Logistic regression

CS 446 / ECE 449

2022-01-25 09:57:46 -0600 (4c2f15b)

Plan for today

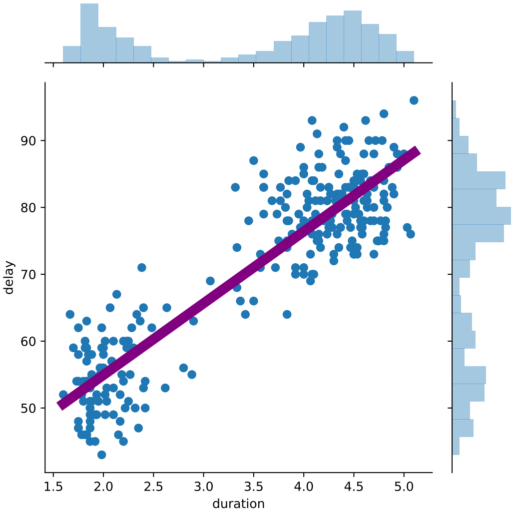
- Linear classifiers.
- ERM for classification.
- Solving the ERM problem.

Linear regression (last lecture) vs logistic regression (this lecture)

1. Start from training data $((\boldsymbol{x}_i, y_i))_{i=1}^n$, with $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$.
   Last lecture: $y_i \in \mathbb{R}$; **this lecture:** $y_i \in \{+1, -1\}$.
2. Model is a linear predictor: pick $\boldsymbol{w} \in \mathbb{R}^d$ with
$$\boldsymbol{x}_i \mapsto \boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i =: \hat{y}_i \approx y_i.$$
3. Choose $\boldsymbol{w}$ by minimizing empirical risk (average loss over training set):
$$\widehat{\mathcal{R}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^n \ell(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i, y_i).$$
   Last lecture: squared loss ℓ_{sq}; **this lecture:** logistic loss ℓ_{logistic}.
4. Basic method: gradient descent. Set $\boldsymbol{w}_0 = 0$, and thereafter
$$\boldsymbol{w}_{i+1} := \boldsymbol{w}_i - \eta\nabla\widehat{\mathcal{R}}(\boldsymbol{w}_i),$$
   where η is a learning rate (step size).
   Same in both lectures, however **least squares also had SVD solution**.

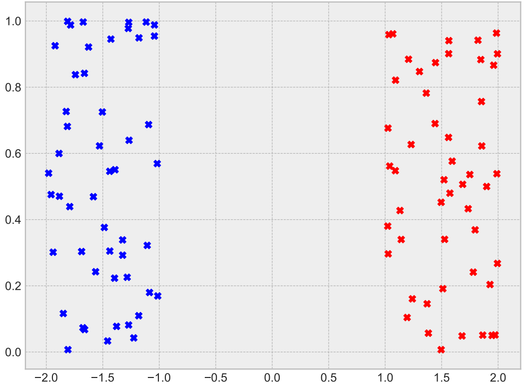
Linear regression

Last lecture, we studied regression;
the output/label space was $\mathbb{R}$.



Linear classification

Today, the goal is a classification;
the output/label space is discrete.

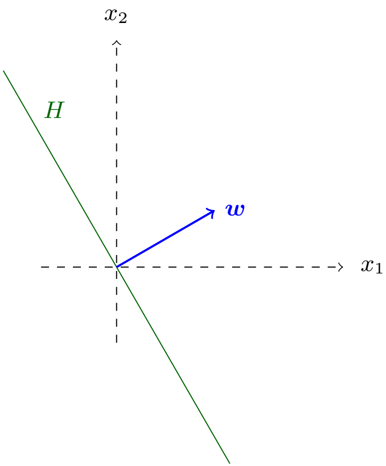


Notation

Binary classification means output space $\mathcal{Y} = \{-1, +1\}$.
A linear predictor $\boldsymbol{w} \in \mathbb{R}^d$ classifies according to $\operatorname{sign}(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}) \in \{-1, +1\}$.

