classifier that an email is spam?

Building a ROC curve

- Sort the instances according to the confidence of instance being in class 1 (positive class).
  - How confident is the classifier that an email is spam?
- Step through the sorted list from high to low confidence.
- As you step through, build a binary classifier that predicts the top k instances as belonging to class 1 and the remaining to class 0.
  - How aggressively do you want to mark a message as spam ?

Building a ROC curve

- Sort the instances according to the confidence of instance being in class 1 (positive class).
  - How confident is the classifier that an email is spam?
- Step through the sorted list from high to low confidence.
- As you step through, build a binary classifier that predicts the top k instances as belonging to class 1 and the remaining to class 0.
  - How aggressively do you want to mark a message as spam ?
- Compute the TPR (True Positive Rate) and FPR (False Positive Rate).
  - High up in the list: Low TPR and low FPR.
  - As we move down, TPR increases but so does FPR.

Building a ROC curve

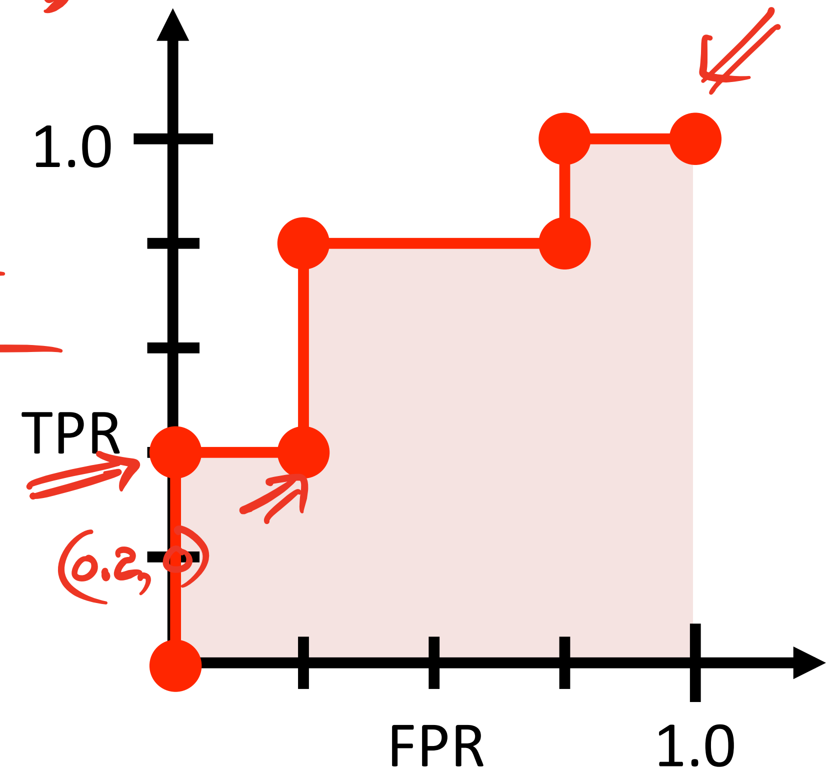
- Sort the instances according to the confidence of instance being in class 1 (positive class).
  - How confident is the classifier that an email is spam?
- Step through the sorted list from high to low confidence.
- As you step through, build a binary classifier that predicts the top k instances as belonging to class 1 and the remaining to class 0.
  - How aggressively do you want to mark a message as spam ?
- Compute the TPR (True Positive Rate) and FPR (False Positive Rate).
  - High up in the list: Low TPR and low FPR.
  - As we move down, TPR increases but so does FPR.
- Plot TPR vs FPR.

Example

$$TPR = \frac{TP}{\text{Positives}} = \frac{1}{5} \qquad FPR = 1 - \frac{TN}{\text{Negatives}} = 1 - \frac{5}{5} = 0$$

Instance	Confidence in positive	Actual class
9	0.99	1
7	0.98	1
1	0.72	0
2	0.70	1
6	0.65	1
10	0.51	0
3	0.39	0
5	0.24	1
4	0.11	0
8	0.01	0