Given $((\boldsymbol{x}_i, y_i))_{i=1}^n$ and a weight vector $\boldsymbol{w} \in \mathbb{R}^d$,
we want $\hat{y}_i := \operatorname{sign}\left(\boldsymbol{w}^\mathsf{T}\boldsymbol{x}_i\right) \in \{-1, +1\}$ and y_i to agree.

Geometry of linear classifiers



Given $\boldsymbol{w} \in \mathbb{R}^d$, predict with

$$\boldsymbol{x} \mapsto \operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x}) \in \{\pm 1\}.$$

Let H be the hyperplane orthogonal to $\boldsymbol{w}$:

$$H := \left\{ \boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{x}^\top \boldsymbol{w} = 0 \right\}.$$

H splits $\mathbb{R}^d$ into points we label positive,

$$\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} > 0\},$$

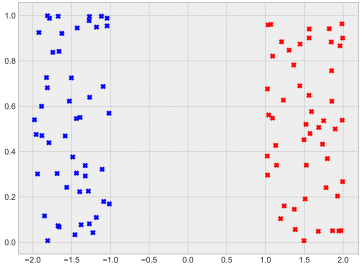
and points we label negative,

$$\{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{w}^\top \boldsymbol{x} < 0\}.$$

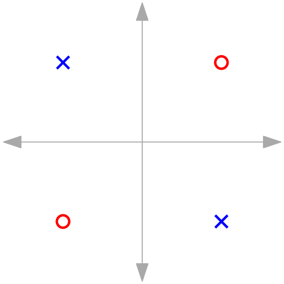
The decision boundary is H.

Linear separability

Is it always possible to find $\boldsymbol{w}$ with $\operatorname{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i) = y_i$?
I.e., is it always possible to find a (homogeneous) hyperplane which separates the data?



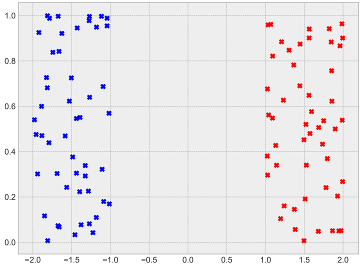
Linearly separable.



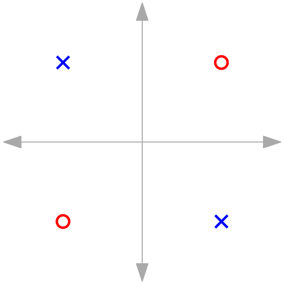
Not linearly separable.

Linear separability

Is it always possible to find $\boldsymbol{w}$ with $\mathrm{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i) = y_i$?
I.e., is it always possible to find a (homogeneous) hyperplane which separates the data?



Linearly separable.

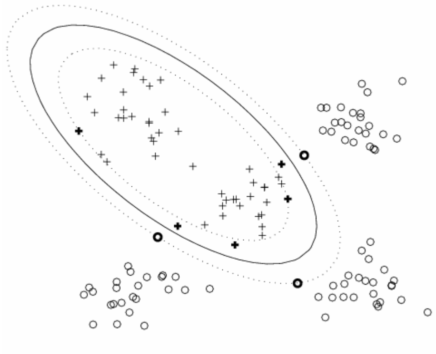


Not linearly separable.

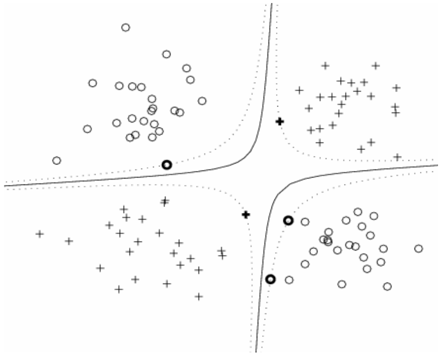
Lecture 4: adding features can make things separable.

Decision boundary with quadratic feature expansion

With old faithful data, we appended 1 to $\boldsymbol{x}$.
In general, we can map $\boldsymbol{x}$ to more exotic things, and separate more data.



elliptical decision boundary



hyperbolic decision boundary

We'll discuss this next lecture.

Finding linear classifiers with ERM

Finding linear classifiers with ERM

Why not try "`pytorch` meta-algorithm", with empirical risk

$$\arg\min_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[\operatorname{sign}(\boldsymbol{w}^\mathsf{T} x_i) \neq y_i] \ ?$$

- Discrete/combinatorial search;
  NP-hard in general; awkward for continuous optimization algorithms.

Relaxing the ERM problem

- **Step 1:** remove $\text{sign}(\cdot)$. Let's remove one source of discreteness:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\text{sign}(\boldsymbol{w}^{\mathsf{T}} x_i) \neq y_i] \qquad \longrightarrow \qquad \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left[y_i(\boldsymbol{w}^{\mathsf{T}} x_i) \leq 0\right] \quad .$$

  Are these equivalent?

- **Step 2:** remove $\mathbb{1}[\cdot]$. Rewrite the preceding as

$$\widehat{\mathcal{R}}_{\text{zo}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell_{\text{zo}}(y_i \boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_i) \qquad \text{where } \ell_{\text{zo}}(z) = \mathbb{1}[z \leq 0].$$

  Here, $y_i(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_i)$ is the (unnormalized) margin of $\boldsymbol{w}$ on example i.

  Next let's replace ℓ_{zo} with something continuous!

(Side note: squared loss had two arguments, today we'll use one. We'll discuss this point next lecture.)

Logistic loss

We want to choose a nice loss ℓ in

$$\widehat{\mathcal{R}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \boldsymbol{w}^\top \boldsymbol{x}_i).$$

Key desired properties:

- ℓ is continuous (for sake of optimization);
- ℓ prefers correct classifications:
  if $y_i \boldsymbol{w}^\top \boldsymbol{x}_i > 0$ (correct), then $\ell(y_i \boldsymbol{w}^\top \boldsymbol{x}_i) < \ell(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)$.

Logistic loss

We want to choose a nice loss ℓ in

$$\widehat{\mathcal{R}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i \boldsymbol{w}^\top \boldsymbol{x}_i).$$

Key desired properties:

- ℓ is continuous (for sake of optimization);
- ℓ prefers correct classifications:
  if $y_i \boldsymbol{w}^\top \boldsymbol{x}_i > 0$ (correct), then $\ell(y_i \boldsymbol{w}^\top \boldsymbol{x}_i) < \ell(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)$.

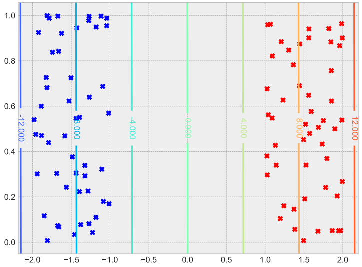
Examples.

- Squared loss, written in margin form: $\ell_{\mathsf{ls}}(z) := \frac{1}{2}(1-z)^2$; note

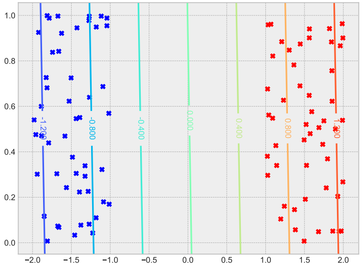
  $$2\ell_{\mathsf{ls}}(y\hat{y}) = (1-y\hat{y})^2 = y^2(1-y\hat{y})^2 = (y-\hat{y})^2.$$

  Squared loss doesn't just push $y_i\hat{y}_i$ positive; it wants $y_i\hat{y}_i = 1$!
- Logistic loss: $\ell_{\log}(z) = \ln(1+\exp(-z))$.
  This one doesn't care so long as $z = y\hat{y} > 0$.