$(\frac{2}{5}, 0)$

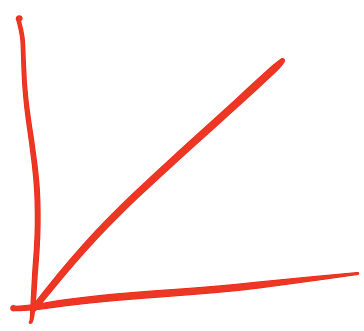


$$TPR = \frac{2}{5} \qquad FPR = \frac{1}{5}$$

Interpreting ROC curves

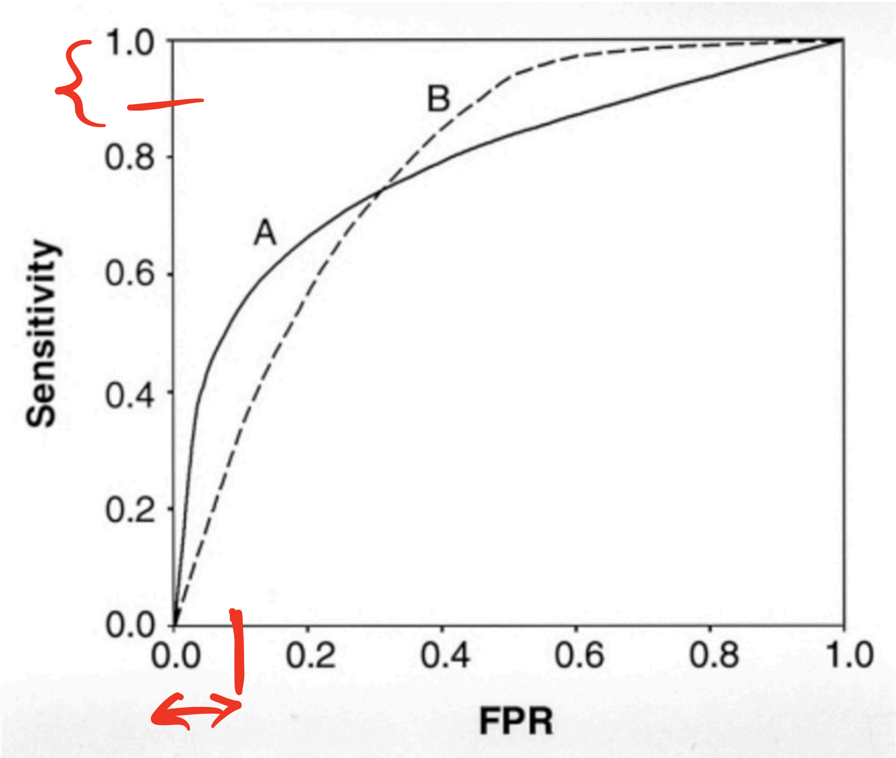
Area under the ROC Curve (AUC or AUROC)

- The probability that a classifier will rank a randomly chosen positive instance (instance with class label 1) higher than a randomly chosen negative one (instance with class label 0).
- A measure of the overall performance of a classifier.
- A random classifier has AUROC $= \frac{1}{2}$.
- A perfect classifier has AUROC $= 1$.



Interpreting ROC curves

Classifiers A and B with equal AUROC



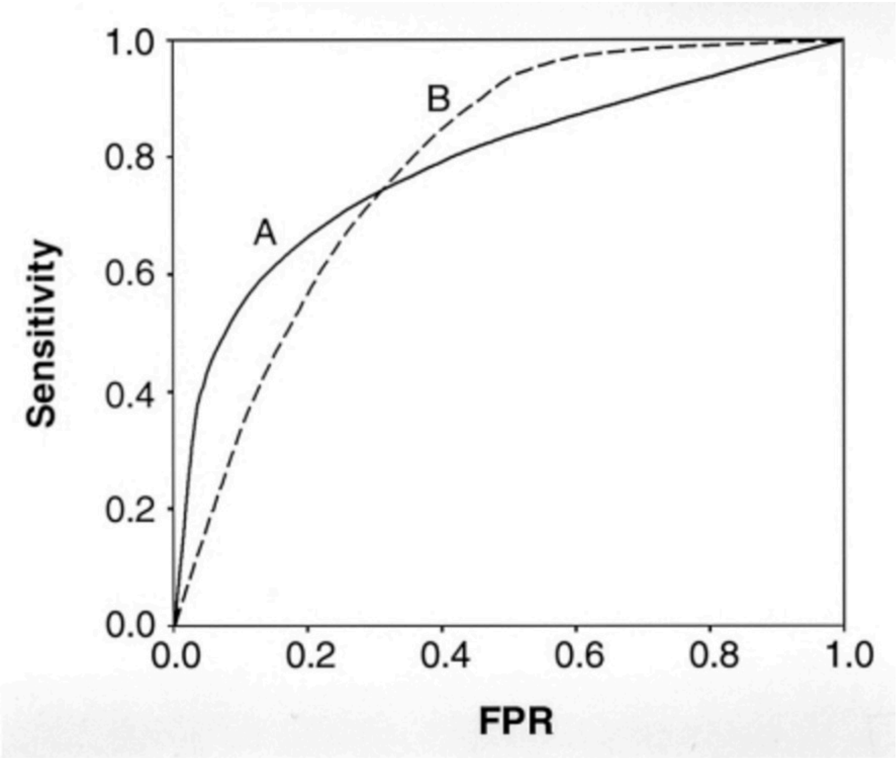
3

Which one do we choose?

Park et al. KJR 2004

Interpreting ROC curves

Classifiers A and B with equal AUROC



3

Which one do we choose?