Squared and logistic losses on linearly separable data I

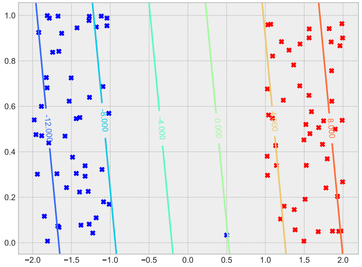


Logistic loss.

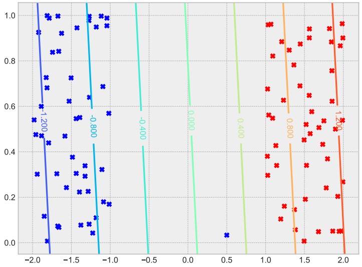


Squared loss.

Squared and logistic losses on linearly separable data II

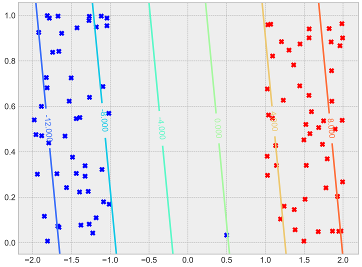


Logistic loss.

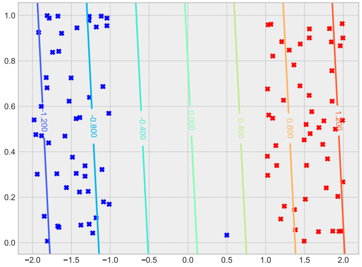


Squared loss.

Squared and logistic losses on linearly separable data II



Logistic loss.



Squared loss.

(Math note: it's easy to prove this.)

Least squares and logistic ERM

Least squares and logistic ERM

Least squares:

- Take gradient of $2n\widehat{\mathcal{R}}_{\mathsf{ls}}(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$, set to 0;
  obtain normal equations $\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^{\mathsf{T}}\boldsymbol{y}$.
  Suffice to take OLS solution $\hat{\boldsymbol{w}}_{\mathsf{ols}} = \boldsymbol{X}^{+}\boldsymbol{y}$.
- Alternatively, gradient descent.

Logistic loss:

- Unclear how to solve

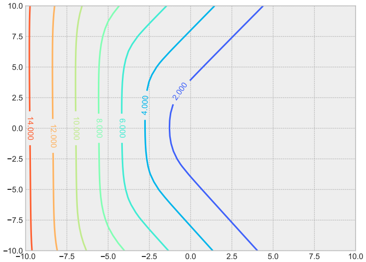
$$\nabla_{\boldsymbol{w}}\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}}\frac{1}{n}\sum_{i=1}^{n}\ln(1+\exp(-y_i\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}_i)) = 0.$$

- Gradient descent still fine!

Given an empirical risk $\widehat{\mathcal{R}} : \mathbb{R}^p \to \mathbb{R}$, gradient descent is the iteration

$$\boldsymbol{w}_{i+1} := \boldsymbol{w}_i - \eta_i \nabla_{\boldsymbol{w}} \widehat{\mathcal{R}}(\boldsymbol{w}_i),$$

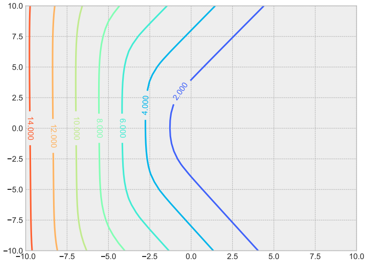
where $\boldsymbol{w}_0$ is given, and η_i is a learning rate (step size).
Gradient descent goes down the contours of $\widehat{\mathcal{R}}$:



Given an empirical risk $\widehat{\mathcal{R}} : \mathbb{R}^p \to \mathbb{R}$, gradient descent is the iteration

$$\boldsymbol{w}_{i+1} := \boldsymbol{w}_i - \eta_i \nabla_{\boldsymbol{w}} \widehat{\mathcal{R}}(\boldsymbol{w}_i),$$

where $\boldsymbol{w}_0$ is given, and η_i is a learning rate (step size).
Gradient descent goes down the contours of $\widehat{\mathcal{R}}$:



Remarks.

- In the convexity lecture, we'll show this works for linear and logistic regression.
- For deep networks, $\boldsymbol{w}_0$ is random, and η_i is highly tuned/varying.

For logistic loss and linear predictors, typically $\boldsymbol{w}_0 = 0$, and

For logistic loss and linear predictors, typically $\boldsymbol{w}_0 = 0$, and

$$\boldsymbol{w}_{i+1} := \boldsymbol{w}_i - \eta_i \nabla_{\boldsymbol{w}} \widehat{\mathcal{R}}_{\log}(\boldsymbol{w}_i) = \boldsymbol{w}_i - \frac{\eta_i}{n} \sum_{j=1}^{n} \ell'_{\log}(y_j \boldsymbol{x}_j^{\mathsf{T}} \boldsymbol{w}_i) y_j \boldsymbol{x}_j,$$

where $\ell'_{\log}(z) = \frac{-1}{1+\exp(z)}$.

For logistic loss and linear predictors, typically $\boldsymbol{w}_0 = 0$, and

$$\boldsymbol{w}_{i+1} := \boldsymbol{w}_i - \eta_i \nabla_{\boldsymbol{w}} \widehat{\mathcal{R}}_{\log}(\boldsymbol{w}_i) = \boldsymbol{w}_i - \frac{\eta_i}{n} \sum_{j=1}^{n} \ell'_{\log}(y_j \boldsymbol{x}_j^{\mathsf{T}} \boldsymbol{w}_i) y_j \boldsymbol{x}_j,$$

where $\ell'_{\log}(z) = \frac{-1}{1+\exp(z)}$.

But who cares, `pytorch` does it for us:

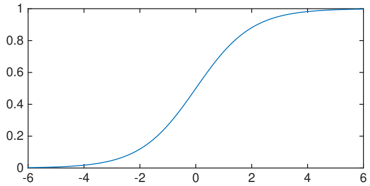
```
def GD(X, y, loss, step = 0.1, n_iters = 10000):
    w = torch.zeros(X.shape[1], requires_grad = True)
    for i in range(n_iters):
        l = loss(X, y, w).mean()
        l.backward()

        with torch.no_grad():
            w -= step * w.grad
            w.grad.zero_()

    return w
```

"Logistic" "regression"?

The (negative) derivative $-\ell'_{\log}(z) = \frac{1}{1+e^z}$ is the logistic function.



It can suggest a probability / confidence of the output label.
We'll revisit this next lecture.

Warning: many treat these explicitly as confidences, but they are not.

Summary for today

- Linear classifiers.
- ERM for classification.
- Solving the ERM problem.

(Appendix.)

Logistic risk and separation

If there exists a perfect linear separator,
empirical logistic risk minimization should find it.

Theorem.

Logistic risk and separation

If there exists a perfect linear separator,
empirical logistic risk minimization should find it.

Theorem. If there exists $\bar{\boldsymbol{w}}$ with $y_i \bar{\boldsymbol{w}}^\mathsf{T} \boldsymbol{x}_i > 0$ for all i,
then every $\boldsymbol{w}$ with $\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) < {}^{\ln(2)}/_{2n} + \inf_{\boldsymbol{v}} \widehat{\mathcal{R}}_{\log}(\boldsymbol{v})$,
also satisfies $y_i \boldsymbol{w}^\mathsf{T} \boldsymbol{x}_i > 0$.

Logistic risk and separation

If there exists a perfect linear separator,
empirical logistic risk minimization should find it.

Theorem. If there exists $\bar{\boldsymbol{w}}$ with $y_i \bar{\boldsymbol{w}}^\top \boldsymbol{x}_i > 0$ for all i,
then every $\boldsymbol{w}$ with $\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) < {}^{\ln(2)}/_{2n} + \inf_{\boldsymbol{v}} \widehat{\mathcal{R}}_{\log}(\boldsymbol{v})$,
also satisfies $y_i \boldsymbol{w}^\top \boldsymbol{x}_i > 0$.