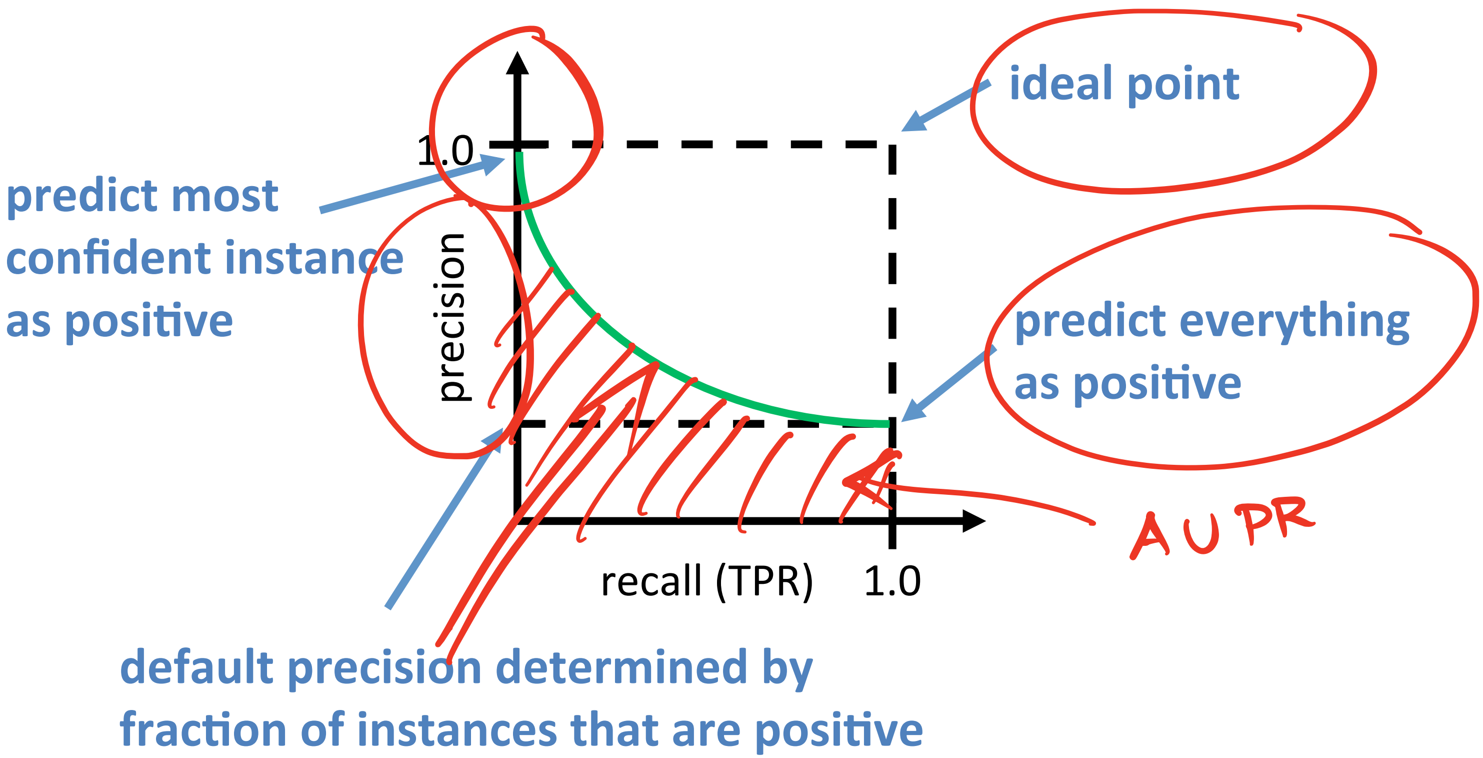
Depends on whether we care about sensitivity or FPR

Park et al. KJR 2004

Other trade-offs

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision-recall curve



Interpreting precision-recall curves

Area under the precision-recall curve (AUPR)

- Can compute the area under the precision-recall curve.

Interpreting precision-recall curves

F_1-score

$$F_1 = \frac{1}{\frac{1}{2}\frac{1}{\textsc{Precision}} + \frac{1}{2}\frac{1}{\textsc{Recall}}}$$

- Want F_1 to be high.
- The Harmonic mean of precision and recall.
- A single number that trades off precision and recall.
- If a classifier has high precision and low recall (or other way round), its F_1-score will be low.
- Encourages precision and recall values that are similar.

Summary

- Many ways to measure accuracy.
- Think carefully about which ones are most relevant.
- Often there are trade-offs
  - TPR vs FPR, Precision vs recall
- Graphical displays useful.
  - ROC and Precision-recall curves display performance at various levels of confidence.
  - Can be summarized by area under the curve (AUROC, AUPR) as well as numbers such as the F_1-score.