Proof.
Step 1: low risk implies few mistakes. For any $\boldsymbol{w}$ with $y_j \boldsymbol{w}^\top \boldsymbol{x}_j \le 0$ for some j,

$$\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) \ge \frac{1}{n} \ln(1 + \exp(-y_j \boldsymbol{w}^\top \boldsymbol{x}_j)) \ge \frac{\ln(2)}{n}.$$

By contrapositive, any $\boldsymbol{w}$ with $\widehat{\mathcal{R}}_{\log}(\boldsymbol{w}) < {}^{\ln(2)}/_{n}$ makes no mistakes.
Step 2: $\inf_{\boldsymbol{v}} \widehat{\mathcal{R}}_{\log}(\boldsymbol{v}) = 0$. Note:

$$0 \le \inf_{\boldsymbol{v}} \widehat{\mathcal{R}}_{\log}(\boldsymbol{v}) \le \inf_{r>0} \frac{1}{n} \sum_{i=1}^{n} \ln(1 + \exp(-r y_i \bar{\boldsymbol{w}}^\top \boldsymbol{x}_i)) = 0.$$

$\square$

Remark. We didn't prove that gradient descent finds such a predictor, but that in turn is an easy consequence of the above and one bound from the upcoming convexity lecture.

The Perceptron algorithm (streamed data!)

Start from $\boldsymbol{w}_0 = 0$, and thereafter rotate towards $y_i \boldsymbol{x}_i$ if wrong:

$$\boldsymbol{w}_{i+1} := \boldsymbol{w}_i + y_{i+1}\boldsymbol{x}_{i+1}\mathbb{1}[y_{i+1}\boldsymbol{w}_i^\mathsf{T}\boldsymbol{x}_{i+1} \leq 0].$$

Remark: this is a specific subgradient of $\partial_{\boldsymbol{w}} \max\{0, -y_{i+1}\boldsymbol{x}_{i+1}^\mathsf{T}\boldsymbol{w}_i\}$.

Theorem (perceptron convergence, Novikoff '62). Suppose there exists $\boldsymbol{u}$ with $\|\boldsymbol{u}\|_2 = 1$ and $\boldsymbol{u}^\mathsf{T}\boldsymbol{x}_i y_i \geq \gamma > 0$, and $\|\boldsymbol{x}_i\| \leq 1$. Then perceptron makes at most $1/\gamma^2$ mistakes:

$$\sum_{i<t} \mathbb{1}[\boldsymbol{w}_i^\mathsf{T}\boldsymbol{x}_{i+1}y_{i+1} \leq 0].$$

Proof. Define mistake set $\mathcal{M} := \{i < t : \mathbb{1}[\boldsymbol{w}_i^\mathsf{T}\boldsymbol{x}_{i+1}y_{i+1}]\}$. Note

$$\|\boldsymbol{w}_t\| \geq \boldsymbol{w}_t^\mathsf{T}\boldsymbol{u} = \sum_{i\in\mathcal{M}} y_i \boldsymbol{x}_i^\mathsf{T}\boldsymbol{u} \geq \gamma|\mathcal{M}|.$$

On the other hand, by induction,

$$\begin{aligned}
\|\boldsymbol{w}_t\|^2 &= \|\boldsymbol{w}_{t-1}\|^2 + y_t\boldsymbol{x}_t^\mathsf{T}\boldsymbol{w}_{t-1}\mathbb{1}[t-1\in\mathcal{M}] + \|y_i\boldsymbol{x}_i\mathbb{1}[t-1\in\mathcal{M}]\|^2 \\
&\leq \|\boldsymbol{w}_{t-1}\|^2 + \mathbb{1}[t-1\in\mathcal{M}] \leq \cdots \leq |\mathcal{M}|.
\end{aligned}$$

Together, $\gamma|\mathcal{M}| \leq \|\boldsymbol{w}_t\| \leq \sqrt{|\mathcal{M}|}$, and $|\mathcal{M}| \leq 1/\gamma^2$. $\square$